SZAKDOLGOZAT

Molnár Tibor

Debrecen
2007

Debreceni Egyetem
Informatika Kar

AUTOMATIZÁLT KERESKEDÉS PROGRAMOZÁSA A DEVIZAPIACON

Témavezető:
Dr. Kormos János
Egyetemi tanár

Készítette:
Molnár Tibor
Programozó matematikus

Debrecen
2007

Tartalomjegyzék

Köszönetemet fejezem ki témavezetőmnek, dr. Kormos Jánosnak, a szakdolgozat írása során nyújtott segítségéért.

1. Bevezetés

A mai világban – sajnos vagy nem sajnos – nagyon fontos a pénz fogalma. Életünk során legtöbbször hazai pénznemünket használjuk, de egyre gyakrabban előfordul az is, hogy egy – manapság teljesen hétköznapi – pénzforgalmi művelet kapcsán idegen pénznemekkel is kapcsolatba kerülünk. Például, ha külföldre utazunk, át kell váltani forintunkat az idegen ország valutájára. Az átváltás történhet bankon keresztül, vagy akár valamely pénzváltó cég is jegyezhet nekünk árat. Az átváltási folyamat kapcsán mi is részesévé válunk a devizapiacnak. A devizapiacnak mégsem a mi költőpénz-átváltásunk lesz a mozgatórugója, annál sokkal nagyobb összegű ügyleteket bonyolítanak egymás között bankok, brókercégek, amelyek alapvetően meghatározzák a piac működését, felépítését.

Célkitűzések

Dolgozatom célja, hogy a devizapiaci alapfogalmak magyarázata után programozott megoldási lehetőséget nyújtsak az árfolyamváltozások prognosztizálásához. Ezzel elősegíthetem a kereskedelmi ügyleteket előkészítő döntéshozatalt és gyakran magát a döntést is. A fő célkitűzésem nem a devizapiaci döntési folyamat menetének megadása, hiszen ez a gazdasági szakemberek dolga, hanem a döntési folyamat – azaz egy megadott stratégia – programozása, automatizálása.

A dolgozat felépítése

A döntés meghozásához rengeteg információ (jelenlegi, azonnali, múltbeli adatok, események) áll rendelkezésre, ezek figyelése, felhasználása ad lehetőséget az automatizálásra.
A fentiek miatt dolgozatom első részében leírom a szükséges alapfogalmakat, melyek ismerete nélkül nem lehet eligazodni a devizapiacon. Ismertetem az árfolyam elemzésének azt a módszerét (technikai elemzés), amely lehetőséget ad arra, hogy a gazdasági események kiszűrése mellett is értelmezni lehessen a devizapiaci eseményeket.
Bemutatok továbbá egy olyan eszközt – a chartot –, amely elengedhetetlen a piac szemmel tartásához. A chart minden információt biztosít nekünk – illetve programunknak – amire szükség van a technikai elemzés lebonyolításához.

A dolgozat második részében bemutatok egy online kereskedési platformot, a MetaTrader 4-et, amely az otthonról, Internet használatával történő devizapiaci kereskedést teszi lehetővé. A platform – számunkra – legfontosabb része, a Client Terminal, mert ebben található a programozáshoz szükséges eszközrendszer, a fejlesztői- és futtatási környezet. A Client Terminal kapcsán bemutatom a programok írásának, futtatásának módját valamint a – most már – programozott kereskedési stratégia tesztelési- és optimalizálási lehetőségeit is.
Dolgozatom harmadik része egy programnyelvről, az MQL4-ről szól. Ez a programnyelv a Client Terminal beépített nyelve. Az alapelemek részletes leírása után bemutatom a programnyelv használatát egy kereskedési stratégia megvalósításával, – jelen esetben tehát – a programozásával. A fejezet tartalmazza a program teljes forráskódját magyarázatokkal ellátva. A program használható devizapiaci előrejelzések generálásához, sőt futtatásával a teljesen automatizált kereskedés is megvalósul, azaz a „program helyettünk kereskedik".

A devizapiaci kereskedés kapcsán viszont a következő megállapítás érvényes, melyet érdemes átgondolni:

„*Úgy tisztességes, ha előrebocsátjuk: egyetlen olyan modell, elmélet, képlet, csodamódszer sincs, amellyel a devizaárfolyamok alakulását minden időben kielégítően megbízható módon előre lehetne jelezni.*"[1]

[1] Érsek Zs. 2002. *Bevezetés a devizapiacokra*. KJK-KERSZÖV Jogi és Üzleti Kiadó Kft. 191. o.

2. Alapfogalmak

Ebben a fejezetben olyan fogalmak magyarázata szerepel, amelyek elengedhetetlenek a dolgozat megértéséhez. A fogalmak bővebb leírása a [1, 2, 3, 6, 7] publikációkban találhatók. Az alapfogalmak az automatizált kereskedéssel, vagy a kereskedés devizapiaci hátterével kapcsolatosak. Vannak olyan devizapiaci fogalmak, amelyek – konvencionálisan – programnyelvi szinten is megjelennek az MQL4-ben.

2.1. A devizapiacról

A hazai szóhasználatban nem mindig érzékelhető a deviza és valuta fogalmak közötti különbség. A **deviza** idegen pénznemre történő számlakövetelést, míg a **valuta** általában magát, a kézzel fogható pénzt jelenti. A devizapiac az a piac, amelyen a devizapiac szereplői egymás között devizát tudnak átváltani. A gyakorlatban ezt a piacot forexnek[2] nevezik, és általában FX-szel jelölik [1]. A forex huszonnégy órás piac, elméletileg valamennyi jegyzett devizára történhet üzletkötés a nap bármely pillanatában. A központosított tőzsdéken folyó forex tevékenység miatt beszélhetünk **devizatőzsdéről** is. Valamennyi devizapiaci üzlettípus tekinthető – az egyik devizáról a másikra történő – egyszerű konverziónak, átváltásnak. Az összes átváltás közel felét teszi ki a **spot** (azonnali, prompt) **üzlet**. Ebben az üzletfajtában a teljesítés az üzletkötést követő második munkanapon történik. A spot piac nagysága például azzal magyarázható, hogy a spekulatív jellegű érdeklődés nagy része ide irányul az üzletek gyors lezajlása miatt. A devizapiac a világ egyik legnagyobb piaca, napi becsült forgalma 2001-es adatok alapján 1210 milliárd amerikai dollár volt [1]. A piac mérete és gyorsasága miatt a kialakuló ár – a piac pillanatnyi helyzete alapján – reálisnak mondható.

Legjelentősebb szereplői a bankok, ezért a devizapiac tekinthető bankközi piacnak. Általában a bankok rendelkeznek akkora tőkeerővel és hitelképességgel, ami a hatalmas méretű ügyletek végrehajtásához szükséges, ezért tranzakcióik alapvetően befolyásolják a piacot.

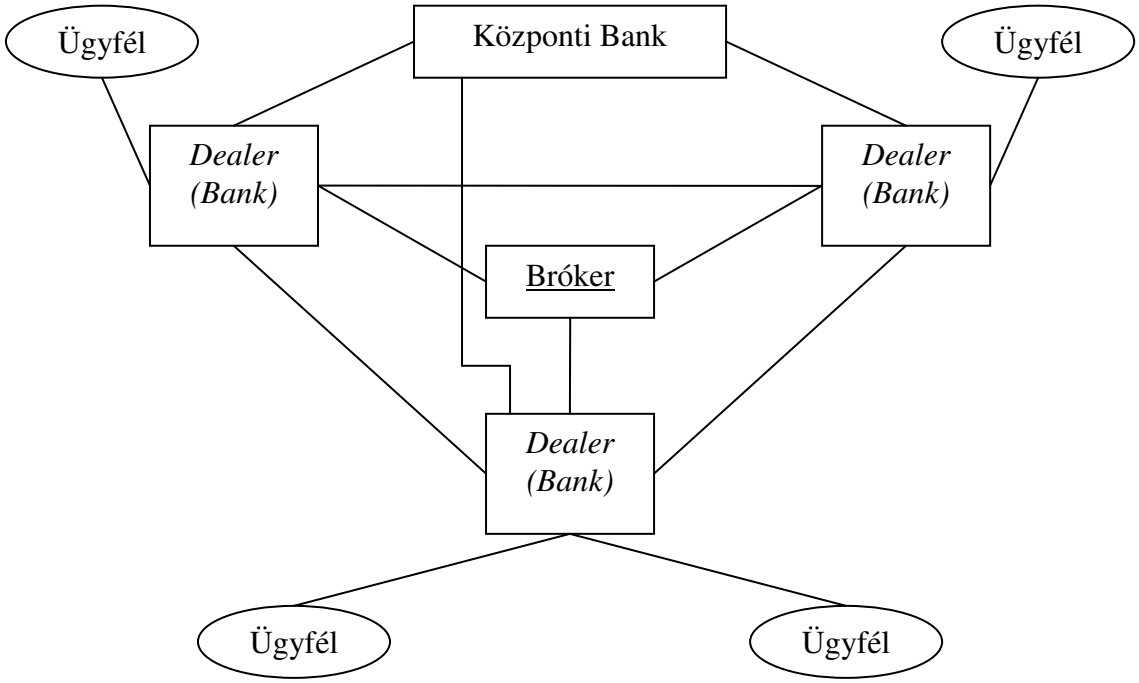
Az **OTC-piac**on[3] történő üzletkötések minden paramétere a két fél szabad megállapodásának eredménye.

[2] Foreign exchange

[3] „Over the counter” – a kereskedés a „pulton történik”. Tőzsdén kívüli piac, a bankközi devizapiac is ennek minősül.

2.1.1. Az OTC-piac felépítése, szereplői

A piac felépítését az 1. ábra mutatja [1].



1**. ábra** *Az OTC-devizapiac szerkezeti felépítése*

Az 1. ábra magyarázata, az egyes szereplők tevékenységi köre:

- **Bróker –** a jegyzett árakat közvetíti a piac szereplői között.
- **Dealer –** feladatai közé tartozik az árak jegyzése valamint az ügyletek realizálása.
- **Központi bank –** nem profitérdekelt. Fő feladata a piac biztonságos működésének felügyelete, és a hazai pénz stabilitásának megőrzése.
- **Ügyfél –** a piac „használói". Ilyenek a befektetők, spekulánsok, különböző pénzügyi intézmények.

2.1.2. Az árfolyamjegyzés

Egy adott deviza (a hivatkozási deviza) ára egy másik devizában (a hivatkozott devizában) kifejezve adja magát az **árfolyam**ot [2, 6]. A devizaátváltás mindig egy meghatározott árfolyamon történik. Az üzletkötő a devizapiacon mindig egy fajta devizát bocsát áruba egy másik devizáért cserébe. Az árfolyam meghatározását és a másik féllel való közlését jelenti a

jegyzés. A jegyzett ár tehát mindig egy bizonyos **devizapár**ra[4] vonatkozik. Egy jegyzett ár mindig konkrét üzleti ajánlat vételre vagy eladásra az azt jegyző banktól.
Például:
1 GBP = 2,0028-2,0031 USD, röviden GBPUSD 2,0028/31

A kisebbik szám a vételi- (**bid**), a nagyobbik az eladási (**ask, offer**) árfolyam a bank szempontjából. A vételi és az eladási ár közötti különbség a **spread** (marzs), ez a bankok nyeresége az átváltási ügyleten. Egy jegyzésben az árak általában négy, vagy két tizedes jegy pontossággal szerepelnek. Egy **pont** (**pip**) az árfolyam legkisebb helyi értékén történő egységnyi változás.

2.1.3. Online kereskedés a devizapiacon

A **pozíció** árfolyamkockázatnak való kitettséget jelent. A devizapiacon való **kereskedés pozíciók nyitásával és zárásával** valósítható meg [1].
A pozíció iránya lehet:

- **Hosszú (long[5]) –** árfolyam emelkedésében való érdekeltség, vételi pozíció.
- **Rövid (short[6]) –** árfolyam esésében való érdekeltség, eladási pozíció.

