

PEMROGRAMAN

Expert Advisor

Mahir membuat sistem perdagangan otomatis pada platform MetaTrader 4 dengan MQL

BASIC EDITION



VisionEA



Yulianto Hiu

Pemrograman Expert Advisor

Mahir membuat sistem perdagangan otomatis pada
platform MetaTrader 4 dengan MQL

Basic Edition

Yulianto Hiu

DAFTAR ISI

PENGANTAR

Pengenalan MQL

- Pengenalan MetaEditor

- Operations and Expression

- Functions

 - Custom Function

 - Common Function

- Standard Default Function

MetaTrader 4

- Layout MetaTrader 4

- Memulai Penggunaan MetaEditor

- Memahami FrameWork dari Expert Advisor di MQL4

- Start Debugging Expert Advisor

Membuka Posisi Trading

- Bid, Ask & Spread

- Order Types

- Membuka Pesanan (order)

- Menghitung Harga StopLoss dan TakeProfit

- Mengambil informasi Pesanan

- Menutup Pesanan

Membuat Aplikasi Perdagangan Otomatis Sederhana

- Penjelasan Kebutuhan Aplikasi

- Penulisan Kode Aplikasi

- Penjelasan Kode

PENGANTAR

Pertumbuhan teknologi digital yang pesat mendorong pertumbuhan di segala sektor perindustrian termasuk pasar valuta asing (forex). Forex telah dengan cepat menjadi salah satu pasar paling populer untuk diperdagangkan. Karena kemudahannya digunakan dan kelebihan lainnya seperti jam kerjanya 24 jam, leverage tinggi dan persyaratan margin rendah, sehingga ribuan orang biasa telah menjadi pedagang aktif dan angka ini terus bertumbuh.

MetaTrader 4 (MT4) telah menjadi salah satu platform perdagangan paling populer untuk perdagangan forex yang dikembangkan oleh MetaQuotes Software Corporation, MetaTrader digunakan oleh ratusan broker ternama di seluruh dunia, termasuk FBS, Hot Forex, IC Markets, Pepperstone.

Popularitas MetaTrader berasal dari kenyataan bahwa itu gratis, didukung pialang, dan menyertakan banyak manfaat alat analisis teknis. Tapi mungkin alasan terbesar kesuksesan MetaTrader adalah pada kekuatan MetaQuotes Language 4 (MQL4).

MQL4 adalah bahasa bawaan baru untuk pemrograman strategi perdagangan. Bahasa ini memungkinkan Anda membuat Expert Advisor (robot forex) yang menjadikan manajemen perdagangan terotomatisasi dan sangat cocok untuk menerapkan strategi perdagangannya sendiri. Selain itu, orang dapat menggunakan MQL4 untuk membuat Custom Indicators, Scripts, and Libraries sendiri.

MetaEditor 4 yang menyoroti berbagai konstruksi bahasa MQL4 digunakan untuk menulis kode program. Ini membantu pengguna mengarahkan diri mereka dalam teks sistem pakar dengan mudah.

Ada banyak faktor yang harus dipertimbangkan saat pemrograman otomatis dengan strategi perdagangan baik, dan MetaTrader sendiri memiliki banyak keistimewaan yang perlu disadari oleh seorang programmer. Butuh waktu puluhan jam untuk

mengatasi masalah dan berlatih untuk mempelajari teknik diperlukannya untuk memprogram Exper Advisors.

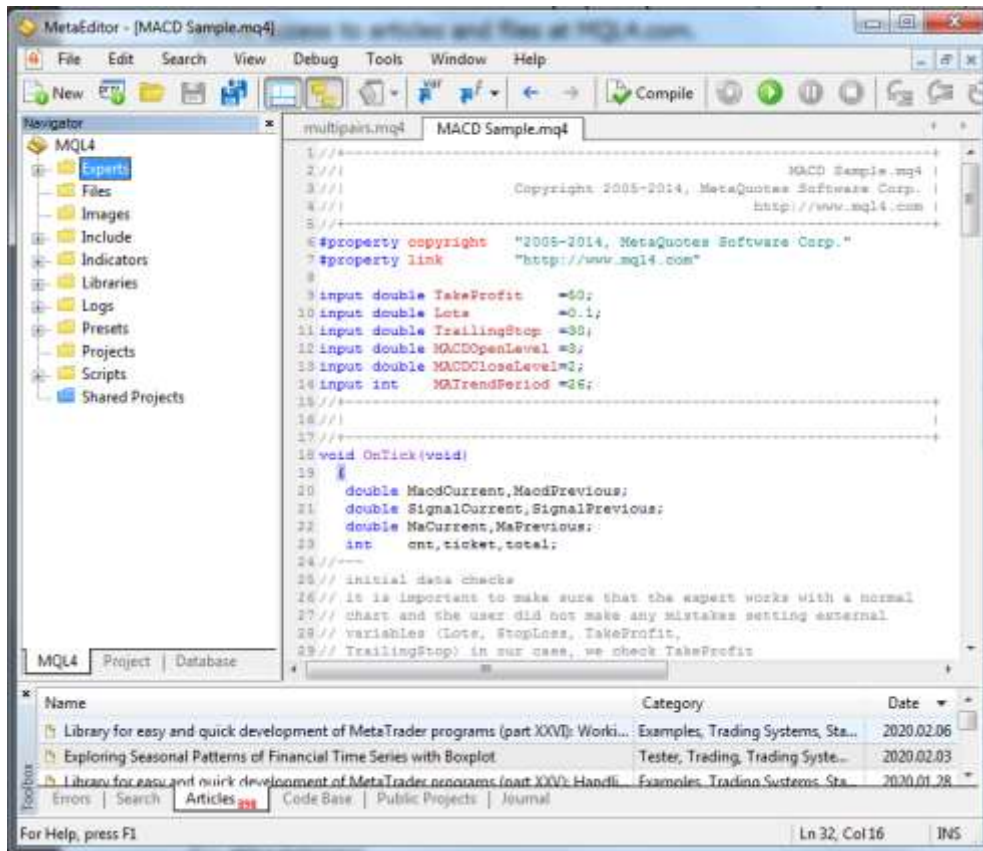
e-book ini diharapkan dapat memperpendek waktu belajar untuk programmer pemula Expert Advisor. Di sini saya akan menyajikan banyak tips dan trik yang telah saya pelajari dalam ratusan jam yang telah saya habiskan dalam pemrograman Expert Advisor ini selama beberapa tahun terakhir.

Pengenalan MQL

Pengenalan MetaEditor

adalah Lingkungan Pengembangan Terpadu (IDE) untuk MQL yang disertakan bersama MetaTrader. Ini termasuk referensi yang berguna, alat pencarian dan pelengkapan otomatis yang membuat pemrograman dalam MQL jauh lebih mudah.

Jendela Editor (editor window) memungkinkan Anda membuka banyak file sekaligus. Anda dapat meminimalkan, memaksimalkan, dan tab antara beberapa jendela yang terbuka. Jendela Navigator menawarkan penjelajahan dan referensi file yang bermanfaat fitur. Jendela Toolbox menampilkan konten bantuan, kesalahan kompilasi, hasil pencarian file, dan akses online ke artikel dan file di MQL4.com.



Program yang ditulis dalam MetaQuotes Language 4 memiliki fitur dan tujuan yang berbeda:

1. Expert Advisor

adalah program perdagangan otomatis yang ditulis dalam MQL. Expert Advisor (umumnya disingkat EA) dapat menempatkan, memodifikasi dan menutup pesanan sesuai dengan algoritma sistem perdagangan. EA umumnya menggunakan indikator untuk menghasilkan sinyal perdagangan. Indikator-indikator ini dapat menjadi indikator yang menyertai MetaTrader, atau mereka bisa menjadi indikator khusus (custom indicators). EA mulai dijalankan dengan setiap tick yang masuk untuk pada simbol tertentu yang diberikan. EA disimpan dalam folder `terminal_directory \ experts \`

2. Custom Indicator (Indikator Khusus)

adalah indikator teknis yang ditulis secara independen selain yang sudah terintegrasi ke dalam terminal klien. Seperti built-in indikator, mereka tidak

dapat berdagang secara otomatis dan dimaksudkan untuk mengimplementasikan fungsi analitis saja. Indikator Khusus disimpan dalam folder `terminal_directory \ indicators \`

3. Script

adalah program yang ditujukan untuk satu eksekusi beberapa tindakan. Tidak seperti Expert Advisors, Script tidak dijalankan dengan cara tickwise, tetapi berdasarkan permintaan.

Skrip disimpan dalam folder `terminal_dictionary \ scripts \`

4. Library



adalah sekumpulan fungsi khusus yang berisi program yang paling sering digunakan. Library tidak dapat memulai eksekusi dengan sendirinya. Library direkomendasikan untuk disimpan di `terminal_directory \ libraries \`

5. Included file

adalah teks sumber dari blok program kustom yang paling sering digunakan. File tersebut dapat dimasukkan ke dalam teks sumber EA, script, indikator khusus, dan library pada tahap kompilasi. Penggunaan file yang disertakan lebih disukai daripada penggunaan library karena beban tambahan yang terjadi pada fungsi panggilan library. File yang disertakan disarankan untuk disimpan di `terminal_directory \ include \`

Format File

File dengan ekstensi **.mq4** adalah file kode sumber. Ini adalah file yang kita edit di MetaEditor. Saat file **.mq4** dikompilasi akan dihasilkan file baru dengan ekstensi **.ex4**.

File dengan ekstensi **.ex4** adalah file yang dapat dieksekusi. Ini adalah file yang kami jalankan di MetaTrader. File ini tidak dapat dibuka di MetaEditor. Jika Anda hanya memiliki file **.ex4** untuk EA atau indikator, ikon di sebelah nama file di jendela Navigator MetaTrader's akan berwarna abu-abu  , namun jika Anda juga memiliki file ekstensi **.mq4** maka ikon pada nama file akan terlihat aktif warna kuning 

File dengan ekstensi **.mqh** adalah file include (included file). File-file ini berisi fungsi-fungsi yang dibuat pengguna direferensikan dalam file **.mq4**. Selama kompilasi, kompiler akan memasukkan isi dari file include ke file **.ex4**.

Ekstensi **.mq4** digunakan untuk file templat. Dimana file-file ini dapat dibuka di MetaTrader, tipe file ini tidak terkait dengan program di Windows. Templat digunakan untuk membuat file baru menggunakan Wizard EA di MetaEditor.

Ekstensi **.log** digunakan untuk menyimpan aktifitas log yang dihasilkan oleh EA. Ini akan bermanfaat untuk debugging EA Anda.

Konsep-konsep Dasar

Ada beberapa hal dasar dalam pemrograman secara umum yang perlu kita ketahui sebelum memulai pemrograman. Jika Anda sudah berpengalaman, Anda bisa melewati bagian ini.

Syntax

Jika Anda terbiasa dengan pemrograman dalam bahasa seperti C++, PHP atau salah satu dari banyak bahasa pemrograman yang berasal dari syntax C, Anda akan sangat nyaman dalam pemrograman MQL. Jika sebelumnya Anda pengalaman pemrograman dalam bahasa seperti Visual Basic atau Delphi, maka Anda mungkin perlu membuat beberapa penyesuaian.

Dalam MQL, setiap pernyataan diakhiri dengan tanda titik koma. Ini disebut ungkapan. Sebuah ekspresi dapat menjangkau beberapa baris, tetapi harus ada titik koma di bagian akhir.

```
input double TrailingStop =30;  
input double MACDOpenLevel =3, MACDCloseLevel=2;
```

Ada beberapa pengecualian untuk blok kode yang terdiri dari beberapa baris di buka dengan tanda { dan ditutup dengan }

Umumnya digunakan untuk operator kontrol (if, switch) dan operator pengulangan (for, while) dan untuk deklarasi fungsi (function)

```
If (a == true)
{
    Print ("Data sudah sesuai");
}
```

Comments

berguna untuk mendokumentasikan kode Anda, serta untuk sementara menghapus kode pengujian dan debugging. Anda dapat mengomentari satu baris dengan dua garis miring:

```
// ini adalah baris komen 1baris
```

Dan untuk mengomentari lebih dari 1 baris bisa menggunakan /* diawal dan ditutup dengan */

```
/*      Ini adalah baris komentar dalam beberapa baris
        semua komentar bisa ditulis di dalam sini      */
```

Identifiers

Identifier adalah nama yang diberikan ke variabel dan fungsi khusus. Identifier dapat berupa kombinasi dari angka, huruf, dan karakter garis bawah (_). Pengidentifikasi dapat dibuat sepanjang hingga 31 karakter.