Az online kereskedéshez egy – ilyen tevékenységet folytató – brókercégnél szükséges számlát nyitni. Ezáltal az ügyfél állandóan frissülő tájékoztatást kap a jegyzett árakról (**beérkező adat, tick data**), és folyamatosan – akár napi 24 órán keresztül – igénybe veheti a brókercég többi szolgáltatását. A brókercégnél elhelyezett pénzösszeg **fedezet**ül (**margin**) szolgál az ügyfél által adott **megbízás**ok (kötések) végrehajtására. A megbízások általában pozíció nyitásával és zárásával kapcsolatosak.
A fedezet lehetőséget nyújt a **tőkeáttétes kereskedés**re [7], amikor az ügyfél megbízás kiadásakor nem csak a saját elhelyezett pénzösszegét használhatja, hanem annak sokszorosát. Ilyen kereskedéssel természetesen nagyobb hozam érhető el, viszont a nagy tőkeáttétel miatt hamarabb elbukható a fedezetül lerakott pénz is.

[4] A MetaTraderben: szimbólum
[5] Az angol *long* szó egyik jelentése: bővében lenni valaminek
[6] Az angol *short* szó egyik jelentése: szűkében lenni valaminek

Egy **azonnali megbízás** az aktuálisan jegyzett áron történő pozíció nyitása, zárása vagy módosítása. **Függő megbízás (pending order)** elhelyezése esetén a vétel/eladás nem azonnal teljesül, hanem egy piaci esemény hatására, például, ha az árfolyam elér egy kívánt szintet.
Azonnali megbízások tulajdonságai:

- **Szimbólum –** az a devizapár, amelyre kereskedünk.
- **Típus** – vétel vagy eladás.
- **Lot –** kötésmennyiség, azaz a vétel/eladás egysége.
- **Stop loss (S/L) szint –** beállítása egy olyan árfolyamszint megadását jelenti, ahol a veszteséges pozíció a veszteség realizálásával automatikusan zárásra kerül.
- **Take profit (T/P) szint** – beállítása egy olyan árfolyamszint megadását jelenti, ahol a nyereséges pozíció a profit kivételével automatikusan zárásra kerül.

2.2. Árfolyamok elemzése

Az árfolyamok változásának prognosztizálására jelenleg nincs olyan módszer, amely megbízhatóan, minden körülmények között jól működik [1]. Az erre irányuló törekvéseknek mégis megadható három fő irányvonala:

- **Fundamentális elemzés –** gazdasági, politikai, társadalmi adatokat használ az árfolyamváltozások előrejelzésére.
- **Technikai elemzés –** a piaci folyamatok tanulmányozása grafikonok segítségével. Legfontosabb eszköz az adott devizapár chart ablaka. A technikai elemzők szerint csak a piacon végbemenő változásokat kell figyelni, mivel azok már minden fundamentális hatást is tükröznek.
- **Hatékony piacok elmélete –** az elmélet szerint az árban az összes elérhető információ benne van, tehát lehetetlen az információk alapján bármiféle jóslásba bocsátkozni. Minden profit, illetve veszteség csak a kockázatvállalásnak köszönhető.

A kereskedésre kialakuló egyéni szokások tekinthetők a **kereskedési stratégiá**nak. A kereskedést automatizálni csak a technikai elemzés szempontjából lehet, hiszen ezen módszer mellett szükséges az adatok folyamatos, azonnali, tömeges vizsgálata, amelyre a számítógép eszközként használható. A fundamentális elemzés gyorsaságát is elősegíti a számítógépes kommunikáció, hiszen az Interneten gyorsan terjednek a gazdaság, politika stb. hírei. A technikai elemzés automatizálhatósága miatt arról külön alfejezetben írok.

2.2.1. A technikai elemzés

Kritikákat fogalmaz meg a fundamentális elemzéssel kapcsolatban [1]:

- Az árfolyamra ható rengeteg tényezőt egy ember nem tudja helyesen súlyozni, az érdektelen adatokat kiszűrni.
- Az adatok begyűjtése, feldolgozása, közzététele időigényes, így a jelzések nem tükrözhetik az aktuális állapotot.
- Az elemzés alapját képező adatok sokszor hibásak, megbízhatatlanok, tehát az elemzés eredménye is az lesz.
- Az elemzésre szolgáló adatok alapján az elemzés lehet jó, de a piac mégis dönthet másként. Az információ a piac szereplői számára nem azonos módon hozzáférhető. De még azonos információ birtokában is elképzelhető, hogy a szereplők egymással ellentétes döntéseket hoznak. Ekkor pedig valamelyik szereplőnek biztosan nem lesz igaza.
- A fundamentális elemzés szükségszerűvé teszi a piac egy-egy részterületének magas szintű ismeretét. Sok devizapárt lehetetlen egyszerre követni.

A technikai elemzés elviekben az összes felvetett problémára megadja a választ. Alapvető különbség, hogy a technikai elemzés csak a piacon bekövetkező változásokat figyeli, azokat elemzi az ár, a forgalom és a nyitott kötésállomány alapján. Három alaptétele:

- ***„Az árban minden más hatás tükröződik."*** – Azaz a fundamentális alapokon nyugvó megközelítés eredményét is tükrözi az ár, tehát elég azt figyelni.
- ***„A történelem ismétli magát."*** – A piaci árakat végső soron az emberek alakítják. Az ember ugyanolyan helyzetben gyakran ugyanazt a döntést hozza meg újra, tehát csak meg kell találni a hasonló lefutású árgörbéket ahhoz, hogy az elemzés sikeres legyen.
- ***„Az árak trendszerűen mozognak."*** – Az árak nem véletlenszerűen mozognak, hanem trendszerűen. Ez azt jelenti, hogy egy megkezdett irányú mozgás folytatódására nagyobb az esély, mint a trend fordulására. Ezt a tételt a fundamentális elemzés is erősíti.

A technikai elemzés nem más, mint a múltbeli adatok és a jelenleg ismert árfolyamgörbék felhasználásával a rövid- illetve hosszú távú trend meghatározása.

Fő ágai:

- **Jellegzetes árfolyamgörbék vizsgálata** – chartelemzés, hasonló alakzatok keresése.
- **Trendkövető módszerek használata** – a trend meghatározása az elsődleges.
- **Piaci állapot-elemzés** – az ár helyességét próbálja meghatározni, például a forgalom mérete, nyitott pozíciók száma alapján
- **Ciklusok, hullámok vizsgálata** – időben általánosan érvényes összefüggések meghatározása a cél.

A devizapiacon érvényes, hogy az összes szükséges információ, alapadat rendelkezésre áll a technikai elemzés lebonyolításához. További segítséget jelentenek a számítógépek, mert folyamatosan képesek az árfolyamgörbén kialakult alakzatok vizsgálatára, indexek elemzésére, sőt, a – technikai elemzés alapján – programozott stratégia megvalósítására, eladási-, vételi jel generálására, adás-vételre.

2.3. A chart

A chart lehetővé teszi az árfolyam változásának vizuális észlelését [3]. A chart ablaknak nagy szerepe van a trendek, alakzatok, formációk felismerésében, ezért a technikai elemzés elengedhetetlen kelléke. Az árfolyamdiagram rajzolására többféle módszer alakult ki, így létezik vonal-, japán gyertya-, oszlop diagram. A 2. ábra egy olyan chart ablak, amely japán gyertyákkal szemlélteti az árfolyam változásait.

2. ábra *Japán gyertya diagram*

Egy egyszerű vonaldiagramnál a japán gyertya diagram több információt hordoz, ezért a továbbiakban azt használjuk.

A chart tulajdonságai

A chart mindig egy adott termék árának (a chart Y tengelye) változását mutatja egy adott időszakra (a chart X tengelye). A devizapiacon ennek a terméknek egy devizapár felel meg. Az a devizapár, amelyet a chart használ a chart **szimbólum**a. Az adatok (jegyzett árak) a piac természeténél fogva folyamatosan érkeznek. Arra, hogy minden adat a chartba kerüljön, a technikai elemzés szempontjából általában nincs szükség, ezért a chart nem minden adatot jelenít meg, hanem azokat egységekbe rendezi. Egy egység az adott időintervallumon belül beérkezett összes adatot reprezentálja. Ez az időegység, **periódus** lehet egy perc vagy akár egy egész hónap is. A 2. ábrán a GBPUSD szimbólum ötperces adategységei láthatók 2007. 04. 23. öt óra tíz perctől nyolc óra tíz percig.

Az ábrán az egységek megjelenítésére japán gyertyák láthatók. Egy gyertya felépítése kétféle lehet aszerint, hogy az általa ábrázolt időszakban milyen volt az árfolyamváltozások iránya:

- **Üres gyertya** – a gyertyatest alja jelzi a nyitó árat az adott periódusra vonatkozóan, a teteje a záró árat.
- **Teli gyertya** – az üres gyertya ellentéte.

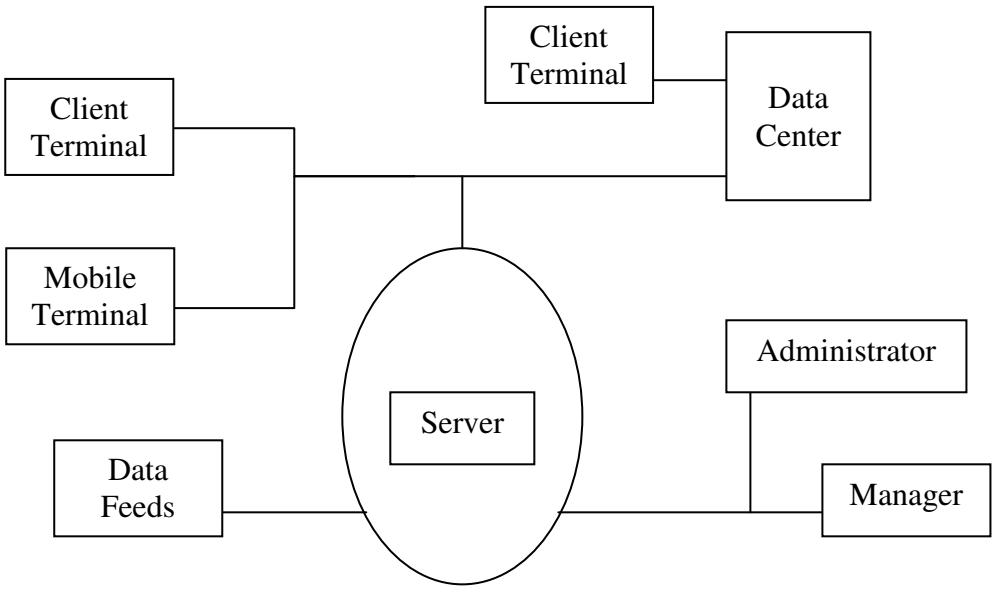
A gyertya „felső kanóca”, **felső árnyék**a jelzi a periódus legmagasabb árát, az „alsó kanóca”, **alsó árnyék**a a legalacsonyabb árát.

3. A MetaTrader 4

A MetaTrader 4 egy olyan komplex kereskedési platform, amelyet használóinak teljes körű kiszolgálására hoztak létre. A platform használata mellett ugyanis nincs szükség más eszközre devizapiaci ügyletek végrehajtásához. A platform különböző típusú devizapiacokon használható, ilyen piac például a forex. A fejezet megírásához a [11] nyújtott segítséget.

3.1. A platform felépítése és működése

Felépítése



3. ábra *A MetaTrader 4 platform szerkezeti felépítése [11]*

Működése

A MetaTrader 4 részei – amelyeket a 3. ábra szemléltet – a következő feladatokat látják el:

- **Server –** a rendszer magja. Minden felhasználói kérés közvetve vagy közvetlenül ide érkezik be. Feldolgozza a beérkező árajánlatokat, és a kereskedéssel kapcsolatos híreket, műveleteket. Megbízásokat fogad, és végrehajtja őket. Data Centereken

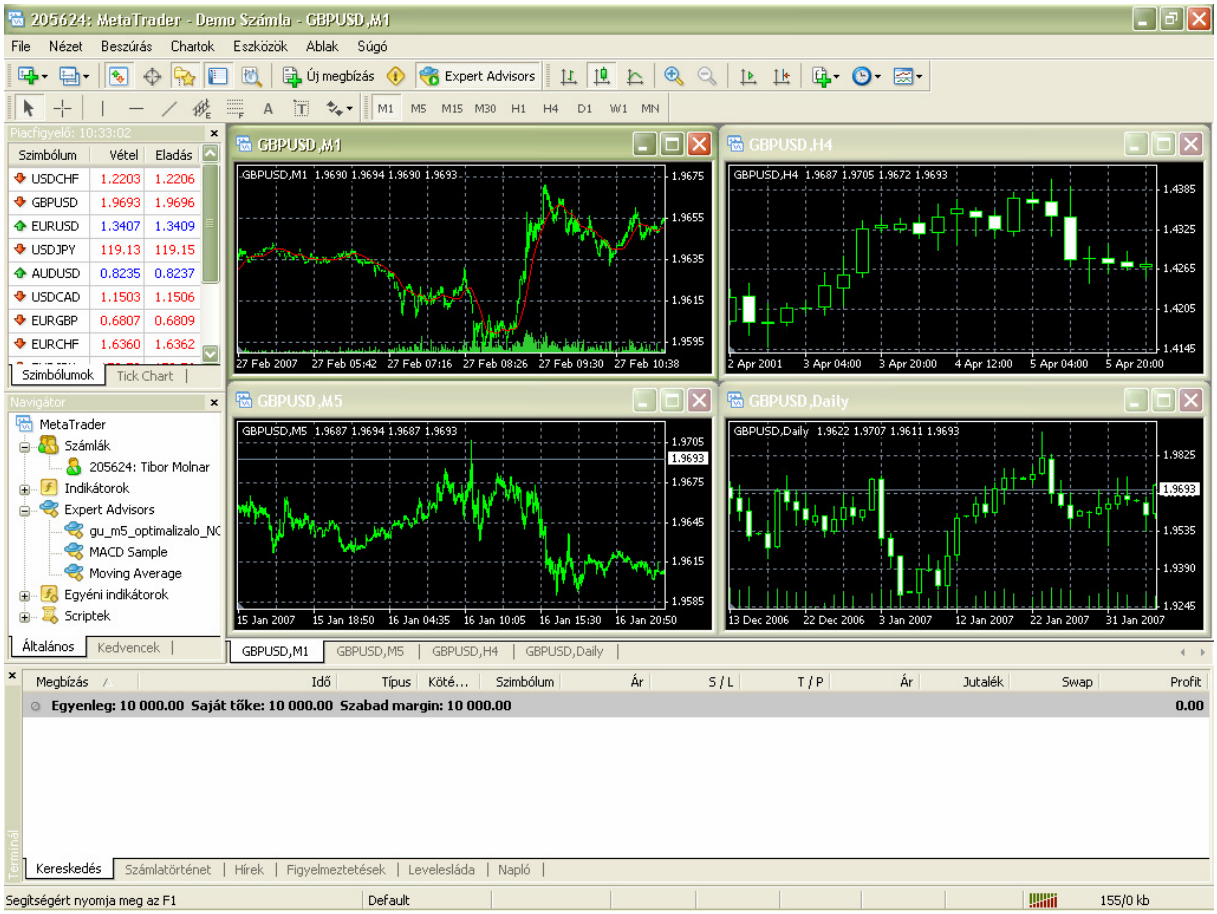
keresztül dolgozik, ami hatékonyan növeli a DDoS[7] támadások elleni védettséget a Server terheltségét csökkentve.

- **Data Feeds –** a platformon elérhető hírek, ajánlatok közvetítője a szerver felé.
- **Mobile Terminal –** PDA vagy mobiltelefon közreműködésével valósítja meg egy felhasználói fiók vezérlését. A kapcsolódás WAP[8]-on vagy GPRS[9]-en keresztül történik.
- **Client Terminal –** a MetaTrader 4 legfontosabb része a kereskedés megvalósításának szempontjából. A Client Terminal teszi lehetővé az MQL4 nyelv használatát, és ezáltal az automatizált kereskedést. A platform ezen része – fontossága miatt – külön fejezetben kerül bővebb bemutatásra.
- **Data Center –** olyan proxy szerver, amely a Server és a Client Terminalok közötti kapcsolat létrehozásáért felelős. Képes bizonyos beérkező kérések kiszolgálására anélkül, hogy a valódi szerverhez továbbítaná őket. Ezáltal kiküszöbölhetővé válik a Client Terminalok közvetlen kapcsolódása a fő szerverhez, és csökken a szerver hálózati forgalma. Például a Data Centerek gyűjtik a múltbéli adatokat, így ha ezen adatokra vonatkozó kérés érkezik be hozzájuk, arra önállóan válaszolnak. A DDoS támadások ellen is hatékonyan lépnek fel azáltal, hogy a valódi szerver IP címét elrejtik. Ha a támadások eredményeként egy Data Center kiesik, a szerver akkor is tovább tudja folytatni a működését. A kieső helyét egy erre a célra tartalékolt Data Center veszi át, ehhez lesznek átirányítva a megfelelő Client Terminalok kérései.
- **Administrator –** a szerverek távoli felügyeletének megvalósítása a feladata. Távoli eléréssel lehetőséget biztosít a felhasználói csoportok kezelésére, Data Centerek irányítására, a rendszer biztonsági és integritási kontrolljára, múltbéli adatok szinkronizációjára. Az Administrator és a Server közötti kommunikáció 128 bites kulccsal titkosított.
- **Manager –** a MetaTrader azon része, amelynek feladata a kereskedési műveletek végrehajtásának irányítása, és a kereskedők felhasználói fiókjainak ellenőrzése.

[7] A DDoS (Distributed Denial of Service) támadások célja egy informatikai szolgáltatás teljes, vagy részleges megbénítása, a helyes működéstől való eltérítése.
[8] A WAP (Wireless Access Protocol) a vezeték nélküli adatátvitel egy nyílt nemzetközi szabványa.
[9] A GPRS (General Packet Radio Service) adatátviteli technológia, amelyet mobiltelefonok használnak.

3.2. A Client Terminal

A Client Terminal (továbbiakban: Terminál) az online kereskedési platform része [12]. Használatához internet hozzáférés és Microsoft Windows 98SE/ME/2000/XP/2003 operációs rendszerek valamelyikével konfigurált számítógép szükséges. A Terminál többnyelvű, használható magyarul is, grafikus felhasználói felületét a 4. ábra mutatja.



4. ábra *A Client Terminal*

Ebben a programban minden eszköz megtalálható az – akár manuális, akár automatizált - online kereskedéshez a devizapiacon.

A következő funkciókat látja el:

- Beérkező adatok (hírek, árfolyamok stb.) fogadása
- Kereskedelmi ügyletek végrehajtása
- Nyitott pozíciók és függő megbízások kezelése
- Technikai elemzés támogatása

- Programok, modulok írásának lehetősége MQL4 nyelven
- Kereskedési stratégiák tesztelése, optimalizálása

3.2.1. A Terminál használata

A futtatható állomány a MetaTrader 4 honlapjáról [11] ingyenesen letölthető. A Terminál installálásának menete és futtatása a szokásos – Windows operációs rendszer alatti – procedúra.

Account nyitása

Ahhoz, hogy valaki jogot kapjon a szerver funkcióinak elérésére, regisztrálnia kell magát. Ez egy account[10] létrehozását jelenti.

Kétféle account nyitására van lehetőség, melyek:

- **Real Account –** csak brókercégen keresztül lehet nyitni, olyan pénzösszeg átutalásával, amelynek minimális összege a brókercég által szabályozott. Szerződéskötés után így lehetőség nyílik a valódi online kereskedésre.
- **Demo Account –** virtuális pénzösszeg használatával lehet kipróbálni a platformot, valódi befektetés nélkül. Minden funkció elérhető, ami Real Account használatával.

A program első indításakor felajánlja Demo Account nyitását. Lehetőség van több számla nyitására, használatára is.

Az azonosító adatok megadása után a Terminál csatlakozni tud a szerverhez.

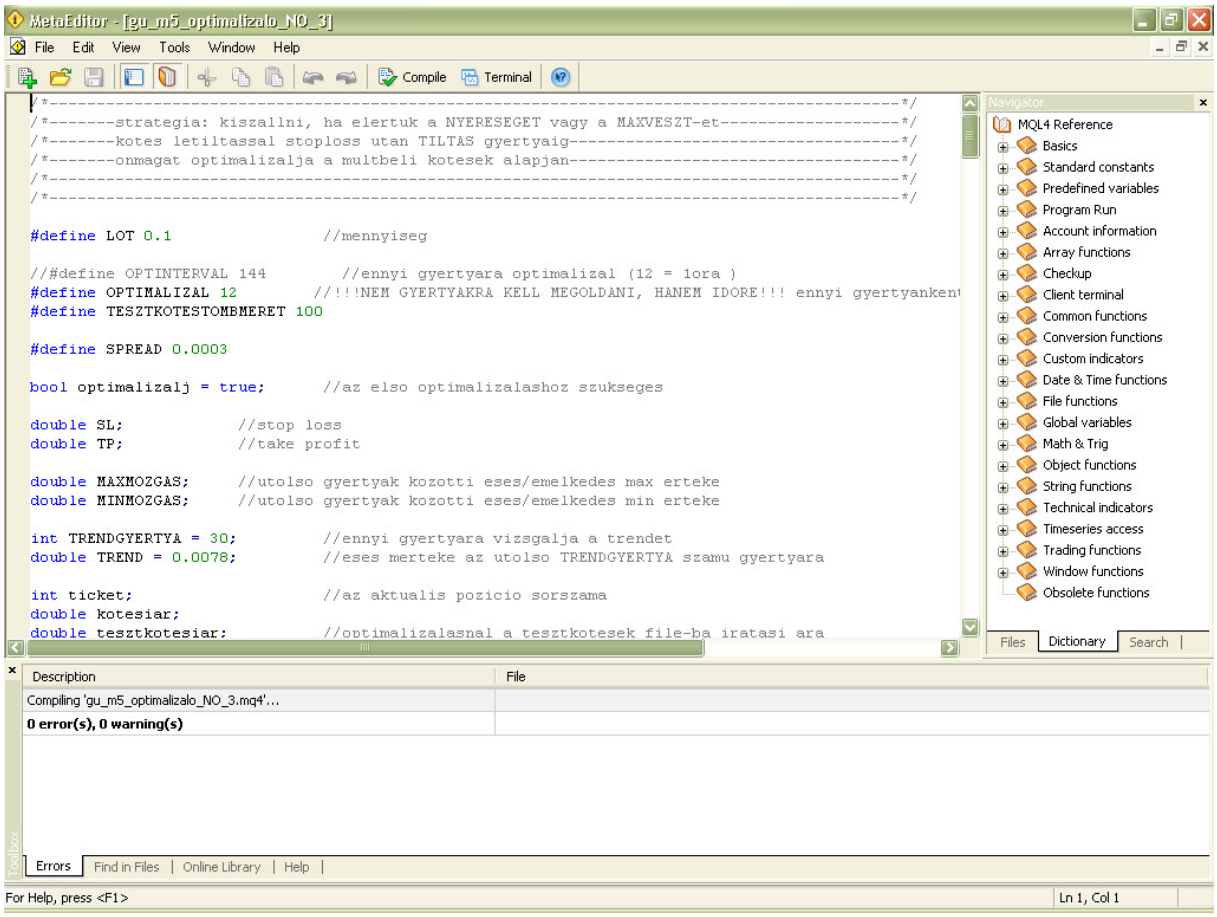
3.2.2. Automatizált kereskedés

Az automatizált kereskedés megvalósítására programozási szinten az MQL4 nyelv használható, amiről bővebben másik fejezetben írok. Ebben a nyelvben – a fentebb már említett – Expert Advisorok írása teszi lehetővé a kereskedés teljes vagy részleges automatizálását. A programfejlesztésre a platformon belül a MetaEditor programozási környezet áll rendelkezésre.

[10] Felhasználói fiók. Felhasználói névből és a hozzá tartozó jelszóból áll. A felhasználó azonosítására használható.