Identifier dalam MQL peka terhadap huruf besar-kecil. Penulisan nama identifier TakeProfit berbeda dengan takeprofit.

Variables

Variabel adalah unit penyimpanan dasar dari setiap bahasa pemrograman. Variabel menyimpan data yang diperlukan program kita agar bisa berfungsi.

Variable harus dideklarasikan sebelum digunakan. Deklarasikan variable untuk menentukan data type yang akan digunakan. Data type adalah jenis informasi yang akan disimpan oleh variable, seperti text/string, bilangan bulat, desimal, tanggal, warna dan sebagainya.

- `int` – adalah angka dari 0 hingga 9; nol tidak boleh menjadi angka pertama
- `double` – angka pecahan seperti 1.3289, 0.12 atau -1.236. Gunakan ini untuk data harga, atau dalam ekspresi matematika yang melibatkan pembagian
- `string` – string teks seperti "Ayo belajar koding EA!". string harus dikelilingi oleh tanda kutip ganda.
- `bool` – boolean memiliki nilai `true` atau `false` atau sering juga menggunakan 1 dan 0.
- `datetime` – Nilai waktu dan tanggal seperti 2009.01.01 00:00. Secara internal, variabel `datetime` direpresentasikan sebagai jumlah detik yang berlalu sejak 1 Januari 1970.
- `color` – Konstanta mewakili warna, seperti `Merah` atau `DarkSlateBlue`. Ini umumnya digunakan untuk mengubah warna indikator atau objek.

Contoh deklarasi variabel:

```
int n = 30;
```

```
double price = 0.0;
```

```
price = 1.3188;
```

```
datetime tanggal = D'2019.01.01';
```

Variable Scope

Variable scope menentukan sejauh mana cakupan sebuah variable dalam suatu function, dan berapa lama variable itu tersimpan di memori. Dalam MQL, ruang lingkup bisa lokal atau global. Variabel lokal juga bisa statis.

Variable **local** adalah variable yang dideklarasikan di dalam sebuah function. Variable ini tidak dikenal diluar function. Sehingga saat keluar dari function, nilai variable ini akan dibersihkan dari memori.

Variable **global** adalah variable yang dideklarasikan di luar function dan bisa dikenal oleh semua function.

Variable **static** adalah variable yang dideklarasikan dengan menggunakan keyword static diawal data type variable. Variable static bisa menggunakan scope local maupun global dimana masa penyimpanan data di dalam memori bertahan lebih lama dan tidak akan hilang bila keluar dari sebuah function.

Operations and Expressions

Beberapa karakter dan urutan karakter sangat penting. Ini disebut simbol operasi, misalnya:

+ - * / %	Simbol operasi aritmatika
&&	Simbol operasi logis
= += *=	Operator penugasan Karakter

Operators

- Operator gabungan {}

Satu atau lebih operator dari jenis apa pun, terlampir dalam kurung kurawal

```
{}
```

```
if (x == 0)
```

```
{
```

```
    Print ("Nilai x = ", x );
```

```
}
```

- Operator ekspresi (;)

Ekspresi apa pun yang berakhir dengan titik koma (;)

```
x = 3;
```

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

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

- **if – else** adalah operator kondisional

Digunakan ketika perlu untuk membuat pilihan

```
if (a > 5)
```

```
{
```

```
    Print ("Nilai a lebih besar dari 5 ");
```

```
}else{
```

```
        Print ("Nilai a kurang atau sama dengan 5");  
    }
```

- **switch** adalah operator pemilihan. Melewati kontrol ke operator, yang sesuai dengan nilai ekspresi.

```
switch(x)  
{  
    case 'A': Print("A");  
             break;  
    case 'B':  
    case 'C': Print("B atau C");  
             break;  
    default: Print("BUKAN A, B atau C");  
            break;  
}
```

- **while** adalah operator pengulangan
Melakukan operator hingga ekspresi yang diperiksa menjadi salah. Ekspresi diperiksa sebelum setiap iterasi

```
while (n < 10)  
{  
    total += harga;  
    n++;  
}
```

- **for** adalah operator pengulangan
Melakukan operator hingga ekspresi yang diperiksa menjadi salah. Ekspresi diperiksa sebelum setiap iterasi.

```
for (int n=1; n<10; n++)  
{  
    Print ("Nilai N adalah ", n);  
}
```

Constants

adalah nilai data yang tidak pernah berubah. Misalnya angkanya 5 adalah konstanta integer, huruf 'A' adalah konstanta karakter, dan 2009.01.01 adalah konstanta datetime untuk 1 Januari 2009.

SQL memiliki berbagai konstanta standar untuk hal-hal seperti data harga, periode grafik, warna, dan operasi perdagangan. Misalnya PERIOD_H1 adalah konstanta untuk kerangka waktu grafik H1, OP_SELL mengacu pada pesanan pasar jual, dan clr_red adalah konstanta warna untuk warna merah.

Anda bahkan dapat membuat konstanta sendiri menggunakan #define preprocessor.

Contoh penggunaan konstanta:

```
#define PI          3.14
#define PERUSAHAAN "MetaQuotes Software Corp."
```

Functions

adalah blok bangunan bahasa pemrograman modern. Function adalah blok kode yang dirancang untuk melakukan tugas tertentu, seperti menempatkan pesanan atau menghitung stop loss. MQL memiliki banyak function bawaan untuk berbagai kegunaan mulai dari indikator teknis hingga penempatan pesanan. Sebuah function bisa dipanggil oleh function lain.

Custom Functions

Ada 2 jenis function, yaitu function yang memiliki nilai pengembalian atau tidak ada nilai pengembalian. Function dengan nilai pengembalian harus menentukan data type nilai pengembalian pada bagian depan nama function, sedangkan function tanpa nilai pengembalian menggunakan void type.

Contoh function tanpa nilai pengembalian :

```
void errmsg (string s)
{
    print ("error: " + s);
}
```

nama function adalah **errmsg**, tidak memiliki nilai pengembalian yang ditandai dengan void type di depan nama function. Sebuah function diawali dengan tanda { dan ditutup dengan }. Diantara { dan } berisi intruksi sesuai dengan tujuan atau nama function.

Contoh function dengan pengembalian nilai :

```
double hitungTotal (double a, double b)
{
    return (a+b);
}
```


atau

```
double hitungTotal (double a, double b)
{
    double total = 0.0;
    total = a + b;
    return (total);
}
```

nama function adalah **hitungTotal**, nilai pengembalian dengan data type double. Nilai pengembalian menggunakan operator **return**.

Common Function

fungsi bawaan dengan tujuan umum yang sering digunakan adalah :

- Alert : menampilkan pesan pada window terpisah
- Comment : tampilan comment di sebelah pojok kiri atas layar chart.
- ExpertRemove : menghentikan EA dan menghapus EA dari layar chart.
- PlaySound : mengaktifkan suara dari file.
- Print : menampilkan pesan di log.

Standard Default Function

Account Information

Berfungsi untuk mengembalikan parameter pada akun yang digunakan saat ini.

- **AccountInfoDouble** : mengembalikan nilai tipe ganda dari properti akun
- **AccountInfoInteger** : mengembalikan nilai tipe integer (bool, int atau long) dari properti akun.
- **AccountInfoString** : mengembalikan nilai tipe string dari properti akun.
- **AccountBalance** : mengembalikan nilai saldo pada akun.
- **AccountCredit** : mengembalikan nilai kredit pada akun.
- **AccountCompany** : mengembalikan nama perusahaan broker di mana akun didaftarkan.
- **AccountCurrency** : mengembalikan nama mata uang pada akun.
- **AccountEquity** : mengembalikan nilai ekuitas pada akun.
- **AccountFreeMargin** : mengembalikan nilai margin pada akun.
- **AccountFreeMarginCheck** : mengembalikan margin yang tersisa setelah dibukanya sebuah posisi pada harga yang telah ditentukan pada akun.
- **AccountLeverage** : mengembalikan leverage akun.
- **AccountMargin** : mengembalikan nilai margin pada akun.
- **AccountName** : mengembalikan nama akun.
- **AccountNumber** : mengembalikan nomor akun.
- **AccountProfit** : mengembalikan nilai keuntungan (profit) pada akun.
- **AccountServer** : mengembalikan nama server yang tersambung.

Conversion Functions

Fungsi-fungsi yang digunakan untuk mengkonversi data dari 1 format ke format data lainnya.

- `DoubleToString` : mengkonversi bilang menjadi tipe data string
- `EnumToString` : mengkonversi data enumeration menjadi tipe data string
- `NormalizeDouble` : pembulatan bilangan ke dalam format desimal tertentu
- `StringToDouble` : mengubah string yang berisi bilangan menjadi format bilangan double
- `StringToInteger` : mengubah string yang berisi bilangan menjadi format bilangan integer
- `StringToTime` : mengubah string yang berisi waktu atau tanggal menjadi waktu dengan tipe data datetime

Mathematical Functions

Fungsi untuk perhitungan atau matematika

- `MathAbs` : mengubah semua bilangan menjadi bilangan positif
- `MathCeil` : pembulatan bilangan keatas menjadi bilangan integer terdekat
- `MathFloor` : pembulatan bilangan kebawah menjadi bilangan integer terdekat
- `MathMax` : mencari nilai terbesar diantara dua nilai yang dibandingkan
- `MathMin` : mencari nilai terkecil diantara dua nilai yang dibandingkan
- `MathMod` : bilangan bulat sisa hasil pembagian 2 bilangan
- `MathRound` : pembulatan bilangan ke bilangan integer terdekat
- `MathIsValidNumber` : melakukan pengecekan apakah angka tersebut adalah sebuah bilangan.

Date and Time

Berfungsi untuk mengatur atau mengelola data tanggal dan waktu

- **TimeCurrent** : mengembalikan waktu terakhir diketahui dalam format tanggal dan waktu.
- **TimeLocal** : mengembalikan waktu pada komputer dalam format tanggal dan waktu.
- **Day** : mengembalikan tanggal hari sesuai dengan bulan terakhir diketahui.
- **DayOfWeek** : mengembalikan tanggal hari berbasis nol pada minggu terakhir diketahui.
- **DayOfYear** : mengembalikan tanggal hari pada tahun terakhir diketahui.
- **Hour** : mengembalikan waktu (jam) menyesuaikan zona saat program dibuka.
- **Minute** : mengembalikan waktu (menit) menyesuaikan zona saat program dibuka.
- **Month** : mengembalikan bulan sebagai angka sesuai dengan yang terakhir diketahui.
- **Seconds** : mengembalikan jumlah detik yang berlalu menjadi menit sejak program dibuka.
- **TimeDay** : mengembalikan tanggal yang spesifik sesuai dengan tahun.
- **TimeHour** : mengembalikan waktu (jam) yang spesifik.
- **TimeMinute** : mengembalikan waktu (menit) yang spesifik.
- **TimeMonth** : mengembalikan bulan yang spesifik.
- **TimeSeconds** : mengembalikan jumlah detik yang berlalu sejak tanggal pertama.
- **TimeYear** : mengembalikan tahun yang spesifik.
- **Year** : mengembalikan tahun sesuai dengan terakhir yang diketahui.