A MetaEditor

A MetaEditor a Terminál egyik alkotórésze, abból közvetlenül az F4 billentyű lenyomásával vagy az „Eszközök/MetaQuotes Language Editor” almenüvel indítható. Az editor grafikus felülete az 5. ábrán látható. A programok írásán, szerkesztésén kívül a MetaEditorban nyílik lehetőség a forrásszöveg lefordítására is. Különböző varázslók segítségével lehet könnyen új program létrehozásába fogni. Az új fájl a MetaTrader megfelelő alkönyvtárába kerül elmentésre. Az MQL4 nyelv szerkezetéről és elemeiről a MetaEditor tartalmaz egy lexikont (5. ábra jobb oldal), ez a nyelv teljes leírása példákkal illusztrálva.



5. ábra *A MetaEditor*

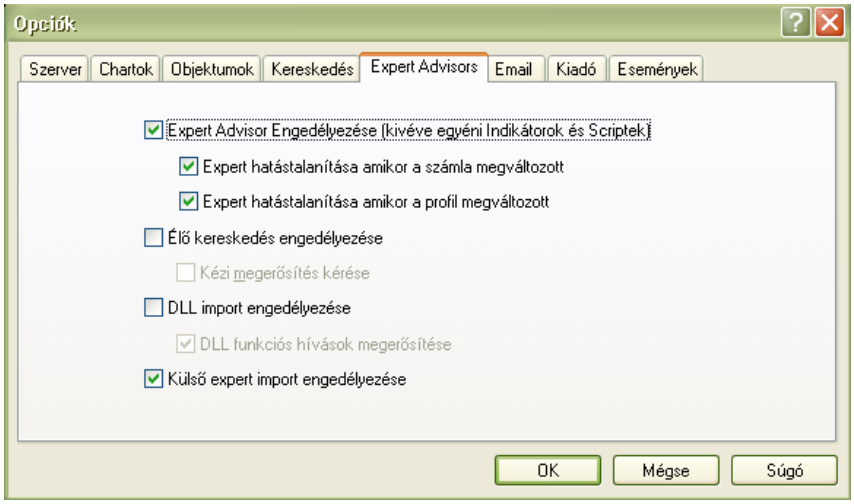
A program megírása és sikeres lefordítása után a futtatható állományt a Terminál Stratégiai Teszter részében lehet tesztelni, vagy chart ablakhoz rendelve valós időben futtatni.

Expert Advisorok

Az Expert Advisorok (expertek) olyan programok, amik a Terminálban futtathatók, és az automatizált kereskedést hivatottak megvalósítani. Azonnali és folyamatos technikai elemzésre képesek a beérkező adatok alapján, és az elemzés eredményétől függően feladatuk lehet a kereskedés realizálása is. Minden – a kereskedéssel kapcsolatos – feladat rábízható az expertekre. Bármelyik szimbólum és megadott periódus adatait használhatják.

Beállítások

Az expertekre vonatkozó globális tulajdonságokat a Terminálban az „Eszközök/Beállítások” almenüben lehet megadni. A beállítások opcióit a 6. ábra szemlélteti.



6. ábra *Expert Advisorok általános beállításai*

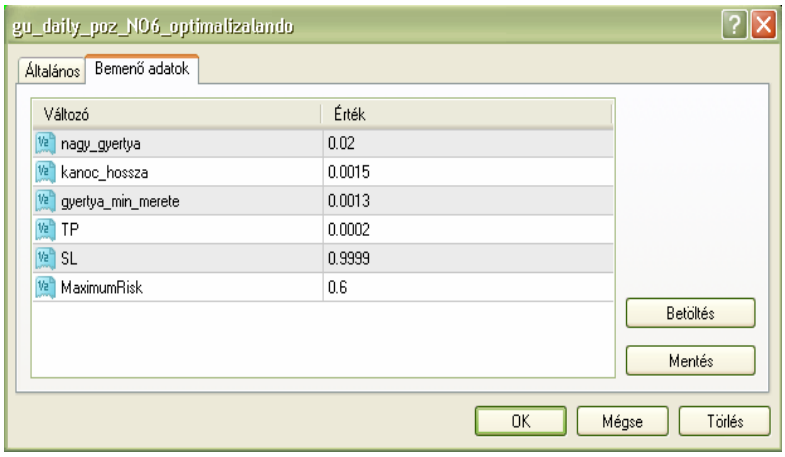
Ahhoz, hogy az expertek működjenek a chartokon, az „Expert Advisorok Engedélyezése” és az „Élő kereskedés engedélyezése” előtti jelölőnégyzeteket ki kell pipálni. Ha a teljes automatizálás a cél, akkor a „Kézi megerősítés kérésére” nincs szükség, de ha valamilyen felügyeletet szeretnénk az expert működése felett, akkor ezt is ki kell pipálni. Ekkor, ha a program kereskedelmi ügyletet szeretne végrehajtani, azt felugró ablak formájában közli a felhasználóval. Ez a funkció a felügyeleti szerep mellett az expertek élő kereskedésben való tesztelésére is alkalmas.

A többi beállítás a külső importtal kapcsolatos, ezek főleg olyan expertek futtatásakor használandók, amelyek nem megbízható forrásból származnak.

Futtatás

A futtatáshoz először a Terminál „Navigátor” ablakából az ott megjelenő Expert Advisorok közül ki kell választani a megfelelőt. Az expertet az éppen aktív chart ablakhoz csatolhatjuk azzal, ha duplán kattintunk a nevére. Lehetőség van arra is, hogy a kiválasztott expertet a megfelelő chart ablakra helyezzük úgynevezett drag & drop művelettel.

Csatoláskor megjelenik a megfelelő expert általános beállításaira vonatkozó ablak, ahol megadhatók a futásához szükséges adatok. Ez az ablak hasonló a 6. ábrához. A program bemenő paramétereit a 7. ábrán látható „Bemenő adatok” fülön lehet megadni, abban az esetben, ha a csatolt programban van extern változó[11] deklarálva. Ha nincs extern változó, akkor a „Bemenő adatok” fül nem jelenik meg az ablakban.



7. ábra *Programok bemenő paramétereinek beállítása*

Az ablakon beállított adatok elmenthetők, illetve betölthetők.

Az aktív chart ablakhoz csatolt expert tulajdonságai az F8 billentyű lenyomásával vagy jobb klikk után az „Expert Advisors/Tulajdonságok” menüpontban érhetők el.

Ha egy charthoz expert van csatolva, akkor az a chart ablakának jobb felső sarkában jelenik meg a nevével és mellette egy ikonnal. A megjelenő ikon háromféle lehet, jelentésük:

- ✖ - a Terminál globális beállításainál nincs engedélyezve az expertek futtatása
- ☹ - az expert általános tulajdonságainál nincs engedélyezve az élő kereskedés
- ☺ - az expert fut

[11] Extern változó – külső változó: olyan változó, amelynek érték adható a Terminál megfelelő ablakában is, nem csak a program forrásszövegében

Az expert leállítására valamelyik engedélyezés letiltásával, eltávolítására a chart ablakon jobb klikket nyomva az „Expert Advisors/Eltávolítás” menüponttal van lehetőség.
Egy chart ablakhoz egyszerre csak egyetlen expertet lehet csatolni. Ha egy olyan charthoz akarunk expertet csatolni, amelyikhez már van, akkor a már bent lévő befejezi működését, és a helyére kerül az új.

Az expertek futásával kapcsolatos további fontos információk:

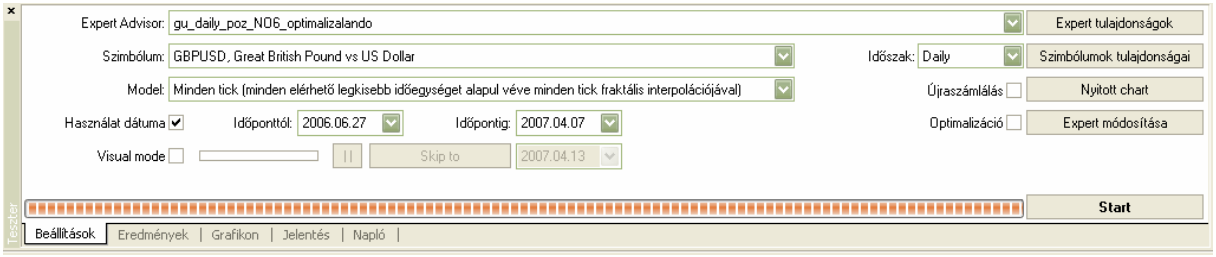
- A Terminál bezárásakor az összes expert futása is befejeződik
- A chart bezárásakor a hozzá csatolt expert futása is befejeződik
- A Terminál „Navigátor” ablakából való törléskor a charthoz csatolt expert futása nem áll meg

3.2.3. A Stratégiai Teszter

A Terminál a - speciálisan erre a célra beépített - Stratégiai Teszter (továbbiakban: Teszter) segítségével lehetővé teszi az expertek tesztelését éles használat előtt. Ezzel a funkcióval vizsgálható a programozott stratégia hasznossága múltbéli adatokon. A tesztelés lehetőséget biztosít a szélesebb körben történő, más szimbólumokon és más időegységeken alapuló vizsgálatára is.
A Teszter az expertek paramétereinek optimalizálására is biztosít megoldást.

A Stratégiai Teszter használata

A Terminál képernyőjén jelenik meg a Stratégiai Teszter ablaka is (8. ábra). Ennek a láthatóságát a „Nézet/Stratégiai Teszter” menüpont alatt, vagy a Ctrl+R billentyűkombináció lenyomásával lehet be-, illetve kikapcsolni.



8. ábra *A Stratégiai Teszter beállításai*

A 8. ábra magyarázata (a Teszter „Beállítások” fülének mezői):

- **Expert Advisor** – ebben a legördülő menüben jelennek meg a Terminál megfelelő könyvtárában lefordított expertek. Itt kell kiválasztani azt az expertet, amelyik tesztelésre vagy optimalizálásra kerül.
- **Szimbólum** – ebben a menüben azt a szimbólumot (devizapárt) kell megadni, amelyre az expert tesztelése során szükség lesz. Például: GBPUSD (angol font / USA dollár)
- **Model** – a modellezési módszert lehet itt kiválasztani. Legmegbízhatóbb, legpontosabb tesztelési eredményt általában a „Minden tick” választás eredményez.
- **Időszak** – annak az időegységnek a típusát kell itt megadni, amelyből az expert a tesztelés során az adatokat nyeri (egyperces adat – napi adat). Általában a **Szimbólum** és az **Időszak** beállításai azonosak az expert későbbi felhasználási beállításaival, azaz a chart tulajdonságaival, amelyhez csatolásra kerül.
- **Használat dátuma** – az az időintervallum adható meg napokra lebontva, amelyre a tesztelést el szeretnénk végezni. Bizonyos tesztelési intervallumokon előfordulhat, hogy nem áll rendelkezésre a megfelelő múltbéli adat, azaz nem végezhető el a tesztelés, vagy csak nem a megadott intervallum kezdő időpontjától.
- **Visual mode** – ha ki van jelölve, akkor a Teszter egy új chart ablakot nyit a tesztelés vizualizálására, amiben a kezdő időponttól indulva folyamatosan mutatja a piac akkori tényleges (vagy modellezett) változásait. Az expert által létrehozott piaci kötések, és azok lezárásai is bekerülnek a chartba. A „Skip to” gomb megnyomásakor a megadott időpontig ugrik a vizualizálás.
- **Újraszámlálás** – ha ki van pipálva, akkor a tesztelés lefutása előtt a Teszter újrakalkulálja a megadott szimbólumra vonatkozó adatokat az adott intervallumon.
- **Optimalizáció** – ha kipiáljuk, akkor nem tesztelés, hanem optimalizálás fog lezajlani a Start gomb megnyomásakor.
- **Expert tulajdonságok** – az expert futás idejű paramétereit lehet itt megadni. Például: kezdő letét, extern változók adatai (optimalizálásra is), optimalizálási tulajdonságok
- **Szimbólumok tulajdonságai** – a kiválasztott szimbólum piaci tulajdonságait jeleníti meg egy felugró ablakban.
- **Nyitott chart** – a tesztelés eredményét jeleníti meg egy külön chartban
- **Expert módosítása** – a kiválasztott expertet megnyitja szerkesztésre a MetaEditorban

- **Start** – megnyomásával indítható a tesztelés vagy az optimalizálás.