String Function

Berfungsi mengerjakan data berupa tipe garis

- `StringAdd` : menambahkan garis di ujung garis lainnya.
- `StringCompare` : membandingkan 2 garis dan mengembalikan 1 garis apabila satu di antaranya lebih baik.
- `StringConcatenate` : membentuk serangkaian parameter yang dilewati.
- `StringFill` : mengisi garis yang ditentukan dengan simbol.
- `StringFind` : mencari sub-garis di dalam garis.
- `StringLen` : mengembalikan angka simbol dalam garis.
- `StringReplace` : mengganti semua sub-garis yang ditemukan dalam garis dengan simbol.
- `StringSubtr` : mengekstrak sebuah sub-garis dari sebuah teks pada posisi spesifik.
- `StringToLower` : mengubah semua simbol yang dipilih menjadi kecil.
- `StringToUpper` : mengubah semua simbol yang dipilih menjadi kapital.
- `StringToLeft` : memotong garis karakter, spasi, dan tab di bagian kiri garis.
- `StringToRight` : memotong garis karakter, spasi, dan tab di bagian kanan garis.

Getting Market Information

Berfungsi mengumpulkan informasi mengenai keadaan pasar.

- `MarketInfo` : mengembalikan berbagai data sekuritas yang terdaftar di 'Market Watch'
- `SymbolTotal` : mengembalikan jumlah simbol yang tersedia (dipilih di Market Watch atau semua).
- `SymbolName` : mengembalikan nama simbol yang ditentukan.

- **SymbolSelect** : memilih simbol di jendela Market Watch atau menghilangkan simbol dari jendela.

Trade Function

Berfungsi untuk mengelola aktivitas perdagangan.

- **OrderClose** : menutup pesanan terbuka.
- **OrderCloseby** : untuk menutup pesanan yang dibuka dengan pesanan baru yang berlawanan.
- **OrderClosePrice** : mengembalikan harga penutupan dari pesanan yang dipilih.
- **OrderCloseTime** : mengembalikan waktu penutupan dari pesanan yang dipilih.
- **OrderComment** : mengembalikan komentar dari pesanan yang dipilih.
- **OrderCommission** : mengembalikan komisi yang terhitung dari pesanan yang dipilih.
- **OrderDelete** : menghapus antrian pesanan yang dibuka sebelumnya.
- **OrderExpiration** : mengembalikan tanggal kadaluarsa pesanan yang tertunda.
- **OrderLots** : mengembalikan jumlah lot pada pesanan.
- **OrderMagicNumber** : mengembalikan nomor identifikasi pada pesanan.
- **OrderModify** : memodifikasi karakteristik pesanan yang sebelumnya dibuka atau tertunda.
- **OrderOpenPrice** : mengembalikan harga terbuka dari pesanan.
- **OrderOpenTime** : mengembalikan waktu terbuka dari pesanan.
- **OrderPrint** : mencetak informasi tentang urutan pesanan.
- **OrderProfit** : mengembalikan keuntungan dari pesanan.
- **OrderSelect** : berfungsi untuk memilih pesanan untuk diproses lebih lanjut.

- **OrderSend** : berfungsi untuk memesan pesanan atau antrian pesanan.
- **OrdersHistoryTotal** : mengembalikan jumlah pesanan tertutup dalam riwayat akun yang dimuat ke terminal.
- **OrderStopLoss** : mengembalikan kerugian dari pesanan.
- **OrdersTotal** : mengembalikan jumlah pasar dan pesanan yang tertunda.
- **OrdersSwap** : mengembalikan nilai tukar pada pesanan.
- **OrderSymbol** : mengembalikan nama simbol pada pada pesanan.
- **OrderTakeProfit** : mengembalikan pengambilan keuntungan pada pesanan.
- **OrderTicket** : mengembalikan nomor tiket pada pesanan.
- **OrderType** : mengembalikan jenis operasi pesanan pada pesanan.

State Checking

Berfungsi untuk mengembalikan parameter pada akun saat ini.

- **GetLastError** : mengembalikan eror terakhir.
- **IsStopped** : mengembalikan nilai asli, apabila program mql4 telah diperintahkan, maka program akan berhenti.
- **Symbol** : mengembalikan nama simbol pada grafik.
- **Period** : mengembalikan zona waktu pada grafik.
- **IsConnected** : memastikan koneksi antara terminal klien dan server.
- **IsDemo** : memastikan Expert Advisor berfungsi dengan baik.
- **IsDllsAllowed** : memastikan fungsi DLL dapat berfungsi di Expert Advisor.

- `IsExpertEnabled` : memastikan Expert Advisor diaktifkan untuk berjalan.

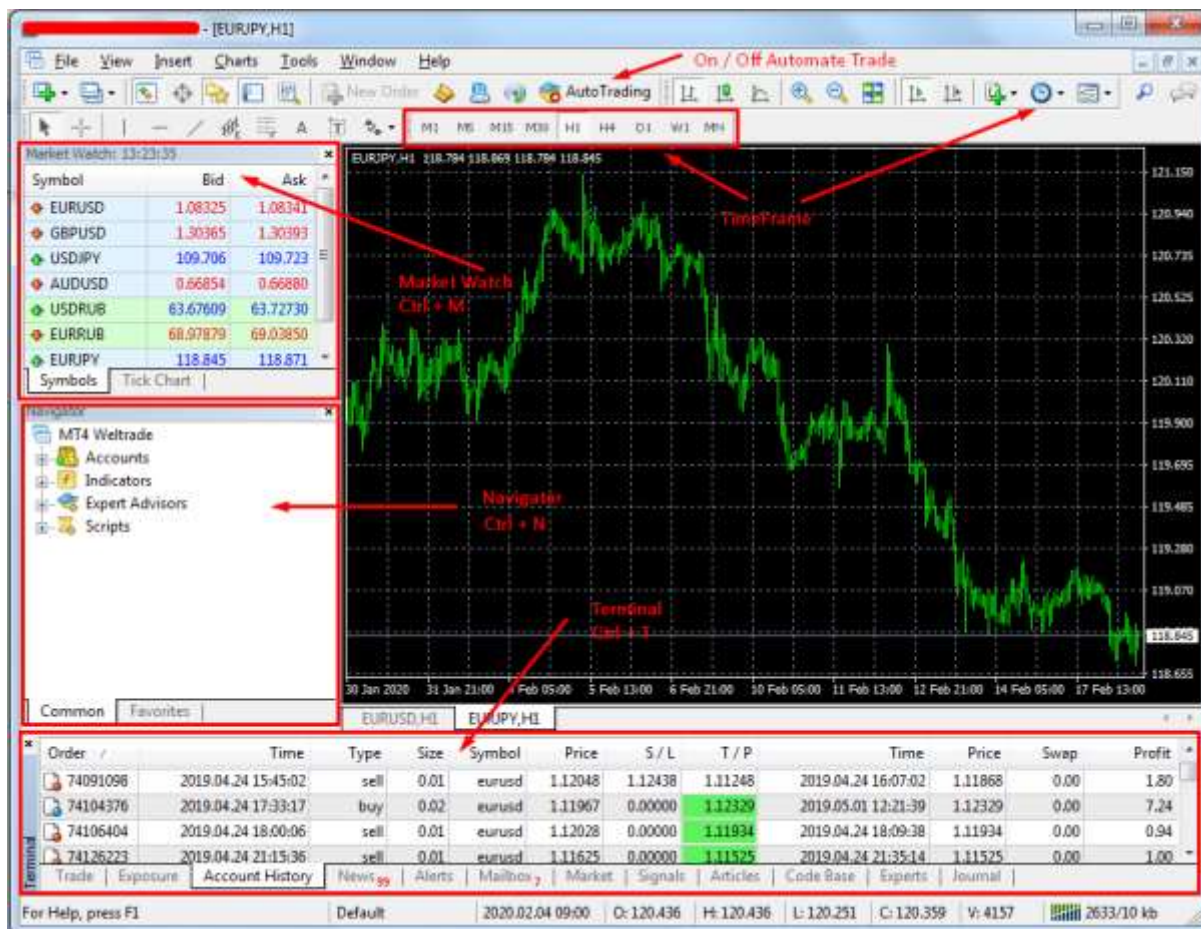
MetaTrader 4

MetaTrader 4 adalah platform untuk perdagangan Forex, menganalisis pasar keuangan dan menggunakan Expert Advisors. Perdagangan seluler, Sinyal Perdagangan, dan Pasar adalah bagian integral dari MetaTrader 4 yang meningkatkan pengalaman perdagangan Forex Anda.

Jutaan trader dengan beragam kebutuhan memilih MetaTrader 4 untuk trading di pasar forex. Platform ini menawarkan banyak peluang bagi para trader dari semua tingkat keahlian: analisis teknis lanjutan, sistem perdagangan fleksibel, perdagangan algoritmik dan Expert Advisors, serta aplikasi perdagangan seluler.

Aplikasi MetaTrader 4 ini bisa didownload melalui website resmi MetaTrader yaitu : <https://metatrader4.com> atau umumnya juga telah disediakan oleh masing-masing broker forex.

Layout MetaTrader 4



Sebelum kita memulai menggunakan MetaTrader 4, ada baiknya kita mengenal fungsi-fungsi yang terdapat pada Platform MetaTrader 4 ini.

- **MarketWatch**

Market Watch: 13:44:26

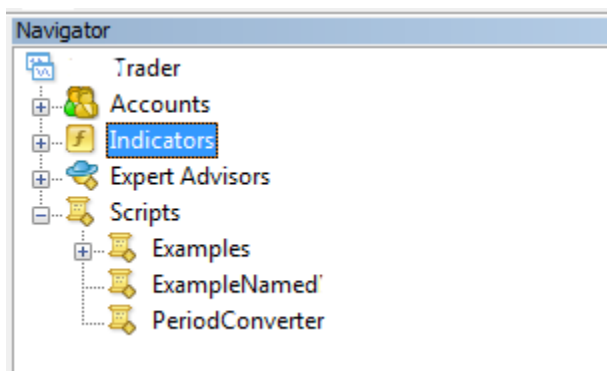
Symbol	Bid	Ask	!
↑ AUDCAD	0.88613	0.88634	21
↓ AUDCHF	0.65557	0.65578	21
↑ AUDJPY	73.345	73.363	18
↑ AUDNZD	1.04487	1.04521	34
↓ CADCHF	0.73976	0.73998	22
↑ CADJPY	82.761	82.785	24
↑ CHFJPY	111.868	111.894	26
↓ EURAUD	1.61935	1.61963	28
↑ EURCAD	1.43546	1.43574	28

Menampilkan semua symbol atau simbol pasangan mata uang yang tersedia oleh broker yang Anda gunakan. Pada bagian window ini terdapat informasi harga Bid, Ask, Spread

Untuk menampilkan window ini, Anda bisa mengklik icon atau dengan tekan tombol **Ctrl + M**



- **Navigator**



Menampilkan informasi Akun Trading Anda, daftar Expert Advisor, Indicator, dan Scripts yang bisa digunakan.

Untuk menampilkan window navigator ini bisa dengan mengklik icon atau dengan tekan tombol **Ctrl + N**



- **Terminal**

Order /	Time	Type	Size	Symbol	Price	S / L	T / P	Time	Price	Swap
1425...	2020.02.11 16:15:01	sell	0.01	eurusd	1.09066	0.00000	1.08966	2020.02.12 17:03:19	1.08966	0.00
1425...	2020.02.12 00:30:06	sell	0.03	eurusd	1.09139	1.08673	1.08339	2020.02.13 17:22:01	1.08339	0.00
1426...	2020.02.12 18:01:04	sell	0.01	eurusd	1.08931	0.00000	1.08831	2020.02.12 19:13:01	1.08831	0.00
1426...	2020.02.12 22:30:08	sell	0.01	eurusd	1.08683	0.00000	1.08583	2020.02.13 13:58:02	1.08583	0.00
1426...	2020.02.13 18:00:03	sell	0.01	eurusd	1.08452	0.00000	1.08352	2020.02.14 02:06:23	1.08352	0.00
1426...	2020.02.14 20:00:07	sell	0.01	eurusd	1.08408	0.00000	1.08308	2020.02.14 23:44:52	1.08308	0.00
1426...	2020.02.17 01:30:12	sell	0.01	eurusd	1.08371	0.00000	1.08271	2020.02.18 02:20:27	1.08271	0.00

Profit/Loss: 500.07 Credit: 0.00 Deposit: 1 000.00 Withdrawal: 0.00

Trade | Exposure | Account History | News 35 | Alerts | Mailbox 8 | Market | Signals | Articles | Code Base | Experts

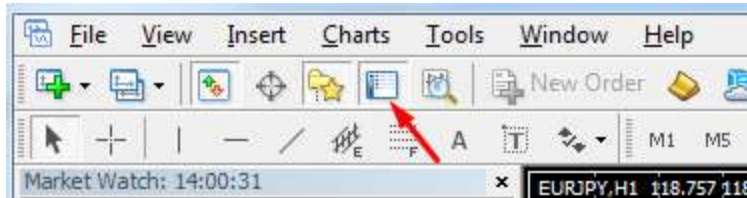
Pada window terminat ini terdapat beberapa tab seperti tab Trade, Exposure, Account History, News, Alerts, Mailbox, Market, Signals, Articles, Code Base, Experts dan Journal.