A 8. ábra a „gu_daily_poz_NO6_optimalizalando” nevű expert tesztelése utáni állapotot mutatja. Napi (daily) adatokon dolgozva a „Minden tick” modellezést választva a 2006. 06. 27-től 2007. 04. 07-ig terjedő időszakra tesztelt.

Tesztelési eredmények

A tesztelési folyamat lezajlása után a Teszterben elérhetővé válnak a tesztelési eredmények, néhány további fül megjelenésével (8. ábra alja). A fülek magyarázata:

- **Eredmények** – a tesztelés során létrejött kereskedelmi ügyleteket tartalmazza táblázatos formában. A táblázat oszlopai rendre: sorszám, időpont, típus, megbízás, kötésegységek, ár, S/L, T/P, profit, egyenleg.
- **Grafikon** – a kötések okozta változásokat mutatja az egyenlegre vonatkozóan. *X tengelye* a kötések sorszámát, *Y tengelye* az egyenleget jelenti. Kék vonallal jelzi a grafikonon az aktuális egyenleget, zöld vonallal a felhasználható egyenleget[12], és ha a kötésegységek a tesztelés során változnak, akkor a grafikon alján függőleges zöld oszlopok mutatják a megfelelő mennyiségeket.
- **Jelentés** – a stratégia eredményességének összegzése.
- **Napló** – az expert standard kimenete. A kereskedési műveletek végrehajtásával kapcsolatos információk itt jelennek meg. A tesztelés után a napló automatikusan mentésre kerül a tester/logs könyvtárban.

Lehetőség van a tesztelési eredmények mentésére jelentésként HTML formátumban, illetve a grafikon mentésére képként, gif formátumban.

Expertek optimalizálása

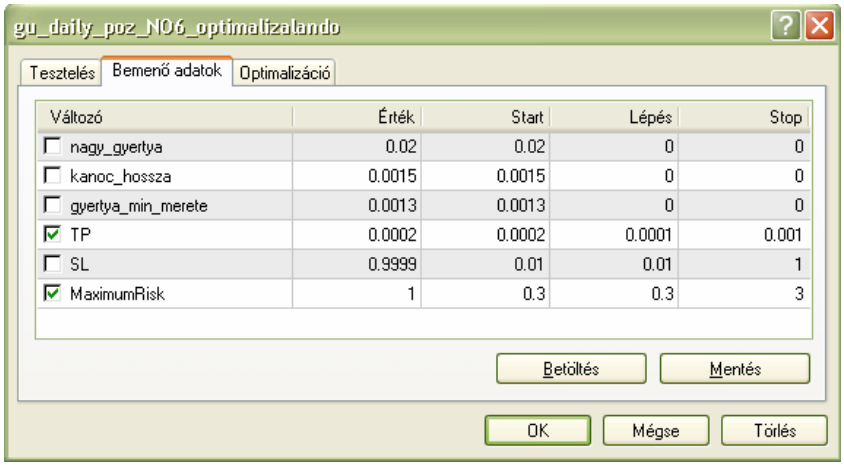
Az optimalizálás ugyanazon expert sokszori tesztelését jelenti automatizáltan a bemenő paraméterek – azaz a program extern változóinak – változtatásával. Eredményeként kiválasztható az expert leghatékonyabb beállítása. Az optimalizálás lehetősége rengeteg munkától szabadítja meg a felhasználót.

[12] Az a pénzösszeg, ami nincs lekötve a nyitott pozíciók fedezetéül.

Az optimalizálást is a Teszterben lehet elvégezni, ha az „Optimalizáció”-t bekapcsoljuk, és a „Start” gombot megnyomjuk. Mivel az optimalizálás is a tesztelési folyamatokra épül, az alapbeállításait ugyanúgy kell megadni, mint tesztelés esetén (8. ábra és magyarázata).
Az „Expert tulajdonságok” nyomógombra kattintva beállíthatók a tesztelési adatok, a bemenő paraméterek intervalluma és lépésköze, valamint egyéb optimalizálási lehetőségek.

Bemenő paraméterek megadása

A „Bemenő adatok” fülön az expert extern változói jelennek meg azzal a kezdőértékkel, ami a forráskódban található. Az optimalizálási folyamat során az expert az összes itt kijelölt változó intervallumának („Start”, „Stop”) és lépésközének megfelelően minden lehetséges beállításra lefut. Az esetleges nagy számolási igény miatt az optimalizálás rendkívül időigényes művelet lehet.



9. ábra *Optimalizálási beállítások*

Az optimalizálandó változót ki kell pipálni. Kipipáláskor alapértelmezetten beállításra kerülnek a start, lépés és stop értékek, amiket érdemes felülvizsgálni. A 9. ábrán a *TP* és a *MaximumRisk* paramétereket jelöltem ki optimalizálásra. Ez alapján a tesztelési folyamatok darabszáma már kiszámolható: az intervallumában a *TP* változó kilenc, a *MaximumRisk* változó pedig tíz lehetséges értéket vehet fel, a lépésközének megfelelően. A darabszám ezen értékek szorzata lesz, azaz $9 \times 10 = 90$.
A start, stop és lépésköz tulajdonságokat úgy célszerű megadni, hogy ne legyen túl nagy a tesztelési folyamatok száma, de ez ne csorbítsa az optimalizálás eredményességét.

Az „Optimalizáció” fülön olyan további beállítások adhatók meg, amelyek az optimalizálási folyamat során szűrőként működnek. Ha valamelyik beállítás teljesül, akkor a paraméterek azon beállításai az optimalizálás szempontjából értéktelenek.

Az „Optimalizáció” fül lehetséges beállításai:

1. táblázat *Optimalizálási szűrőfeltételek*

Korlátozás	Érték
Minimum egyenleg	200
Profit maximum	10000
Minimális margin szint %	30
Maximális lehívás	70
Folyamatos, egymást követő veszteség	5000
Folyamatos, egymást követő veszteséges kötések	10
Folyamatos, egymást követő nyereség	10000
Folyamatos, egymást követő nyereséges kötések	30

Az 1. táblázatban szereplő mezőket úgy kell beállítani, ahogyan a kereskedési stratégia megkívánja.

Optimalizálási eredmények

Az optimalizálás hatására további két fül jelenik meg a Teszterben:

- **Optimalizációs eredmények –** a profittal lezárult tesztelési folyamatok tulajdonságait írja le táblázatban. A táblázat oszlopai rendre: folyamat sorszáma (pass), profit, összes kereskedés, profit faktor, várt eredmény, visszaesés, bemenő adatok
- **Optimalizációs grafikon –** az optimalizálási eredmények között felsorolt tesztek eredményességét mutatja

4. Az MQL4 nyelv bemutatása

Az MQL4 (MetaQuotes Language 4) a MetaTrader 4 platform beépített nyelve [4, 5, 9, 10]. Azzal a céllal került a platformba, hogy felhasználásával le lehessen programozni a devizapiaci kereskedési stratégiákat.

Imperatív, eljárásorientált nyelv.

Ez a nyelv lehetőséget ad arra, hogy – **Expert Advisor** létrehozásával – megvalósítható legyen a teljesen automatizált kereskedés, saját kereskedési stratégia alapján.

A fejlesztői környezet a MetaEditor 4. A MetaEditor tartalmaz egy egyszerű szövegszerkesztőt – amelyben színek emelik ki a különböző funkciójú kódrészleteket –, valamint a compilert. A MetaEditor 4 **mq4** kiterjesztéseket használ a forrásfájloknál, amiket **ex4** kiterjesztésűre fordít.

Az MQL4-ben írt forrásfájlok működésük szerint a következő típusúak lehetnek:

- **Expert Advisor (EA, expert)** – automatizált kereskedő program, amely egy adott charthoz kapcsolva működik, annak beállításait használva alapértelmezésként. Minden, a charton beérkező árfolyam adatra elkezdi futását, kivéve, ha az előző adattal kapcsolatos műveletsort még nem végezte el. Egy expert képes informálni használóját, ha paraméterei, és a tőzsdei mozgások alapján úgy ítéli meg, hogy pozíció nyitása esedékes, illetve informálás mellett vagy helyett pozícióba is állhat, azaz a megfelelő utasítást is elküldheti a kéréseket váró szervernek. A MetaTrader 4, mint általában a kereskedelmi platformok, lehetőséget biztosít az expertek tesztelésére múltbéli adatokon.
- **Custom Indicator** – a beépített indikátoroktól függetlenül készített technikai indikátorok gyűjtőneve. Önálló, automatizált kereskedésre nem képesek. A technikai elemzés implementálására szolgálnak.
- **Script** – olyan program, amely az expertekkel ellentétben nem fut le minden egyes beérkező tick adatra, hanem csak meghatározott kérésekre.
- **Library** – a különböző programok által legtöbbet használt függvények gyűjteménye, futtatni nem lehet.

- **Included File** – a legtöbbször használt programozási eszközök forrásszövegének gyűjteménye. Ilyen típusú fájlokat lehet inkludálni a fordítási szakasz folyamán az expertekbe, scriptekbe, indicatorokba és librarykba. Az included file-ok használata célszerűbb a libraryk használatánál, mert az utóbbiak függvényeinek meghívása további nehézséget jelent.

4.1. Alapelemek

Az MQL4 nyelv általában a C programozási nyelv felépítését követi a következő kivételekkel:

- NINCS mutatóaritmetika, do-while ciklus, goto utasítás, háromoperandusú feltételes utasítás, struktúra
- A logikai kifejezések kiértékelése mindig teljes

Van egysoros (`//...`), és többsoros (`/*...*/`) megjegyzés is.
Az azonosítók használata a nyelvben a szokásos módon történik.

4.1.1. Kulcsszavak

A 2. táblázat tartalmazza a nyelv foglalt szavait. Ezekhez az azonosítókhoz a nyelv rendel jelentést, amely nem változtatható meg a programozó által.

2. táblázat *Az MQL4 kulcsszavai*

Típusok	Memóriatárolási osztályok	Operátorok	Egyéb
bool	extern	break	false
color	static	case	true
datetime		continue	
double		default	
int		else	
string		for	
void		if	
		return	
		switch	
		while	

4.1.2. Típusok, konstansok

Logikai típus

Alapszava: `bool`

Konstansai: `true, TRUE, false, FALSE`

Numerikus jelentésük 1 illetve 0.

```
bool igaz = TRUE;
```

Egész típus

Alapszava: `int`

Értéktartománya -2147483648 és 2147483647 közé esik. Lehet decimális, és hexadecimális.

Példák konstansokra:

```
1234
0xab, 0Xab, 0XAB
```

A nyelv a karaktereket is egészként reprezentálja, nincs külön típusuk vagy tartományuk. A karakterkonstans aposztrófok között helyezkedik el.

Például: `'c'`

```
int a = 5;
int b = 'b';
```

Valós típus

Alapszava: `double`

Értéktartománya -1.7 * e-308 és 1.7 * e308 közé esik. Felépítése: egészrész, tizedespont, törtrész.

```
double a = 12.123;
```

Szöveges típus

Alapszava: `string`

A szöveges konstans idézőjelek között elhelyezkedő ASCII karakterek sorozata. A megadott karaktersorozat minimális hossza 0, maximális hossza 255 karakter lehet.

Példa szöveges konstansra:

```
"Ez egy szoveges konstans"
```

Szín típus

Alapszava: `color`

Konstansait háromféle módon lehet megadni: speciális literállal, egész számmal vagy névvel. Speciális literálja C betűvel kezdődik, utána aposztrófok között kell megadni a három alapszín (red, green, blue) mennyiségét vesszővel elválasztva.

```
C'128,128,128'
```

Egész számmal hexadecimálisan 0xRRGGBB formában, decimálisan pedig a hexadecimális értékre visszavezetve lehet megadni.

A névvel megadott színkonstans általában megegyezik a szín angol nevével.

Például:

```
Black, DarkGreen, SandyBrown, LightSkyBlue
```

Dátum és idő típus

Alapszava: `datetime`

Konstansa D betűvel kezdődik, utána aposztrófok között egy hattagú számsorozat szerepelhet egész számmal megadva, megfelelő karakterrel elválasztva. A sorozat elemei rendre: év, hónap, nap, óra, perc, másodperc. A dátum részt fordított sorrendben is meg lehet adni. A dátum elválasztó karaktere a pont, az idő elválasztó karaktere a kettőspont. A literál dátumot vagy időt kifejező része is elhagyható, esetleg mindkettő. Értékét a 1970. 01. 01. - 2037. 12. 31. intervallumon veheti fel.