Umumnya kita akan menggunakan tab Trade untuk melihat informasi posisi trading yang berjalan, Account History untuk menampilkan informasi posisi trading yang telah ditutup,

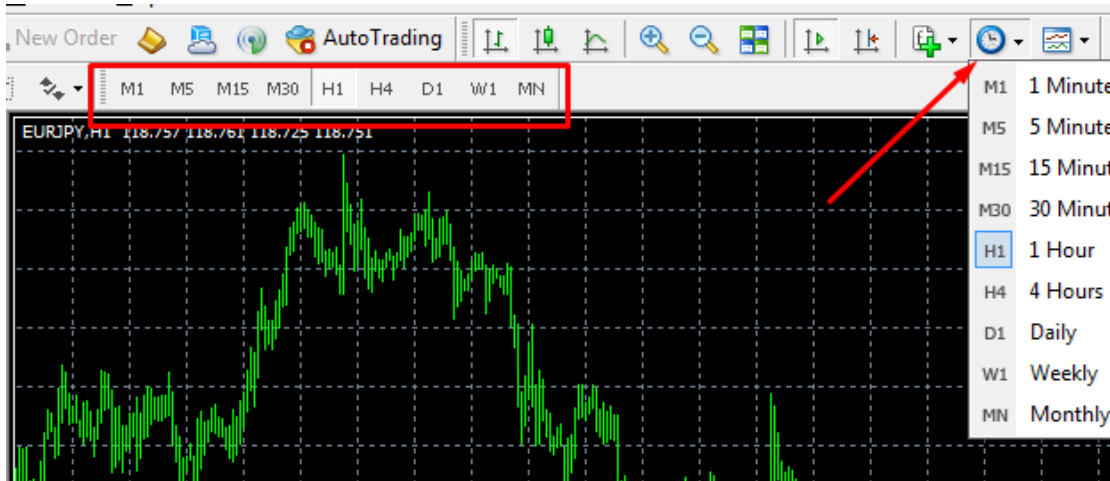
Experts berisi informasi log aktifitas pada Expert Advisor, Indicator atau Script

dan Journal berisi informasi log aktifitas akun trading ini.

Untuk menampilkan window Terminal ini, Ada bisa klik icon atau dengan tekan tombol **Ctrl+T**

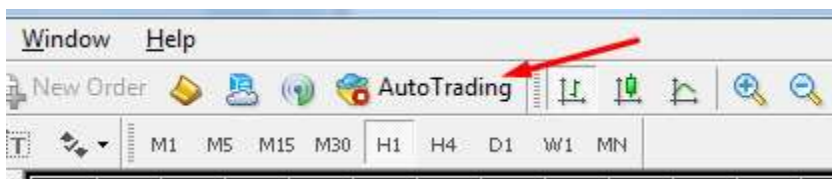


- **TimeFrame**



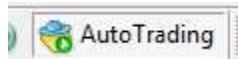
Time Frame dalam forex adalah kurun waktu tertentu yang ditentukan sebagai masa pengamatan pergerakan harga. Pada kurun waktu yang berbeda, kondisi harga yang ditampilkan bisa diterjemahkan secara berbeda. Grafik forex terbentuk dari data pergerakan harga yang dikumpulkan dalam kurun waktu (Time Frame) tertentu.

- **AutoTrading**



Agar Expert Advisors bisa melakukan trading, maka tombol AutoTrading ini harus diaktifkan.

Saat tombol AutoTrading diaktifkan, tombol ini akan berubah menjadi



Memulai Penggunaan MetaEditor

Di sini Anda akan mempelajari dasar-dasar penggunaan terminal MT4. Anda akan menemukan semua langkah di bawah dalam urutan yang tepat. Kami hanya memasukkan hal-hal yang paling penting bagi Anda. Ada banyak detail kecil yang dihilangkan untuk menghemat ruang waktu. Semuanya di sini mencakup apa yang Anda harus tahu untuk menggunakan MT4.

Izinkan kami menyarankan, hanya membaca perlahan dan menikmati proses pembelajaran, sehingga semua informasi meresap dan selalu diingat di masa depan.

Membuka Aplikasi MetaEditor

Untuk membuka MetaEditor, Anda bisa klik “Start Menu”, kemudian “All Programs”. Kemudian cari folder MetaTrader atau nama broker jika Anda download installer MetaTrader 4 dari sebuah broker.



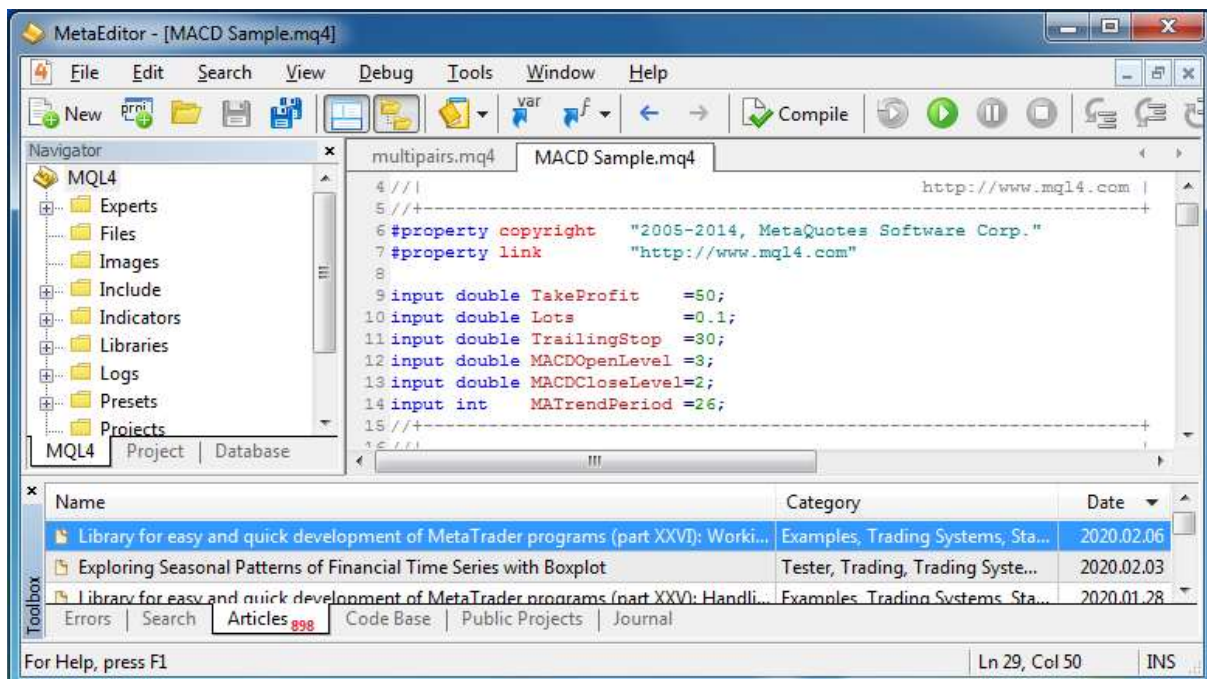
Kemudian klik icon MetaEditor.

Kita juga dapat membuka MetaEditor ini melalui Aplikasi MetaTrader 4. Anda bisa membuka Aplikasi MetaTrader 4 terlebih dahulu melalui “**Start Menu**”, kemudian klik “**All Programs**” dan pilih icon MetaTrader, biasanya jika kita mengdownload (unduh) dari website broker, maka nama file MetaTrader akan menggunakan label yang sama seperti contoh berikut :

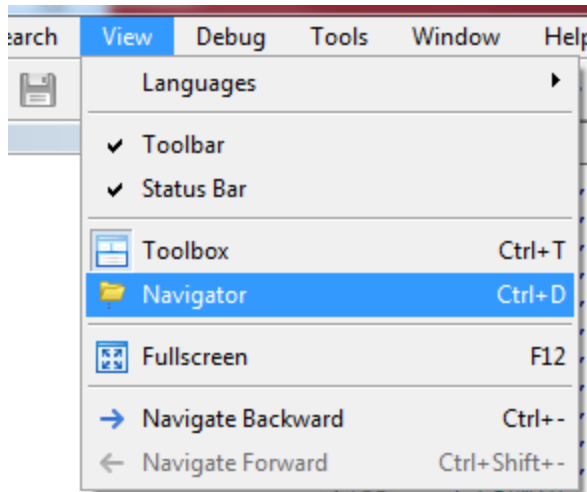


Jika aplikasi MetaTrader sudah terbuka, silakan klik kanan pada file yang akan kita buka pada bagian direktori Navigator, kemudian pilih **“Modify”** atau bisa juga pilih **“Modify Expert”** dari **“Strategy Tester”** atau melalui menu **“Tools”** kemudian **“MetaQuotes Language Editor”**. Kemudian MetaEditor akan terbuka.

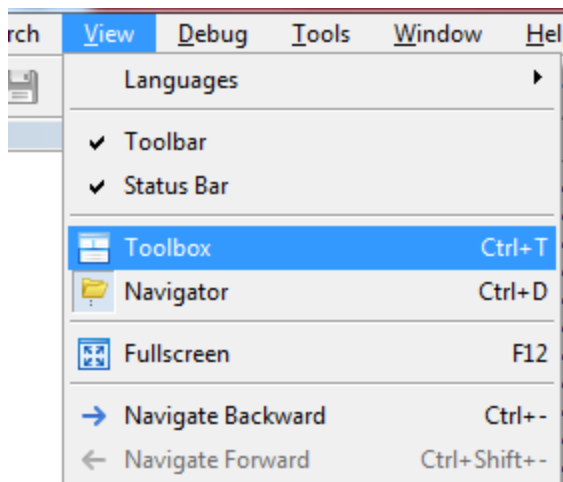
Tampilkan MetaEditor



Pada tampilan MetaEditor ini, kita melihat ada window Navigator dan Toolbox, bila tidak muncul Anda bisa aktifkan melalui menu **“View”** kemudian klik **“Navigator”** untuk menampilkan window Navigator. Atau bisa menggunakan shortcuts **Ctrl + D**.



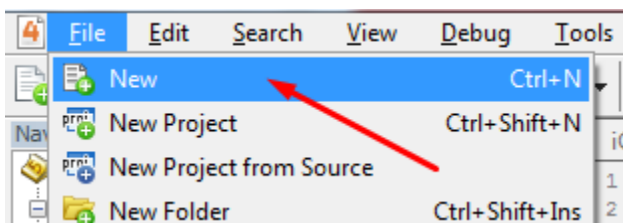
Begitu juga dengan window Toolbox bisa melalui menu **“View”** kemudian klik **“Toolbox”** atau dengan shortcuts **Ctrl + T**.



Memahami Framework dari Expert Advisor di MQL4

Untuk lebih mudah memahami pengguna MetaEditor untuk pemrograman Expert Advisors, kita akan mulai dengan memahami lebih baik apa saja yang ada dan fungsi dari setiap bagian dari framework Expert Advisor di MQL4.

Buka file baru dengan klik menu **"File"** kemudian **"New"** atau dengan tombol **Ctrl + N**



Pilih Expert Advisor (template), kemudian berikan name untuk expert Anda dan lanjutkan hingga akhir wizard.

Dan tampilan awalnya adalah seperti berikut :

```

1 //+-----+
2 //|                                     Belajar_01.mq4 |
3 //|                                     Copyright 2020, VisionEA |
4 //|                                     https://VisionEA.net |
5 //+-----+
6 #property copyright "Copyright 2020, VisionEA"
7 #property link      "https://VisionEA.net"
8 #property version   "1.00"
9 #property strict
10 //+-----+
11 //| Expert initialization function |
12 //+-----+
13 int OnInit()
14 {
15 //---
16
17 //---
18     return(INIT_SUCCEEDED);
19 }
20 //+-----+
21 //| Expert deinitialization function |
22 //+-----+
23 void OnDeinit(const int reason)
24 {
25 //---
26
27 }
28 //+-----+
29 //| Expert tick function |
30 //+-----+
31 void OnTick()
32 {
33 //---
34
35 }
36 //+-----+
37

```

Kita akan bahas setiap bagian dari framework MQL4 pada expert advisor, dimana secara default MQL memiliki 3 fungsi bawaan untuk mengontrol eksekusi program: **OnInit()**, **OnDeinit ()** dan **OnTick ()**. Fungsi **OnInit()** terdiri dari kode yang dijalankan sekali, ketika EA pertama kali dimulai. Fungsi **OnInit()** bersifat optional bila tidak dibutuhkan, bisa dihapus atau dihilangkan.

Fungsi **OnDeinit()** terdiri dari kode yang dijalankan sekali, ketika EA dihentikan. Fungsi ini juga opsional, dan sepertinya Anda tidak perlu menggunakannya dalam EA.

Fungsi **OnTick()** berisi kode program utama, dan diperlukan di EA Anda. Setiap kali fungsi mulai dijalankan, kondisi perdagangan Anda diperiksa, dan pesanan ditempatkan atau ditutup tergantung pada bagaimana kondisi tersebut dievaluasi.

Fungsi OnTick() dijalankan pada setiap tick. Tick adalah pergerakan harga, atau perubahan harga Penawaran (BID) atau Permintaan (ASK) untuk pasangan mata uang. Selama pasar aktif, mungkin ada beberapa tick per detik. Selama pasar yang lambat, mungkin sampai menit pun belum ada tick. Dan kode program atau fungsi dituliskan dalam bagian fungsi OnTick() akan dijalankan setiap adanya tick terjadi.

Selain fungsi default atau bawaan dari MQL, kita juga bisa membuat fungsi sendiri sesuai dengan kebutuhan dan kegunaan. Fungsi-fungsi tersebut sebaiknya diletakan setelah ketiga fungsi bawaan MQL.

Berikut kita akan coba membuat EA sederhana untuk memudahkan pemahaman fungsi-fungsi dan cara penggunaannya.

```
//+-----  
-----+  
  
//|                                     Belajar_01.mq4  
|  
  
//|                                     Copyright 2020,  
VisionEA |  
  
//|                                     https://www.visionea.net  
|  
  
//+-----  
-----+  
  
#property copyright    "2020, VisionEA"  
  
#property link         "https://VisionEA.net/"  
  
  
#property strict
```

```

//--- input parameters

input    double    takeProfit  = 20.0;
input    double    stopLoss    = 20.0;
extern   double    lotSize     = 0.01;
input    int       period_ma_fast = 8;
input    int       period_ma_slow = 20;
input    double    minEquity    = 100.0;


//+-----+
-----+

//| Expert initialization function                                     |
//+-----+
-----+

int OnInit()

{
    //validasi input, sebaiknya kita selalu melakukan validasi
    pada inialisasi data input

    if (takeProfit < 10.0 || stopLoss < 10.0 || lotSize < 0.01
    || period_ma_fast < 8 || period_ma_slow < 20){

        Alert("WARNING - Input data initaliasi tidak valid");

        return (INIT_FAILED);

    }

    return(INIT_SUCCEEDED);

}

```

```

//+-----
-----+

//| Expert deinitialization function |

//+-----
-----+

void OnDeinit(const int reason)

{
    Print ("EA telah diberhentikan");
}


//+-----
-----+

//| Expert tick function |

//+-----
-----+

void OnTick()

{
    if (cekMinEquity()){
        //Melanjutkan proses trading

    }else{
        //Stop trading, karena equity tidak cukup


        Alert ("EA akan segera diberhentikan karena equity tidak
mencukup");
    }
}

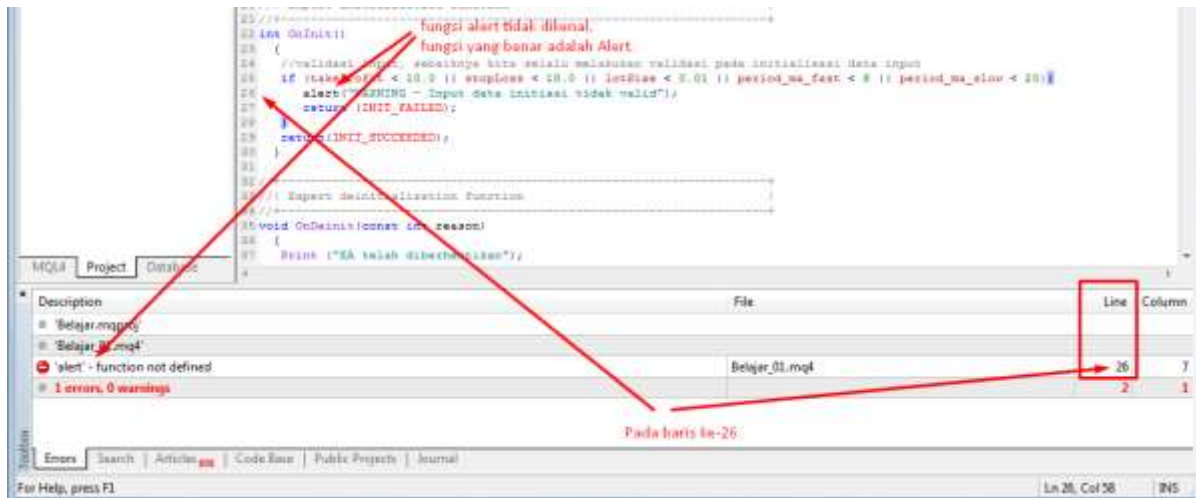
```

```
        ExpertRemove();  
    }  
}
```

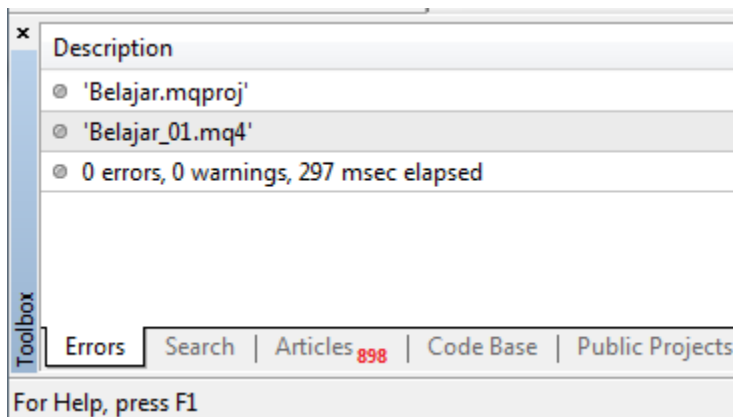
//fungsi tambahan untuk cek equity minimum

```
bool cekMinEquity(){  
    bool valid = false;  
    double equity = 0.0;  
    equity = AccountEquity();  
  
    if (equity > minEquity){  
        valid = true;  
    }  
    return (valid);  
}
```

Jika kita sudah selesai menulis kode program, bisa segera mengompilasi file yang sekaligus akan menyimpan file ini secara otomatis. Klik tombol  untuk mengompilasi file. Bila terdapat kesalahan penulisan kode, maka kompilasi file akan memberikan informasi kesalahan.



Bila berhasil tanpa kesalahan, akan diinformasikan tanpa ada nya error (0 errors)



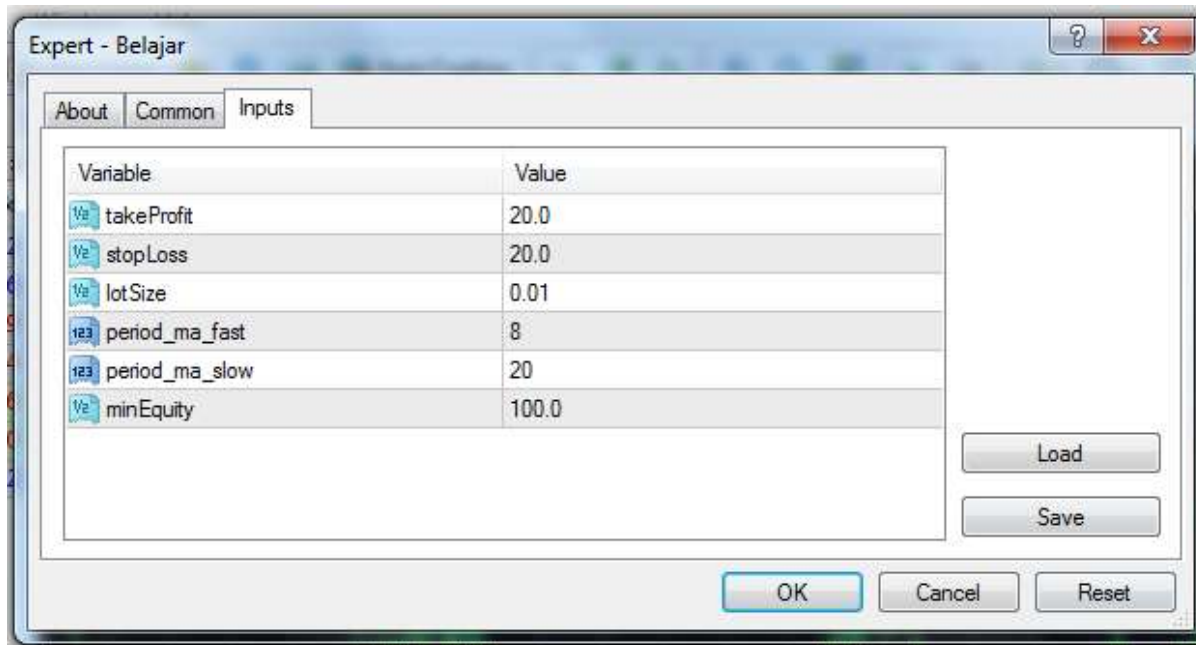
Start Debugging Expert Advisor

Bila program EA telah dikompilasi tanpa kesalahan, maka EA sudah siap untuk menjalankan proses debugging untuk mengetahui proses program berjalan sesuai dengan keinginan atau kebutuhan dan tujuan program ini dibuat. Tekan tombol **F5** atau klik icon untuk memulai proses debugging EA.



Input parameters

Adalah input variable yang didapat dari external



Pada window inputs akan menampilkan nilai-nilai variable external yang telah ditulis dalam kode program seperti kode berikut ini :

```

11 //--- input parameters
12 input    double    takeProfit  = 20.0;
13 input    double    stopLoss    = 20.0;
14 extern   double    lotSize     = 0.01;
15 input    int       period_ma_fast = 8;
16 input    int       period_ma_slow = 20;
17 input    double    minEquity    = 100.0;
18

```

Pengguna program EA ini dapat merubah nilai-nilai variable external ini melalui window inputs saat program EA ini dijalankan.

Stop Debugging Expert Advisors



Klik tombol Stop Debugging untuk stop proses debugging EA.

Program EA akan mulai dijalankan dari fungsi bawaan OnInit() kemudian akan dilanjutkan ke OnTick(), fungsi OnTick() akan terus berulang dijalankan setiap ada perubahan harga penawaran dan permintaan. Saat EA diberhentikan, maka akan dilanjutkan dengan menjalankan fungsi OnDeinit(). Fungsi OnInit() dan OnDeinit() bersifat optional bila ada dituliskan dalam kode program.