Reprezentálására előjel nélküli egészeket használ a nyelv. Egy ilyen típusú változó az 1970. 01. 01. 0:00 óta eltelt másodpercek számát jelenti.

Példák dátum konstansokra:

```
D'2011.12.13 14:15:16'
D''
```

4.1.3. Típuskonverzió

Csak implicit típuskonverziót ismer a nyelv, ami kifejezések kapcsán adódhat. A konverzió akkor jöhet létre, ha két, nem egyforma típusú operandus találkozik egy műveletben.

A típusok konverzió szerinti prioritása a következő módon csökken:

```
string
double
int (datetime, color, bool)
```

Példák:

```
/* nincs tipuskonverzio, az eredmeny 0 */
int i = 12 / 34;

/* a masodik operandus miatt a kifejezes double, ez tipuskonverzioval a cel
tipusára valtozik, tehat az eredmeny 0 */
int j = 1 / 2.0;

/* a kifejezes double tipusu, az eredmeny 0.5 */
double d = 1 / 2.0;

/* a kifejezes int tipusu, ez tipuskonverzioval a cel tipusara (double)
változik, az eredmény 0.0 */
double e = 12 / 34;

/* a kifejezes double tipusu, ami atalakul a cel tipusava (string), az
eredmeny tiz karakteren tarolódik: "0.50000000" */
string s = 1.0 / 2;

/*konverzio utan az eredmeny: "Ticket #12345"*/
string t = "Ticket #" + 12345;
```

4.1.4. Operátorok, kifejezések, utasítások

A C-ben szokásos aritmetikai, relációs, logikai, bitenkénti logikai és értékadó operátorok találhatók meg az MQL4-ben is. Használatuk is C-szerű. Az inkrementáló és dekrementáló operátorok postfixek.

A vessző operátorral (,) elválasztott kifejezések kiértékelése balról jobbra történik.

Az olyan kifejezés értéke, amelyben hasonlító (relációs) operátor van, 0 vagy 1, aszerint, hogy a kifejezés hamis, illetve igaz.

A nyelv precedenciatáblázata a Függelékben található.

Egy olyan kifejezés, mint az `x = 0` vagy `i++` utasítássá válik, ha egy pontosvesszőt írunk utána:

```
x = 0;
i++;
```

Az utasításlezáró jel tehát a pontosvessző.

A kapcsos zárójelek használatával deklarációk és utasítások egy csoportját lehet együtt kezelni összetett utasításba (blokkba) szervezve, ami szintaktikailag egyetlen utasítással egyenértékű.

Az MQL4-ben megtalálható utasítások:
Összetett- (blokk), kifejezés- (értékadó, függvényhívó, üres), break-, continue-, return-, feltételes- (kétirányú elágaztató), switch- (többirányú elágaztató), ciklusszervező- (while, for), üres utasítás.
Az utasítások használta általában C-szerű.

Ha a return utasítás után kifejezés áll, azt zárójelbe kell tenni. Ekkor a kifejezés értéke kerül átadásra a hívó alprogramnak. Ha az alprogram visszatérési típusa void, akkor a return utasítás után kifejezés nem állhat, vagy az egész utasítás elhagyandó. Az ilyen alprogramok esetén a törzset lezáró kapcsos zárójel egy kifejezés nélküli return utasítás implicit futtatását is jelenti.

4.1.5. Függvények

A függvény egy olyan névvel ellátott programozási eszköz, amely arra hivatott, hogy a program szövegében egyszer már megírt kódrészletet újrafelhasználhatóvá tegyen.
Függvény deklarációjakor meg kell adni annak visszatérési típusát, nevét, formális paramétereit zárójelek közé téve, valamint az újrafelhasználandó kódrészletet egy blokkba zárva. Az MQL4 nyelvben egy függvénynek maximálisan 64 paramétere lehet.

```
int pelda_fgv(int a, int b)   // a fuggveny feje
{                             // a fuggveny torzsenek kezdete
      return (a * b);         // a fuggveny visszateresi erteke
}                             // a fuggveny torzsenek vege
```

A `return` utasítás az utána megadott kifejezés értékét adja vissza a hívó alprogramnak. Ha szükséges, a kifejezés értéke a függvény visszatérési típusára konvertálódik, ha a típusok megengedik. Az olyan függvénynek, melyben nincs return utasítás, vagy a return önmagában áll – azaz nem ad vissza értéket a hívó alprogramnak –, kötelezően void visszatérési típusúnak kell lennie.

A függvény paramétereinek megadható alapértelmezett érték, amelyet megfelelő típusú konstanssal lehet beállítani a függvény deklarációjakor.

```
int pelda_fgv2(int a, int b, int c = 0, int d = 1, int e = 2)
{
      return (e);
}
```

Ha a függvény egyik paramétere rendelkezik alapértelmezett értékkel, akkor a paraméter listán utána következő összes paraméternek kötelező alapértelmezett értéket adni.

Függvényhívás

Ha egy kifejezésben olyan azonosító szerepel, amely a kifejezést tartalmazó alprogramban nincs deklarálva, és az azonosító után nyitó zárójel helyezkedik el, akkor azt az azonosítót a fordító egy függvény neveként értelmezi.
A `fgv_nev` nevű függvény hívása:

```
fgv_nev(kif_1, kif_2,…, kif_n)
```

A nyelvben érték szerinti paraméterátadás van, azaz először minden – a paraméterlistán lévő – kifejezés kiértékelődik, majd az értéke átadódik a függvénynek. A program fordítása során a rendszer ellenőrzi a függvényhíváskor megadott paraméterek számát és típusát, és egyezteti a formális paraméter listán találhatóval.
A függvényhívás egy kifejezés, értéke a függvény visszatérési értékével egyezik meg.
Függvénydeklaráció a programban bárhol elhelyezhető, másik függvény törzsét kivéve.
Függvényhíváskor azoknak a paramétereknek, amelyeknek van alapértelmezett értékük, nem kötelező értéket adni, de csak abban az esetben, ha utána a többi paramétert is elhagyjuk az aktuális paraméterlistáról. Szemléletesen: nem lehetnek „lyukak” az aktuális paraméterlistán.

```
/* a hivas hibas a harmadik parameter elhagyasa miatt! */
pelda_fgv2(5,5, ,3,4);
```

Speciális függvények

A nyelvben létezik három – a programok futásának szempontjából speciális jelentőséggel bíró – alapfüggvény:

init()

A modul[13] kezdőértékeinek beállítására használható. A programok működésüket mindig az `init()` függvény futtatásával kezdik, ha az létezik bennük. Ha nem létezik, akkor inicializáláskor nem hívódik függvény. Charthoz csatolt expert esetén annak `init()` függvénye a Terminál indításakor és a chart lényeges tulajdonságainak – szimbólum és/vagy intervallum – megváltoztatásakor újra lefut.

start()

Ez a modulokban a legfontosabb függvény. Charthoz csatolt Expert Advisorok esetén minden adat beérkezésekor meghívódik. Viszont az újonnan beérkező adatokat nem veszi figyelembe a program, amíg a `start()` függvény be nem fejezi – az előző adatra – a működését.

Custom indicatorok esetén a `start()` az indikátor charthoz csatolásakor, a Terminál megnyitásakor (ha az indikátor valamelyik chart-hoz csatolt) és minden beérkező adat esetén lefut.

Scriptek esetén a script charthoz csatolása és inicializálása után a `start()` függvény rögtön lefut.

Amelyik modulban nincs `start()` függvény, az nem indítható, nem használható.

deinit()

A modul deinicializáló függvénye. Ha nem létezik, akkor deinicializáláskor nem hívódik függvény. A programok befejeződésekor fut le: a Terminál vagy a chart ablak bezárásakor, és a chart lényeges tulajdonságainak – szimbólum és/vagy intervallum – megváltoztatásakor.

A három speciális függvényt is lehet parametrizálni. A Terminál csak paraméter megadása nélkül hívhatja őket, ekkor az alapértelmezett értékeket használja a függvény. Bármelyik meghívható viszont a modul tetszőleges részéről, ahogyan a többi függvény is.

[13] Expert Advisor, Custom Indicator vagy Script

Nem ajánlott az init() függvényből a start() függvény meghívása vagy kereskedelmi művelet végrehajtása, mert a chart adatok, a piaci árak még lehet, hogy nem, vagy csak hiányosan állnak rendelkezésre az inicializáció egy adott pillanatában. Az `init()` és `deinit()` függvényeknek olyan gyorsan be kell fejezniük működésüket, amilyen gyorsan csak lehet.

4.1.6. Változók, tömbök, formális paraméterek

Változók deklarálása

A változókat használat előtt deklarálni kell. Elnevezésükre azonosítók használhatók. A deklarációkor használhatók az alaptípusok (`bool, int, double, string`), és a kiegészítő típusok (`datetime, color`).

A típusleírás után a változó nevét kell megadni.

Például:

```
bool igaz;
int egy_szam;
double egy_lebegopontos_szam;
string egy_sztring;
datetime egy_datum = D'2000.01.01 00:00';
color egy_szin = C'128,128,128';
```

Tömbök használata

Egy tömb minden elemének típusa azonos.

```
int egy_dim_tomb[10];         // egeszeket tartalmazo egy dimenzios tomb
int ket_dim_tomb[10][20];     // egeszeket tartalmazo ket dimenzios tomb
```

A tömb index csak egész típusú lehet. Az MQL4 nyelvben a tömb dimenzióinak száma maximálisan négy lehet. A tömb elemeinek sorszámozása 0-val kezdődik, és n elemű tömb esetén (n-1)-ig tart. Ez azt jelenti, hogy a fent deklarált `egy_dim_tomb` utolsó elemére való hivatkozás így valósítható meg:

```
egy_dim_tomb[9]
```

Az indexelés több dimenziós tömbökre is így működik.

Ha olyan indexre történik hivatkozás, amely kívül esik a tömb indextarományán, akkor a futtató alrendszer hibát generál (`ERR_ARRAY_INDEX_OUT_OF_RANGE`).
Az utoljára generált hiba minden esetben lekérdezhető a `GetLastError()` függvénnyel.

Lokális változók

Egy függvény törzsében deklarált változó mindig lokális változója a függvénynek. Láthatósága csak a deklaráló függvényre terjed ki. Bármilyen kifejezéssel inicializálható. Ez az inicializáció a függvény minden lefutásakor végbemegy. A lokális változó a megfelelő függvény memória területén kerül eltárolásra.

Formális paraméterek

Egy függvénynek átadott paraméterek is lokálisak. Hatáskörük a függvény törzse. Egy paraméter neve nem egyezhet meg sem a globális változókéval, sem a függvényben deklarált lokális változókéval. A függvény törzsében egy formális paraméter lokális változóként használható, neki érték adható.
A formális paraméter listán található paraméterek konstansokkal inicializálhatók, ebben az esetben a konstans értéke a paraméter kezdő értéke. Inicializált paraméter után következő paramétereket is inicializálni kell.

```
void teszt_fgv(int a, double b = 1.0, double c = 2.0)
{
 . . .
}
```

Ilyen függvények hívása esetén a kezdőértékkel ellátott paraméterek elhagyhatók. A fenti `teszt_fgv` helyesen meghívható a következő módon:

```
teszt_fgv(1,0.5)
```

Az érték szerinti paraméterátadás miatt a paramétereken – a hívott függvény törzsében – történt változtatásokat a hívó függvény nem fogja érzékelni. Paraméterként tömb is átadható, de elemeinek értéke nem változtatható.