Pada contoh program diatas, fungsi **OnInit()** melakukan validasi terhadap nilai-nilai variabel external.

```

22 int OnInit()
23 {
24     //validasi input, sebaiknya kita selalu melakukan validasi pada inialisasi data input
25     if (takeProfit < 10.0 || stopLoss < 10.0 || lotSize < 0.01 || period_ma_fast < 8 || period_ma_slow < 20){
26         Alert("WARNING - Input data inisiasi tidak valid");
27         return (INIT_FAILED);
28     }
29     return (INIT_SUCCEEDED);
30 }

```

Jika nilai variabel external tidak sesuai dengan ketentuan, maka akan menampilkan window alert dan kemudian program EA ini akan diberhentikan, sehingga akan dilanjutkan dengan memanggil fungsi **OnDeinit()**;

```
35 void OnDeinit(const int reason)
36 {
37     Print ("EA telah diberhentikan");
38 }
39
```

Bila validasi nilai variabel external berhasil, maka akan dilanjutkan ke fungsi **OnTick()**.

Fungsi **Ontick()** ini akan terus diulang setiap kali ada tick pada layar grafik tersebut.

Membuka Posisi Trading

Bid, Ask & Spread

Sebagai seorang trader Forex, Anda mungkin sudah terbiasa dengan harga Penawaran (Bid) dan Permintaan (Ask). Tetapi Anda mungkin tidak menyadari peran mereka dalam penempatan pesanan. Sangat penting menggunakan harga yang benar saat membuka atau menutup pesanan (order).

Harga **Bid** adalah apa yang Anda lihat di grafik MetaTrader. Biasanya itu yang kita pikirkan ketika kita memikirkan "harga saat ini." Harga **Ask** umumnya hanya beberapa pip di atas harga Penawaran. Perbedaan antara Bid dan Ask adalah **spread**, yang merupakan komisi broker untuk menempatkan pesanan.

Harga ask adalah tempat kami membuka pesanan beli (open buy orders), dan menutup pesanan jual (close sell orders). Harga Bid adalah tempat kami membuka pesanan jual (open sell orders), dan menutup pesanan beli (close sell orders). Anda harus menunjukkan harga yang benar saat membuka pesanan pasar, atau ketika menutup pesanan di pasar, jadi ingatlah perbedaan di antara keduanya.

Order Types

Ada dua jenis orders yang dapat ditempatkan di MetaTrader: **Instant Order**, **Pending Order**. Pending Order dibagi menjadi 2 jenis, yaitu **Stop Order** dan **Limit Order**.

Instant Order adalah yang paling umum. Instant order segera membuka posisi pada harga Bid atau Ask yang berlaku.

Saat menempatkan instant order dalam MQL, kita harus menentukan harga pembukaan (umumnya ask atau bid terbaru). Jika harga pembukaan yang ditentukan sudah usang, karena pasar yang bergerak cepat atau penundaan dalam pelaksanaan

program, terminal akan berusaha untuk menempatkan pesanan pada harga pasar saat ini, asalkan itu dalam slippage maksimum.

Jika Anda melakukan instant order menggunakan dialog New Order di MetaTrader, Anda akan melihat pengaturan di bagian bawah berlabel "Enable maximum deviation from quoted price." Ketika ini dicentang, Anda kemudian dapat menentukan deviasi maksimum dalam pips. Ini adalah slippage maksimum.

Jika harga saat ini jatuh di luar harga pembukaan yang ditentukan kami, plus atau minus slippage, kesalahan requote akan terjadi dan pesanan tidak akan dilakukan. Anda mungkin telah memperhatikan ini ketika mencoba melakukan pemesanan pasar selama pasar yang bergerak cepat. Perhatikan bahwa broker ECN / STP tidak menggunakan pengaturan slippage, dan akan selalu membuka pesanan pasar pada harga saat ini.

Stop Order adalah jenis pending order. Pending order adalah permintaan untuk membuka order pasar dengan harga tertentu. Buy stop order ditempatkan di atas harga saat ini, sementara order stop jual ditempatkan di bawah harga saat ini. Harapannya adalah bahwa harga pada akhirnya akan naik atau turun ke level itu dan berlanjut ke arah itu, menghasilkan keuntungan.

Limit Order adalah kebalikan dari stop order. Pesanan batas pembelian ditempatkan di bawah harga saat ini, sementara pesanan batas jual ditempatkan di atas harga saat ini. Harapannya adalah bahwa harga akan naik atau turun ke level itu, memicu pesanan, dan kemudian berbalik. Limit order tidak terlalu sering digunakan dalam perdagangan otomatis.

Waktu kedaluwarsa (expiration) dapat ditetapkan untuk pending order. Jika sebuah order ditentukan waktu kadaluwarsa, maka order tersebut akan otomatis dihapus bila sudah kadaluwarsa. Tidak semua broker mendukung berakhirnya perdagangan.

Membuka Pesanan (order)

Proses membuka pesanan dalam MQL melibatkan beberapa langkah. Kita harus menentukan yang berikut sebelum menempatkan pesanan:

- **Order Type** : Jenis pesanan yang akan ditempatkan - BUY atau SELL, BUY STOP atau SELL STOP, BUY LIMIT atau SELL LIMIT.
- **Pair to trade** : Pasangan mata uang yang akan diperdagangkan - umumnya bagan (CHART) yang dilampirkan EA.
- **Lot Size** : Ini bisa berupa ukuran lot tetap, atau yang dihitung menggunakan manajemen keuangan.
- **Open Price** : Harga pembukaan pesanan. Untuk instant order, ini akan menjadi harga Ask atau bid saat ini. Untuk pending order, harga pembukaan harus jarak minimum dari harga saat ini, dan harus di atas atau di bawah harga saat ini seperti yang dipersyaratkan oleh jenis pesanan.
- **Slippage** : adalah eksekusi order dengan harga yang lebih buruk dari harga order yang ditempatkan sebelumnya di pasar. Memperbaiki harga yang meningkat. Slippage sering terjadi kompilasi berita penting yang diperbaiki kuat memberikan dampak ke pasar dirilis. Selip besaran Paling sederhana dari satu hingga akhirnya poin.
- **Stoploss Price** : Stop loss dapat berupa harga yang telah ditentukan, nilai indikator, jumlah pip tetap dari harga pembukaan pesanan, atau dapat dihitung secara dinamis menggunakan rutin manajemen risiko. Stop loss dapat ditempatkan dengan pesanan, atau dapat ditambahkan ke pesanan sesudahnya.
- **Takeprofit Price** : Ini umumnya jumlah tetap dari harga pembukaan pesanan, meskipun dapat dihitung dengan menggunakan metode lain juga. Take profit dapat ditempatkan dengan pesanan, atau dapat ditambahkan ke pesanan sesudahnya.

- **Order identifiers** : Pengidentifikasi pesanan seperti comment, atau "magic number" yang mengidentifikasi pesanan sebagai ditempatkan oleh penasihat ahli tertentu.
- **Expiration** : Harga kedaluwarsa opsional untuk pending order, jika broker mendukungnya.

OrderSend()

Fungsi **OrderSend ()** digunakan untuk melakukan pemesanan dalam MQL. Sintaksnya adalah sebagai berikut:

```
int OrderSend(
    string symbol,           // symbol atau nama pair
    int cmd,                 // order type
    double volume,          // volume, lot size
    double price,            // Open price
    int slippage,            // slippage
    double stoploss,         // stop loss
    double takeprofit,       // take profit
    string comment=NULL,     // comment
    int magic=0,             // magic number
    datetime expiration=0,    // pending order expiration
    color arrow_color=clrNONE // color
);
```

Contoh:

```
double lotsize = 0.01;

int slippage = 3;

double stoploss = 0.0;
```

```
double takeprofit = 0.0;

string comment = "Order Buy";

int magicnumber = 0;
```

```
OrderSend(Symbol(), OP_BUY, lotsize, Ask, slippage, stoploss, takeprofit,
comment, magicnumber, 0, clrBlue);
```

Fungsi **Symbol()** akan mengembalikan symbol pada grafik yang sedang digunakan, **OP_BUY** menunjukkan perintah untuk membuka posisi order BUY, **Lotsize** adalah besarnya volume transaksi, pembukaan order buy menggunakan harga permintaan (**Ask**). Stoploss digunakan untuk menentukan dimana batas harga menutup dengan kerugian. Harga stoploss berada dibawah harga pembukaan, takeprofit adalah batas harga menutup order dengan keuntungan dimana harga takeprofit berada diatas harga pembukaan.

Sedangkan untuk membuka posisi order Sell, bisa lihat contoh berikut :

```
OrderSend(Symbol(), OP_SELL, lotsize, Bid, slippage, stoploss, takeprofit,
comment, magicnumber, 0, clrRed);
```

Order yang digunakan adalah **OP_SELL** dan harga pembukaan menggunakan harga penawaran (**Bid**).

```
OrderSend(Symbol(), OP_BUYSTOP, lotsize, HargaPending_BuyStop,
slippage, stoploss, takeprofit, comment, magicnumber, 0, clrRed);
```

HargaPending_BuyStop harus lebih besar dari harga Ask saat ini.

```
OrderSend(Symbol(), OP_SELLSTOP, lotsize, HargaPending_SellStop,
slippage, stoploss, takeprofit, comment, magicnumber, 0, clrRed);
```

HargaPending_SellStop harus dibawah harga Bid saat ini.

```
OrderSend(Symbol(), OP_BUYLIMIT, lotsize, HargaPending_BuyLimit,
slippage, stoploss, takeprofit, comment, magicnumber, 0, clrRed);
```


HargaPending_BuyLimit harus dibawah harga Bid saat ini.

```
OrderSend(Symbol(), OP_SELLLIMIT, lotsize, HargaPending_SellLimit,  
slippage, stoploss, takeprofit, comment, magicnumber, 0, clrRed);
```

HargaPending_SellLimit harus diatas harga Ask saat ini.

Menghitung Harga StopLoss dan TakeProfit

Ada beberapa cara menghitung harga StopLoss dan TakeProfit. Metode yang paling umum adalah menentukan jumlah pips dari harga pembukaan pesanan. Misalnya, jika kita memiliki pengaturan stoploss 20 pips dan takeprofit 30 pips, itu berarti bahwa harga stoploss akan bergeser 20 pips dari harga pembukaan dan begitu juga dengan harga takeprofit akan bergeser 30 pips dari harga pembukaan dengan arah yang berlawanan dengan arah stoploss.

Contoh pembukaan order OP_BUY di harga 1.3123, maka harga stoploss adalah harga pembukaan dikurangi 20 pips sehingga harga stoploss adalah $1.3123 - 0.0020 = 1.3103$ sedangkan untuk harga takeprofit akan ditambahkan 30 pips dari harga pembukaan menjadi 1.3153.

Untuk pembukaan order OP_SELL, maka harga stoploss dihitung dari harga pembukaan ditambahkan 20 pips dan harga takeprofit dihitung dari harga pembukaan dikurangi 30 pips.

Kita juga dapat menggunakan nilai indikator, parameter eksternal atau beberapa jenis perhitungan harga lainnya untuk menentukan harga stoploss dan takeprofit. Yang perlu kita lakukan adalah memverifikasi bahwa harga stop loss atau take profit itu valid.

Point

Point adalah variabel yang telah ditentukan sebelumnya dalam MQL yang mengembalikan unit harga terkecil dari suatu mata uang, tergantung pada jumlah tempat desimal. Untuk pasangan mata uang 4 desimal, intinya adalah 0,0001. Untuk pasangan Yen, ini 0,01.

Mari kita hitung stoploss untuk order **OP_BUY**. Kita akan menetapkan harga Permintaan (**Ask**) saat ini untuk harga pembukaan (**OpenPrice**). Kita akan memeriksa

untuk melihat apakah pengaturan StopLoss kami lebih besar dari nol. Jika demikian, kami akan mengalikan StopLoss dengan Point. Kemudian kita akan mengurangi itu dari OpenPrice. Hasilnya akan disimpan dalam variabel BuyStopLoss.

```
double OpenPrice = Ask;  
  
if (StopLoss > 0) double BuyStopLoss = OpenPrice - (StopLoss * Point);  
  
// 1.4600 - (50 * 0.0001) = 1.4550
```

Jika StopLoss tidak lebih besar dari nol, maka BuyStopLoss diinisialisasi dengan nilai 0, dan tidak ada stoploss yang akan ditempatkan dengan pesanan. Dengan asumsi Point sama dengan 0,0001, jika harga pembukaan pesanan adalah 1,4600, dan stoploss kita adalah 50 pips, maka harga stoploss untuk order OP_BUY adalah $1,4600 - (0,0050) = 1,4550$.