A nyelvben lehetőség van paraméterek referencia szerinti átadására. Az ilyen módon átadott paraméteren történő változtatások a hívó oldalon is megjelennek a megfelelő változóban. Tömbelem nem adható át referenciaként, tömb igen.
Csak ugyanazon modulon belül lévő paraméterek adhatók át referenciaként.
Annak jelölésére, hogy referencia paramétert használ a függvény, a paraméter típusa után & jelet kell rakni.

```
void teszt_fgv2(int& elso, int& masodik, int& tomb[])
{
      elso = 0;
      for(int i = 0; i < ArraySize(tomb) i++)
            elso += tomb[i];
}
```

Referenciaként átadott tömb esetén az elemein végzett változások láthatók a hívó oldalon is.
A fenti függvény lefutásának eredményeként az első paraméter – és a hívó oldalon neki megfelelő változó – értéke a tömb elemeinek összege lesz.
Ellentétben az egyszerű paraméterekkel, tömb átadható referenciaként libraryben lévő függvénynek is.
Referencia paraméternek nem adható kezdőérték.

Statikus változók

Statikus memóriaosztálybeli változó deklarálásakor a típus elé a „static” módosítót kell írni. A statikus változók nem veszítik el értéküket, amikor a deklaráló függvényük befejezi működését, hanem a memóriában maradnak.

```
static int statikus_valtozo = 1;
```

A formális paraméterek kivételével minden változó deklarálható statikusként. Inicializációja egy megfelelő típusú konstanssal történhet. Ha nincs explicit inicializáció, a statikus változó értéke nullázódik

Globális változók

Olyan változó, amely nem valamelyik függvényen belül került deklarálásra. Hatásköre az egész program. Egy globális változó elérhető az összes, a programban definiált függvény által. Ha nincs megadva neki explicit kezdőérték, a változó nullázódik. Kezdőérték csak megfelelő típusú konstanssal adható neki.
Egy globális változó a program futása során csak egyszer, az `init()` függvény meghívása előtt inicializálódik.
A globális hatáskörrel deklarált változók nem keverendők össze a Terminál által használt globális változókkal, amelyek elérésére a `GlobalVariable...()` függvényekkel van lehetőség.

Extern változók

Deklarálásuk a statikus változókéval azonos.

```
extern input_valtozo = 0;
```

Ezek a változók a program input paraméterei, annak „Tulajdonságok” ablakából elérhetők a Terminálon keresztül. Tömb nem lehet extern változó.

Kezdőértékadás

Minden változónak explicit módon adható kezdőérték a definiálásakor. Ennek hiányában a változó nullázódik, azaz értéke beállítódik nullára.
Globális és statikus változónak csak a megfelelő típusú konstanssal adható kezdőérték, ezek a program futása során egyszer inicializálódnak.
Lokális változónak bármilyen kifejezéssel adható kezdőérték. Minden alkalommal inicializálódnak, amikor az őket tartalmazó függvény meghívásra kerül.
Tömb kezdőállapotának beállítására kapcsos zárójelek közötti elemsorozat alkalmazható. Értékek megadásának hiányában a tömb elemei nullázódnak. Ha az inicializálandó tömb mérete nincs megadva, akkor a fordító a megadott elemsorozat hosszát veszi annak. Többdimenziós tömböt ugyanúgy kell inicializálni, ahogy egydimenzióst. Tömb elemeinek csak konstansokkal adható kezdőérték. Például:

```
int tomb[3][3] = {1,2,3, 1,2,3, 1,2,3};
```

Ennek a kétdimenziós, kilencelemű tömbnek a reprezentációja táblázatban:

1	2	3
1	2	3
1	2	3

4.1.7. Az előfeldolgozó rendszer

Az előfeldolgozó (preprocessor) az MQL4 fordítójának egy speciális része. Közvetlenül a forrásszöveg lefordítása előtt dolgozik.

Az előfeldolgozó lehetőséget biztosít a forrásszöveg olvashatóságának növelésére, például mnemonikok[14] elhelyezését teszi lehetővé a kódban konstansok használata helyett.

Ha a # jel szerepel egy program egy sorának első (nem whitespace) karaktereként, akkor az a sor az előfordítónak szóló **direktíva**. A direktíva sorának lezárása: soremeléssel.

Nevesített konstans deklarálása

Nevesített konstansokat a `#define` szerkezet használatával lehet létrehozni. A program szövegében a nevesített konstans mindig a nevével jelenik meg. A fordítás előtt az előfeldolgozó ennek a névnek – a program szövegében elhelyezett – az összes előfordulását helyettesíti az értékével. Az érték bármilyen típusú lehet. A név azonosító jellegű, általában nagybetűkkel írva. Használata:

```
#define azonosító érték
```

Például:

```
#define PI 3.141592          // a PI konstans erteke 3.141592 lesz
#define IRO_NEVE ”Hugo” // a NEV konstans erteke a ”Hugo” sztring
```

Propertyk

Minden MQL4 programban lehetőség nyílik további speciális jellemzők megadására a forrásszövegben. Ezek a `#property` paraméterek segítenek a Terminálnak a programok jobb kiszolgálásban. A fordító a `#property` paramétereket a futtatandó modul tulajdonságai közé illeszti. Elsősorban indikátorok külső beállításainak megadásánál alkalmazandók.

```
#property azonosító érték
```

Például:

```
#property copyright ”Kiss Istvan”
```

A Függelék tartalmaz egy táblázatot a propertyk lehetséges értékeiről.

[14] Emlékeztető, „beszédes név”

Fájlok inkludálása

Az `#include` direktíva bárhol elhelyezhető a program szövegében, de általában a forrásszöveg elején található.

Alkalmazása:

```
#include <allomany_nev>
#include ”allomany_nev”;
```

Az előfordító ezt a sort helyettesíti az allomany_nev-vel megadott fájl tartalmával.
Az idézőjelek között levő állományt az alapértelmezett könyvtárban keresi, az aktuális könyvtárban nem. A másik megadási formánál a keresés fordítva megy végbe.

Függvények importálása

Lefordított MQL4 modulokból (.ex4) vagy .dll modulokból lehet függvényeket importálni. A modul nevét `#import` direktívában kell megadni. Ahhoz, hogy a fordító helyesen tudja megvalósítani az importált függvény hívását és a paramétereinek átadását, a függvény teljes meghatározására szükség van importáláskor. Az importált függvény leírását közvetlenül az import direktíva után kell megadni.

```
#import ”allomany_nev”
      fgv1_def;
      fgv2_def;
      .
      .
      .
      fgvn_def;
#import
```

Az importált függvényeknek egyedi névvel kell rendelkezniük, azonos nevű függvények nem importálhatók egyszerre különböző modulokból, és nevük nem egyezhet meg beépített függvényekével sem
Ahogy egy függvény kikerül abból a környezetből, ahol lefordításra került (azaz importálja egy másik modul), a fordító nem tudja ellenőrizni az átadott paraméterek helyességét. Ezért – a futási hibák elkerülése miatt – szükség van a paraméterek típusának és sorrendjének helyes

megadására importáláskor. Az importált függvényeknek nem lehet kezdőértékkel ellátott paramétereik.

Egy program futása során, az importált függvények esetében a nyelv késői kötést alkalmaz, ami azt jelenti, hogy a meghívott modul addig nem töltődik be, amíg az importált függvény meghívásra nem kerül.

Nem ajánlott a betöltendő modul teljes elérési útját megadni.

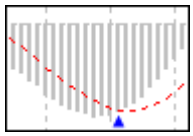
Az MQL4 libraryk alapértelmezetten a terminal_eleresi_utja\experts\libraries könyvtárból kerülnek betöltésre. Ha a modul ott nem elérhető, akkor a rendszer egy szinttel fentebb próbálkozik a könyvtár-hierarchiában.

4.2. A nyelv további elemeinek bemutatása programfejlesztéssel

Ez a fejezet a nyelv további, fontos elemeinek valamint egy program fejlesztési menetének bemutatására került a dolgozatba. A bemutatás nem lesz teljes körű, csak azokat a nyelvi elemeket érinti, amelyek vizsgálhatók egy meghatározott expert segítségével. Ez az expert (*MACD Sample.mq4*) példaprogramként megtalálható a MetaEditorban. A dolgozatban lévő bemutatás alapját egy – az expertről készült – cikk képezi [8].

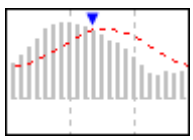
Az expert működésének alapját az MACD indikátor adja. Az indikátor vezérlő jeleket generál pozíció nyitására a következő módon:

- **Vételi jel (long pozíció)** – ha az indikátor értéke negatív, emelkedik és keresztezi a jelvonalat, ahogy a 10. ábra szemlélteti.



10**. ábra** *Vételi jel*

- **Eladási jel (short pozíció)** – ha az indikátor értéke pozitív, csökken és keresztezi a jelvonalat, ahogy a 11. ábra szemlélteti.

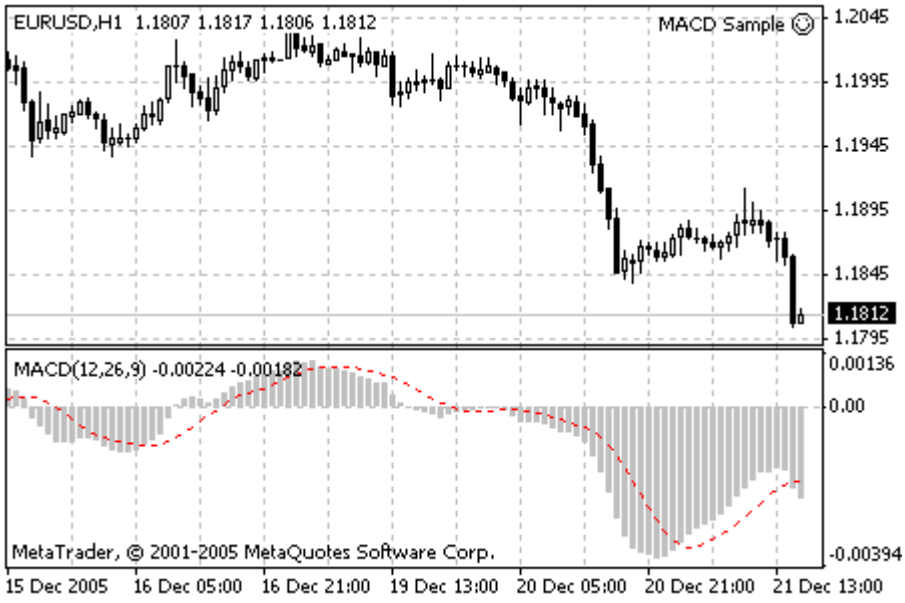


11**. ábra** *Eladási jel*

A pozíciókból való kiszállás a következő módon zajlik:

- **Kiszállás long pozícióból**– a take profit vagy a trailing stop[15] szint elérésekor vagy eladási jel esetén
- **Kiszállás short pozícióból** – a take profit vagy a trailing stop szint elérésekor vagy vételi jel esetén

Fontos, hogy az indikátor jelentéktelen változásait ne vegye figyelembe a program. Ezért az indikátor használata kiegészítésre kerül úgy, hogy az általa generált jelet csak akkor használja a program, ha az indikátor mérete nagyobb egy bizonyos értéknél. Az MACD indikátor jelzéseit egy adott időszakra a 12. ábra mutatja.



12. ábra *MACD indikátor jelzései egy adott árfolyamgörbére (EURUSD, H1)*

4.2.1. A programfejlesztés lépései és a forrásszöveg magyarázata

Ebbe az alfejezetbe a fentebb már említett *MACD Sample.mq4* expert teljes forrásszövege bekerül. A kódon kívül csak a szervesen hozzá tartozó magyarázatok szerepelnek a szövegben. A magyarázatok elhagyásával megkapjuk a teljes, helyes forráskódot.

[15] Követő stop, egy olyan kereskedési stratégia, amelynél a kiszállási árfolyam meghatározott módon (valahány pont távolsággal) követi az árfolyamot.