Saat ini, banyak broker telah bergerak menuju kuotasi harga pip fraksional, dengan 3 tempat desimal untuk pasangan Yen dan 5 tempat desimal untuk semua pasangan lainnya. Jika broker kita menggunakan kutipan pip fraksional, maka dalam contoh kami di atas, Point akan sama dengan 0,00001. Sehingga saat kita ingin merujuk pada nilai 5 pip maka pada akun dengan 3 tempat desimal atau 5 tempat desimal akan berubah menjadi 50 pips.

Dari pada terus menambahkan angka nol pada setiap penggunaannya, lebih baik kita buat sebuah fungsi untuk mengatur perbedaan ini.

Sebagai contoh akan kita buat sebuah fungsi yang bernama PipPoint()

```
double PipPoint(string pair)  
{  
  
    double CalcPoint = 0.0;  
  
    int CalcDigits = MarketInfo(pair, MODE_DIGITS);  
  
    if(CalcDigits == 2 || CalcDigits == 3) CalcPoint = 0.01;
```

```

else if(CalcDigits == 4 || CalcDigits == 5) CalcPoint = 0.0001;

return(CalcPoint);

}

```

Argumen **string pair** adalah simbol dari pasangan mata uang yang ingin kita ambil poinnya. Fungsi **MarketInfo ()** dengan parameter **MODE_DIGITS** mengembalikan jumlah tempat desimal (digit) untuk pasangan itu. Pernyataan **if-else** memberikan nilai poin yang sesuai ke variabel **CalcPoint**, tergantung pada jumlah digit.

Berikut adalah contoh penggunaan fungsi ini. Anda akan menggunakan grafik aktif yang sering digunakan, jadi kita akan mengirim fungsi **Symbol ()** sebagai argumen. Ini akan mengembalikan poin untuk grafik saat ini.

```
double numPoint = PipPoint(Symbol());
```

Contoh lain untuk dengan secara khusus menyebutkan nama pair :

```
double numPoint = PipPoint( "EURUSD" );           //Hasilnya : 0.0001
```

```
double numPoint = PipPoint( "GBPUSD" );           //Hasilnya : 0.0001
```

```
double numPoint = PipPoint( "USDJPY" );           //Hasilnya : 0.01
```

Karena ada perbedaan pip fraksional, maka kita akan membuat sebuah fungsi untuk mengatur nilai slippage supaya dapat dengan mudah disesuaikan.

```

int GetSlippage(string pair, int SlippagePips)

{

    int CalcSlippage = 0;

    int CalcDigits = (int) MarketInfo(pair,MODE_DIGITS);

    if(CalcDigits == 2 || CalcDigits == 4) CalcSlippage = SlippagePips;

```

```

else if(CalcDigits == 3 || CalcDigits == 5) CalcSlippage = SlippagePips * 10;

return(CalcSlippage);

}

```

Kita mengirim 2 parameter yaitu string pair dan int SlippagePips. Dengan menggunakan fungsi MarketInfo() dengan Symbol dari parameter pair dan MODE_DIGITS untuk mendapatkan jumlah digit symbol tersebut yang disimpan ke variable CalcDigits.

Mengambil informasi Pesanan

Setelah kita berhasil membuat pesanan, kita perlu mengambil beberapa informasi tentang pesanan tersebut jika kami ingin memodifikasi atau menutupnya. Kita melakukan ini menggunakan fungsi **OrderSelect ()**. Untuk menggunakan OrderSelect (), kita dapat **OrderTicket()**, atau kita dapat mengulangi kumpulan data pesanan yang terbuka dan memilih masing-masing secara berurutan.

Setelah kita memilih pesanan menggunakan OrderSelect (), kita dapat menggunakan berbagai fungsi informasi pesanan untuk mengembalikan informasi tentang pesanan tersebut, termasuk harga stoploss, takeprofit, harga pembukaan pesanan, harga penutupan dan banyak lagi.

OrderSelect()

Berikut adalah sintaks untuk fungsi **OrderSelect ()**:

```

bool OrderSelect(

    int    index,        // index or order ticket

    int    select,        // flag : SELECT_BY_POS atau SELECT_BY_TICKET

```

```
int pool=MODE_TRADES // mode : MODE_TRADES atau MODE_HISTORY  
);
```

- **index** - Ini adalah nomor tiket pesanan yang ingin kita pilih, atau posisi di kumpulan pesanan. Parameter **select** akan menunjukkan yang mana yang kami gunakan.
- **select** - Konstanta yang menunjukkan apakah parameter Indeks adalah nomor tiket atau posisi kumpulan pesanan:
 - **SELECT_BY_TICKET** - Nilai parameter **index** adalah nomor tiket pesanan.
 - **SELECT_BY_POS** - Nilai parameter **index** adalah urutan posisi pesanan.
- **Pool** - Konstanta opsional yang menunjukkan kumpulan pesanan: pending / open order, atau order yang sudah ditutup.
 - **MODE_TRADES** - Secara default, akan memeriksa kumpulan order yang masih terbuka atau pending order.
 - **MODE_HISTORY** - Memeriksa kumpulan order tertutup (riwayat pesanan).

Jika fungsi **OrderSelect ()** berhasil menemukan orde, nilai kembali akan **true**, jika tidak, nilai kembali akan **false**.

Berikut adalah contoh fungsi **OrderSelect ()** menggunakan nomor tiket pesanan. Variabel Tiket harus berisi tiket pesanan yang valid:

```
OrderSelect(Ticket, SELECT_BY_TICKET);
```

Setelah fungsi **OrderSelect ()** dipanggil, kita dapat menggunakan salah satu fungsi informasi pesanan untuk mengambil informasi tentang pesanan itu. Daftar lengkap fungsi yang dapat digunakan dengan **OrderSelect ()** dapat ditemukan di MQL Reference under *Trading Functions*. Berikut adalah daftar fungsi informasi pesanan yang paling umum digunakan:

- **OrderSymbol ()** - Simbol instrumen tempat urutan dipilih.
- **OrderType ()** - Jenis pesanan dari pesanan yang dipilih: beli atau jual; instant, limit atau stop.
- **OrderOpenPrice ()** - Harga pembukaan pesanan yang dipilih.
- **OrderLots ()** - Ukuran lot dari pesanan yang dipilih.
- **OrderStopLoss ()** - Harga stop loss dari pesanan yang dipilih.
- **OrderTakeProfit ()** - Harga ambil untung dari pesanan yang dipilih.
- **OrderTicket ()** - Nomor tiket pesanan yang dipilih. Umumnya digunakan saat bersepeda melalui kumpulan pesanan dengan parameter **SELECT_BY_POS**.
- **OrderMagicNumber ()** - Nomor ajaib dari pesanan yang dipilih. Saat bersepeda melalui pesanan, Anda harus menggunakan ini untuk mengidentifikasi pesanan yang dilakukan oleh EA Anda.
- **OrderComment ()** - Komentar yang ditempatkan dengan pesanan. Ini dapat digunakan sebagai pengidentifikasi pesanan sekunder.
- **OrderClosePrice ()** - Harga penutupan pesanan yang dipilih. Pesanan harus sudah ditutup (mis. Hadir dalam kumpulan riwayat pesanan).
- **OrderOpenTime ()** - Waktu pembukaan pesanan yang dipilih.
- **OrderCloseTime ()** - Waktu penutupan pesanan yang dipilih.
- **OrderProfit ()** - Mengembalikan profit (dalam mata uang deposit/setoran) untuk pesanan yang dipilih.

kita harus menggunakan **OrderSelect ()** sebelum menutup atau mengubah pesanan. Mari kita ilustrasikan bagaimana kita menggunakan **OrderSelect ()** untuk menutup pesanan.

Menutup Pesanan

Ketika kita menutup pesanan, kita keluar dari/menutup perdagangan pada harga pasar saat ini. Untuk pesanan OP_BUY, kami tutup dengan harga Penawaran (Bid), dan untuk pesanan OP_SELL, kami tutup di Permintaan (Ask). Untuk pesanan yang tertunda, kita cukup menghapus pesanan dari kumpulan order perdagangan.

OrderClose()

Kita menutup pesanan pasar menggunakan fungsi **OrderClose()**. Berikut ini sintaksnya:

```
bool OrderClose(int Ticket, double Lots, double Price, int Slippage, color Arrow);
```

- **Ticket** - Nomor tiket pesanan pasar untuk ditutup.
- **Lots** - Jumlah lot untuk ditutup. Kebanyakan broker mengizinkan penutupan sebagian.
- **Price** - Harga pilihan untuk menutup perdagangan. Untuk pesanan beli, ini akan menjadi harga Penawaran saat ini, dan untuk pesanan jual, harga Permintaan saat ini.
- **Slippage** - Slippage yang diizinkan dari harga penutupan, dalam pips.
- **color** - Konstanta warna untuk panah penutup. Jika tidak ada warna yang ditunjukkan, tidak ada panah yang akan ditarik.

Kita dapat menutup bagian dari perdagangan dengan menentukan ukuran lot parsial. Misalnya, jika Anda memiliki perdagangan terbuka dengan ukuran lot 3, dan Anda ingin menutup 1 lot dari perdagangan, maka tentukan 1 lot untuk argumen Lot. Perhatikan bahwa tidak semua broker mendukung penutupan sebagian.

Dianjurkan bahwa jika Anda perlu menutup posisi di beberapa bagian, Anda harus melakukan beberapa pesanan daripada melakukan penutupan parsial. Dengan

menggunakan contoh di atas, Anda akan menempatkan dua pesanan masing-masing 1lot dan 2 lot, kemudian cukup tutup satu pesanan dengan lot 1 ketika Anda ingin menutup pesanan dengan lot 1.

Contoh berikut ini menutup pesanan beli:

```
OrderSelect(CloseTicket,SELECT_BY_TICKET);
```

```
if(OrderCloseTime() == 0 && OrderType() == OP_BUY)
{
    double CloseLots = OrderLots();

    double ClosePrice = Bid;

    bool Closed=OrderClose(CloseTicket,CloseLots,ClosePrice,UseSlippage,Red);
}
```

OrderDelete()

Ada fungsi terpisah untuk menutup pending order. **OrderDelete ()** memiliki dua argumen, nomor tiket dan warna panah. Tidak diperlukan harga penutupan, ukuran lot, atau selip. Berikut adalah kode untuk menutup order stop buy yang tertunda:

```
OrderSelect(CloseTicket,SELECT_BY_TICKET);

if(OrderCloseTime() == 0 && OrderType() == OP_BUYSTOP)
{
    bool Deleted = OrderDelete(CloseTicket, Red);
}
```

Seperti yang kita lakukan dengan fungsi `OrderClose ()` di atas, kita perlu memeriksa jenis pesanan untuk memastikan itu adalah pending order. Konstanta jenis pesanan tertunda adalah `OP_BUYSTOP`, `OP_SELLSTOP`, `OP_BUYLIMIT` dan `OP_SELLLIMIT`. Untuk menutup jenis pending order lainnya, cukup ubah jenis pesanan.

Jika pesanan telah diisi, maka sekarang adalah pesanan pasar, dan harus ditutup menggunakan **`OrderClose ()`**.

Membuat Aplikasi Perdagangan Otomatis Sederhana

Penjelasan Kebutuhan Aplikasi

Kita akan mencoba praktekkan apa yang telah kita pelajari dari penjelasan diatas. Kita akan memulai membuat aplikasi perdagangan otomatis yang menggunakan indikator Moving Average. Sistem ini akan menggunakan 2 indikator moving average dimana akan membuka posisi BUY bila indikator Moving Average Period 8 (period_ma_fast) memotong keatas indikator Moving Average Period 20 (period_ma_slow) dan sekaligus akan menutup posisi pesanan SELL. Indikator moving average Period 8 memotong kebawah indikator Moving Average Period 20, maka akan membuka posisi pesanan SELL dan sekaligus menutup posisi pesanan BUY.

Tidak tidak terjadi potongan, posisi pesanan akan ditutup berdasarkan harga stoploss atau harga takeprofit yang sudah ditentukan.

Penulisan Kode Aplikasi

```
#property copyright "2020, Yulianto Hiu"
```

```
#property link "https://VisionEA.net/"
```

```
#property strict
```

```
//--- input parameters
```

```
extern double takeProfit = 20.0;
```

```
extern double stopLoss = 20.0;
```

```
extern double lotSize = 0.01;
```

```
extern int    period_ma_fast = 8;

extern int    period_ma_slow = 20;

extern double minEquity  = 100.0;
```

```
extern int Slippage = 3;
```

```
extern int MagicNumber = 889;
```

```
//Global Variables
```

```
double UsePoint  = 0.0;
```

```
int    UseSlippage = 0;
```

```
int    BuyTicket  = 0;
```

```
int    SellTicket = 0;
```

```
//+-----+
```

```
//| Expert initialization function |
```

```
//+-----+
```

```
int OnInit()
```

```
{
```

```
    //validasi input, sebaiknya kita selalu melakukan validasi pada inialisasi data input
```

```
    if (takeProfit < 0.0 || stopLoss < 0.0 || lotSize < 0.01 || period_ma_fast < 0 ||  
period_ma_slow <= period_ma_fast){
```

```
        Alert("WARNING - Input data initaliasi tidak valid");
```

```
    return (INIT_FAILED);  
  
}
```

```
UsePoint = PipPoint(Symbol());
```

```
UseSlippage = GetSlippage(Symbol(),Slippage);
```

```
return(INIT_SUCCEEDED);  
  
}
```

```
//+-----+
```

```
//| Expert deinitialization function          |
```

```
//+-----+
```

```
void OnDeinit(const int reason)
```

```
{
```

```
    Print ("EA telah diberhentikan");
```

```
}
```

```
//+-----+
```

```
//| Expert tick function                      |
```

```
//+-----+
```

```
void OnTick()
```

```
{
```

```

if (cekMinEquity()){

    //Moving Averages

    double FastMA = iMA(NULL, 0, period_ma_fast, 0, MODE_SMA, 0, 1);

    double SlowMA = iMA(NULL, 0, period_ma_slow, 0, MODE_SMA, 0, 1);


    // Buy order

    if(FastMA > SlowMA && BuyTicket == 0)

    {

        if (OrderSelect(SellTicket,SELECT_BY_TICKET)){

            // Close order

            if(OrderCloseTime() == 0 && SellTicket > 0)

            {

                double CloseLots = OrderLots();

                double ClosePrice = Ask;

                bool   Closed   =   OrderClose(SellTicket,CloseLots,ClosePrice,UseSlippage,
clrRed);

                SellTicket = 0;

            }


            double OpenPrice = Ask;

            // Calculate stop loss and take profit

            double BuyStopLoss = 0.0, BuyTakeProfit = 0.0;

```

```

        if(stopLoss > 0) BuyStopLoss = OpenPrice - (stopLoss * UsePoint);

        if(takeProfit > 0) BuyTakeProfit = OpenPrice + (takeProfit * UsePoint);

        // Open buy order

        BuyTicket          =          OrderSend(Symbol(),OP_BUY,
lotSize,OpenPrice,UseSlippage,BuyStopLoss,BuyTakeProfit,          "OP
BUY",MagicNumber,0, clrBlue);

    }

}

// Sell order

if(FastMA < SlowMA && SellTicket == 0)

{

    // Close order

    if(OrderCloseTime() == 0 && BuyTicket > 0)

    {

        if (OrderSelect(BuyTicket,SELECT_BY_TICKET))

        {

            double CloseLots = OrderLots();

            double ClosePrice = Bid;

            bool   Closed   =   OrderClose(BuyTicket,CloseLots,ClosePrice,UseSlippage,
clrBlue);

            BuyTicket = 0;

        }

    }

}

```

```
}
```

```
double OpenPrice = Bid;
```

```
// Calculate stop loss and take profit
```

```
double SellStopLoss = 0.0, SellTakeProfit = 0.0;
```

```
if(stopLoss > 0) SellStopLoss = OpenPrice + (stopLoss * UsePoint);
```

```
if(takeProfit > 0) SellTakeProfit = OpenPrice - (takeProfit * UsePoint);
```

```
// Open buy order
```

```
SellTicket = OrderSend(Symbol(),OP_SELL,  
lotSize,OpenPrice,UseSlippage,SellStopLoss,SellTakeProfit,  
"OP  
SELL",MagicNumber,0, clrRed);
```

```
}
```

```
}else{
```

```
//Stop trading, karena equity tidak cukup
```

```
Alert ("EA akan segera diberhentikan karena equity tidak mencukup");
```

```
ExpertRemove();
```

```
}
```

```
}
```

```
//fungsi tambahan untuk cek equity minimum
```

```
bool cekMinEquity(){
```



```

bool valid = false;

double equity = 0.0;

equity = AccountEquity();


if (equity > minEquity){

    valid = true;

}

return (valid);

}


// Pip Point Function

double PipPoint(string pair)

{

    double CalcPoint = 0.0;

    int CalcDigits = (int) MarketInfo(pair,MODE_DIGITS);

    if(CalcDigits == 2 || CalcDigits == 3) CalcPoint = 0.01;

    else if(CalcDigits == 4 || CalcDigits == 5) CalcPoint = 0.0001;

    return(CalcPoint);

}


// Get Slippage Function

int GetSlippage(string pair, int SlippagePips)

{

```

```
int CalcSlippage = 0;

int CalcDigits = (int) MarketInfo(pair,MODE_DIGITS);

if(CalcDigits == 2 || CalcDigits == 4) CalcSlippage = SlippagePips;

else if(CalcDigits == 3 || CalcDigits == 5) CalcSlippage = SlippagePips * 10;

return(CalcSlippage);

}
```

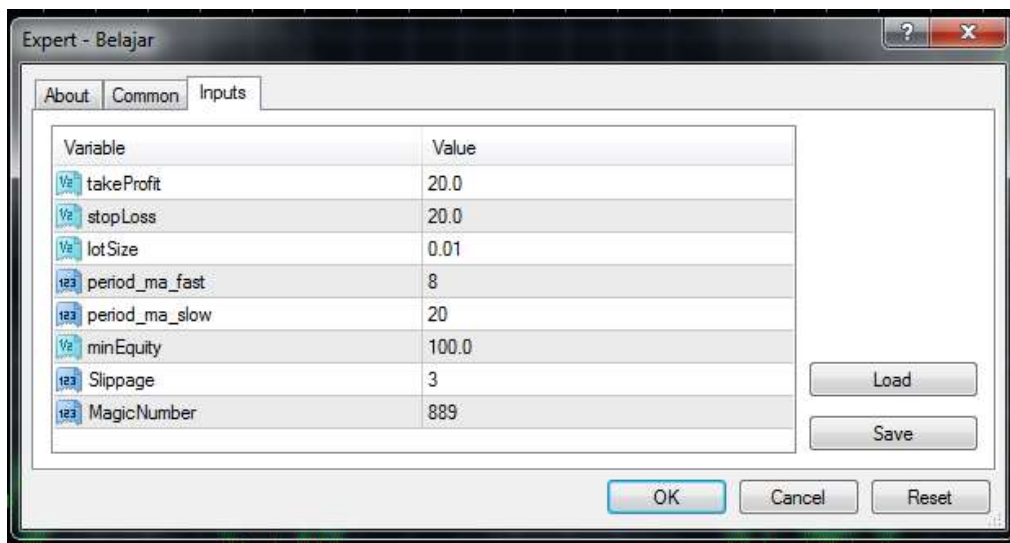
Penjelasan Kode

Kami mulai dengan preprocessor `#property copyright` kami yang mengidentifikasi kode sebagai milik kami dan dilanjutkan pada baris berikutnya `link` website kami. `#property strict` digunakan untuk mengarahkan kompiler yang ketat.

Variable input external didefinisikan diawal dengan format :

```
extern datatype    nama_Variabel = Nilai_Inital;
```

eksternal variable ini akan ditampilkan sekali pada saat pertama kali aplikasi ini diaktifkan



Setelah kita klik tombol OK pada menu input, aplikasi ini akan segera terpasang pada sebuah grafik dan pertama kali akan menjalankan fungsi bawaan yaitu **OnInit()**.

Dimana fungsi **OnInit()** akan melakukan validasi awal terhadap nilai input external tersebut. Bila nilai **takeprofit** kurang dari 0.0 atau **stopLoss** kurang dari 0.0 atau **lotSize** kurang dari 0.01 atau **period_ma_fast** kurang dari 0 atau **period_ma_slow** lebih kecil sama dengan **period_ma_fast**, maka akan menampilkan window pesan "**Warning – Input data initaliasi tidak valid**" dan aplikasi ini akan diberhentikan, sesaat

sebelum berhenti, akan memanggil fungsi **OnDeinit()**. Fungsi **OnDeinit()** akan mencetak informasi ke log expert advisor.

Jika berhasil atau valid, maka akan dilanjutkan dengan memanggil fungsi **PipPoint(Symbol())** dan mengembalikan hasil fungsi ini kedalam variabel global **UsePoint**.

Berikutnya dilanjutkan ke fungsi **GetSlippage(Symbol(), Slippage)**.

Fungsi bawaan **OnInit()** hanya akan dijalankan sekali dan berikutnya akan menjalankan fungsi **OnTick()**. Fungsi **OnTick()** akan berulang-ulang kali dipanggil setiap kali terdapat perubahan harga pasar (tick).

Setiap kali fungsi **OnTick()** dijalankan, akan dimulai dengan cek Ekuitas minimum dengan memanggil fungsi **cekMinEquity()**.

Fungsi **cekMinEquity()** akan mengambil nilai ekuitas akun saat ini dan disimpan dalam variabel **equity**. Kemudian dengan fungsi perbandingan **if** akan membandingkan nilai equity dengan nilai external minEquity, jika equity lebih besar dari minEquity maka akan mengembalikan nilai **true** begitu sebaliknya akan kembalikan nilai **false**.

Proses berikutnya adalah melanjutkan pemanggilan fungsi Moving Average. (Penjelasan detil fungsi Moving Average (iMA ())) akan dijelaskan pada modul berikutnya.

iMA() akan menghitung nilai moving average. Nilai perhitungan iMA() dengan periode 8 akan disimpan dalam variabel **FastMA** dan nilai perhitungan iMA() dengan periode 20 akan disimpan dalam variabel **SlowMA**. Pembukaan posisi pesanan ditentukan pada nilai FastMA dan SlowMA. Jika nilai FastMA lebih besar dari SlowMA dan BuyTicket sama dengan 0 maka terpenuhi kondisi untuk membuka posisi BUY.

Sebelum membuka posisi BUY, kita akan menutup posisi SELL terlebih dahulu jika ada. Kita menggunakan **OrderSelect()** untuk mengambil informasi posisi order SELL dengan mengirim 2 parameter, yaitu nomor tiket (**SellTicket**) dan **SELECT_BY_TICKET**.

Jika berhasil, kita akan melanjutkan dengan menutup posisi order SELL dengan fungsi **OrderClose()** dan mengembalikan SellTicket menjadi nol.

Dan setelah berhasil close posisi SELL, kita akan lanjutkan dengan membuka posisi BUY. Harga **OpenPrice** akan diisi dengan harga permintaan (**Ask**) saat ini. Inialisasi variabel untuk **BuyStopLoss** dan **BuyTakeProfit** dengan nilai **nol**.

Jika external input **stopLoss** lebih besar dari nol, maka **BuyStopLoss** akan dihitung dari **OpenPrice** dikurangi nilai **stopLoss** dikalikan dengan **UsePoint**. Begitu juga dengan nilai **BuyTakeProfit** akan dihitung dari **OpenPrice** ditambahkan dengan **takeprofit** dikalikan dengan **UsePoint**.