Változók inicializálása

A program elején célszerű megadni a külső változókat. Ezeket később a programszöveg átírása nélkül is meg lehet változtatni a Terminálon keresztül.

```
extern double TakeProfit = 120;
extern double Lots = 4;
extern double TrailingStop = 30;
extern double MACDOpenLevel=3;
extern double MACDCloseLevel=2;
extern double MATrendPeriod=26;

int start()
  {
   double MacdCurrent, MacdPrevious, SignalCurrent;
   double SignalPrevious, MaCurrent, MaPrevious;
   int cnt, ticket, total;
```

Kiinduló adatok ellenőrzése

Ez a programrész általában – kis módosításokkal – az összes expertben előfordul. Ellenőrzi, hogy az aktuális chart ablakban mennyi a rendelkezésre álló gyertyák száma (`Bars`), valamint azt, hogy az extern változók (jelen esetben a TakeProfit változó) beállítása megfelelő-e a kezdeti elvárásokhoz.

```
   if(Bars<100)
     {
      Print("bars less than 100");
      return(0);
     }
   if(TakeProfit<10)
     {
      Print("TakeProfit less than 10");
      return(0);
     }
```

Változók beállítása a gyors adathozzáféréshez

Az indikátorok (`iMA, iMACD`) által kalkulált értékeket célszerű változókban tárolni, mert így könnyebben, gyorsabban hozzáférhetővé válnak. Az indikátorok hívásában nevesített konstansok szerepelnek (`PRICE_CLOSE, MODE_MAIN` stb.).

```
MacdCurrent=iMACD(NULL,0,12,26,9,PRICE_CLOSE,MODE_MAIN,0);
MacdPrevious=iMACD(NULL,0,12,26,9,PRICE_CLOSE,MODE_MAIN,1);
SignalCurrent=iMACD(NULL,0,12,26,9,PRICE_CLOSE,MODE_SIGNAL,0);
SignalPrevious=iMACD(NULL,0,12,26,9,PRICE_CLOSE,MODE_SIGNAL,1);
MaCurrent=iMA(NULL,0,MATrendPeriod,0,MODE_EMA,PRICE_CLOSE,0);
MaPrevious=iMA(NULL,0,MATrendPeriod,0,MODE_EMA,PRICE_CLOSE,1);
```

A terminál vizsgálata

A terminálban nyitott pozíciók számát egyszerűen meg lehet határozni az `OrdersTotal()` függvénnyel:

```
total=OrdersTotal();
if(total<1)
  {
```

Ha nincs nyitott pozíció, akkor a piaci helyzet elemzése előtt célszerű megvizsgálni az account tulajdonságait, például anyagi hátterét az `AccountFreeMargin()` függvénnyel. A függvény visszatérési értéke az account szabad fedezete.

```
if(AccountFreeMargin()<(1000*Lots))
  {
   Print("We have no money. Free Margin = ", AccountFreeMargin());
   return(0);
  }
```

Az anyagi feltételek ellenőrzése után meg lehet vizsgálni, van-e lehetőség pozíció nyitására. Vételi pozíció nyitásának feltétele (vételi jel):

```
if(MacdCurrent<0 && MacdCurrent>SignalCurrent &&
```

```
MacdPrevious<SignalPrevious &&
MathAbs(MacdCurrent)>(MACDOpenLevel*Point) &&
MaCurrent>MaPrevious)
  {
```

Vételi jel esetén pozíció nyitás következik az `OrderSend()` függvénnyel. A függvény visszatérési értéke a pozíció sorszáma lesz, amelyet értékül adunk a `ticket` változónak:

```
ticket=OrderSend(Symbol(),OP_BUY,Lots,Ask,3,0,Ask+
TakeProfit*Point,"macd sample",16384,0,Green);
```

Ha sikeres volt az `OrderSend()` függvény, akkor a `ticket` értéke nem 0. További használatra kiválasztjuk a `ticket` sorszámú megbízást az `OrderSelect()` függvénnyel

```
if(ticket>0)
  {
   if(OrderSelect(ticket,SELECT_BY_TICKET,MODE_TRADES))
     Print("BUY order opened : ",OrderOpenPrice());
  }
```

Ha nem volt sikeres az `OrderSend()` függvény, akkor a `ticket` értéke 0, ekkor a hiba kiíratásra kerül a `GetLastError()` függvény segítségével.

```
else
   Print("Error opening BUY order : ",GetLastError());
return(0);
}
```

Az eladási pozíció nyitása analóg módon történik a vételi pozíció nyitásával:

```
if(MacdCurrent>0 && MacdCurrent<SignalCurrent &&
MacdPrevious>SignalPrevious &&
MacdCurrent>(MACDOpenLevel*Point) &&
MaCurrent<MaPrevious)
  {
   ticket=OrderSend(Symbol(),OP_SELL,Lots,Bid,3,0,Bid-
```

```
         TakeProfit*Point,"macd sample",16384,0,Red);
         if(ticket>0)
           {
            if(OrderSelect(ticket,SELECT_BY_TICKET,MODE_TRADES))
               Print("SELL order opened : ",OrderOpenPrice());
           }
         else
            Print("Error opening SELL order : ",GetLastError());
         return(0);
        }
```

Pozíció nyitása után az expert befejezi működését, a következő beérkező adatra pedig újra kezdi.

```
      return(0);
     }
```

Nyitott pozíciók kezelése

A `cnt` a ciklusváltozó, amit a ciklusba lépés előtt deklarálni kell. A ciklus végiglépked a nyitott pozíciókon.

```
   for(cnt=0;cnt<total;cnt++)
     {
      OrderSelect(cnt, SELECT_BY_POS, MODE_TRADES);
      if(OrderType()<=OP_SELL && OrderSymbol()==Symbol())
        {
```

Vételi pozíció esetén megvizsgálja, hogy szükség van-e a pozíció zárására. Ha igen, akkor az `OrderClose()` függvénnyel megteszi azt az aktuális áron (`Bid`).

```
         if(OrderType()==OP_BUY)
           {
            if(MacdCurrent>0 && MacdCurrent<SignalCurrent &&
            MacdPrevious>SignalPrevious &&
            MacdCurrent>(MACDCloseLevel*Point))
               {
                OrderClose(OrderTicket(),OrderLots(),Bid,3,Violet);
```

```
        return(0);
       }
```

Ha a pozíció nem került lezárásra, akkor az expert megvizsgálja annak trailing stop szintjét, ami csak akkor állítódik át, ha a pozíció már ért el profitot. Az `OrderModify()` függvény hívásával lehet megvalósítani a követő stopot, hiszen meghívásakor a stop loss értéket mindig az aktuális árfolyamhoz igazítja (`Bid-Point*TrailingStop`).

```
    if(TrailingStop>0)
      {
       if(Bid-OrderOpenPrice()>Point*TrailingStop)
         {
          if(OrderStopLoss()<Bid-Point*TrailingStop)
            {
             OrderModify(OrderTicket(),OrderOpenPrice(),Bid-
             Point*TrailingStop,OrderTakeProfit(),0,Green);
             return(0);
            }
         }
      }
   }
```

Nyitott eladási pozíciók vizsgálata és a kezelés analóg a vételi pozíciókéhoz:

```
else
  {
   if(MacdCurrent<0 && MacdCurrent>SignalCurrent &&
   MacdPrevious<SignalPrevious &&
   MathAbs(MacdCurrent)>(MACDCloseLevel*Point))
     {
      OrderClose(OrderTicket(),OrderLots(),Ask,3,Violet);
      return(0);
     }
   if(TrailingStop>0)
     {
      if((OrderOpenPrice()-Ask)>(Point*TrailingStop))
        {
```

```
                if((OrderStopLoss()>(Ask+Point*TrailingStop)) ||
               (OrderStopLoss()==0))
                  {
                   OrderModify(OrderTicket(),OrderOpenPrice(),Ask+
                   Point*TrailingStop,OrderTakeProfit(),0,Red);
                   return(0);
                  }
               }
            }
```

Már csak a nyitott zárójelek bezárása és az expert befejeztetése maradt hátra:

```
            }
         }
      }
   return(0);
  }
```

Az ily módon elkészített expert futtatható a Terminálban.

Természetesen a nyelv adta lehetőségek sokkal bővebbek az eddig leírtaknál. Nevesített konstansok, beépített függvények és előre elkészített indikátorok sora segíti az automatizált kereskedés programozását és a devizapiaci döntéshozatalt.

5. Összefoglalás

A dolgozatban leírt eszközrendszer megfelel az automatizált kereskedés programozására a devizapiacon. Megmutattam, hogy a stratégiai programozás és a programok tesztelési, optimalizálási lehetősége hogyan ültethető át a gyakorlatba az MQL4 programozási nyelv és a MetaTrader 4 platform segítségével. A dolgozatot nem a nyelv vagy a platform teljes leírásának szántam, azoknak csak azt a részét mutattam be, amely lehetővé teszi az online kereskedést automatizált módon.

A dolgozat utolsó részében bemutatott program nem túl bonyolult, a kereskedési stratégiának teljesen megfelel, a gyakorlatban is jól használható.

Irodalomjegyzék

1. Érsek Zs. 2002. *Bevezetés a devizapiacokra*. KJK-KERSZÖV Jogi és Üzleti Kiadó Kft.

2. Gyulaffy Béláné Dr. 2005. *Devizagazdálkodás – devizaügyletek.* Dunaújvárosi Főiskola Kiadói Hivatal.

3. Gyulaffyné Dr. Berényi M., Kaszás M. 1994. *Tőzsdeelemzés.* SALDO Pénzügyi Tanácsadó és Informatikai RT.

4. Juhász I. 2003. *Programozás 1.* Egyetemi jegyzet, elektronikus közlés. mobiDIÁK könyvtár, Debrecen.

5. Kernighan, B. W., Ritchie, D. M. 2000. *A C programozási nyelv*. Műszaki Könyvkiadó.

6. *Devizapiaci alapismeretek*: http://devizapont.hu/tananyag.php

7. *Devizapiaci alapismeretek*: http://www.infinad.hu/download/ALAPISMERETEK.pdf

8. *Expert Advisor Sample*: http://articles.mql4.com/84

9. *MetaQuotes Language 4 – Expert Advisors*: http://www.metaquotes.net/experts/mql4/

10. *MetaQuotes Language 4 Documentation*: http://docs.mql4.com/

11. *MetaTrader 4 Trading Platform*: http://metatrader4.com

12. *MetaTrader 4 User's Guide*: http://www.metaquotes.net/files/metatrader4_en.chm

Függelék

1. Táblázat *Az MQL4 Precedenciatáblázata*

()	Függvényhívás	Balról jobbra
[]	Tömbelem hivatkozás	
!	Logikai negálás	Jobbról balra
-	Előjelváltás	
++	Inkrementálás	
--	Dekrementálás	
~	Bitenkénti negálás	
&	Bitenkénti ÉS	Balról jobbra
|	Bitenkénti VAGY	
^	Bitenkénti kizáró VAGY	
<<	Balra shiftelés	
>>	Jobbra shiftelés	
*	Szorzás	Balról jobbra
/	Osztás	
%	Maradékos osztás	
+	Összeadás	Balról jobbra
-	Kivonás	
<	Kisebb	Balról jobbra
<=	Kisebb egyenlő	
>	Nagyobb	
>=	Nagyobb egyenlő	
==	Egyenlő	
!=	Nem egyenlő	
||	Logikai VAGY	Balról jobbra
&&	Logikai ÉS	Balról jobbra
=	Értékadás	Jobbról balra
+=	Értékadás	
-=	Értékadás	
*=	Értékadás	
/=	Értékadás	
%=	Értékadás	
>>=	Értékadás	
<<=	Értékadás	
&=	Értékadás	
|=	Értékadás	
^=	Értékadás	
,	Vessző	Balról jobbra

2. Táblázat *Propertyk és lehetséges értékeik*

Constant	Type	Description
link	string	a link to the company website
copyright	string	the company name
stacksize	int	stack size
library		a library; no start function is assigned, non-referenced functions are not removed
indicator_chart_window	void	show the indicator in the chart window
indicator_separate_window	void	show the indicator in a separate window
indicator_buffers	int	the number of buffers for calculation, up to 8
indicator_minimum	double	the bottom scaling limit for a separate indicator window
indicator_maximum	double	the top scaling limit for a separate indicator window
indicator_colorN	color	the color for displaying line N, where N lies between 1 and 8
indicator_widthN	int	width of the line N, where N lies between 1 and 8
indicator_styleN	int	style of the line N, where N lies between 1 and 8
indicator_levelN	double	predefined level N for separate window custom indicator, where N lies between 1 and 8
indicator_levelcolor	color	level line color
indicator_levelwidth	int	level line width
indicator_levelstyle	int	level line style
show_confirm	void	before script run message box with confirmation appears
show_inputs	void	before script run its property sheet appears; disables show_confirm property