

Programování obchodních strategií pomocí MetaQuotes Language 4

MetaQuotes Language 4 for programming of trading strategies

Petr Kostelanský

Bakalářská práce
2011



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

akademický rok: 2010/2011

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: Petr KOSTELANSKÝ

Osobní číslo: A08056

Studijní program: B 3902 Inženýrská informatika

Studijní obor: Informační a řídicí technologie

Téma práce: Programování obchodních strategií pomocí
MetaQuotes Language 4

Zásady pro vypracování:

1. Seznamte se s obchodní platformou MetaTrader 4.
2. Popište prostředí MetaEditoru.
3. Vytvořte základní manuál k jazyku MetaQuotes Language 4.
4. Vytvořte v tomto prostředí následující typy programů: Script, Custom Indicator, Expert Advisor.

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. MQL4 and Automated Trading in MetaTrader 4 / MetaQuotes Software Corp. [online]. c2000–2011 [cit. 2011-01-27]. Automated Trading. Dostupné z WWW: [http://www.metaquotes.net/en/metatrader4/automated_trading/].
2. MQL4 – automated forex trading, custom indicators and strategy tester [online]. c2000–2011 [cit. 2011-01-27]. Documentation. Dostupné z WWW: [http://docs.mql4.com/].
3. MetaTrader [Financnik] [online]. 26.09.2007 [cit. 2011-01-27]. MetaTrader. Dostupné z WWW: [http://www.financnik.cz/wiki/metatrader?s=mql4].
4. ZABIELSKI, Michaz. Úvodní lekce do MQL : Průvodce automatickým obchodováním. X-TRADE BROKERS S.A.[online]. 2010-04-12, 1, [cit. 2011-01-27]. Dostupný z WWW: [http://www.xtb.cz/repository/Downloads/MQL_Newbie_Book_cz.pdf].
5. X-Trade Brokers. XTB-Trader [počítačový program]. Ver. 4.00 Build 229. [Česka republika], 2010 [cit. 2011-01-27]. Dostupné na [http://www.xtb.cz/obchodni_systemy/xtb_trader/navod_k_obsluze/].

Vedoucí bakalářské práce:

Ing. Radek Matušů, Ph.D.

Ústav automatizace a řídicí techniky

Datum zadání bakalářské práce:

25. února 2011

Termín odevzdání bakalářské práce:

7. června 2011

Ve Zlině dne 25. února 2011

prof. Ing. Vladimír Vašek, CSc.
děkan



prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

ABSTRAKT

V bakalářské práci se seznámíme s jednou z nejrozšířenějších platforem pro online obchodování. Ukážeme si jaké typy programů si můžeme vytvořit. Tyto programy by měly obchodníkovi pomoci určovat vývoj trhu nebo zcela automatizovat obchodování.

Bakalářská práce se skládá ze dvou částí. Teoretická obsahuje informace o obchodní platformě a manuál, který obsahuje základní pravidla a informace o jazyku používající se v této platformě. Praktickou část představují tři jednoduché programy, které mají v obchodní platformě rozdílný význam. Programy jsem v práci rozebral a vysvětlil jejich funkcionalitu.

Klíčová slova: MetaTrader 4, MQL4, Expert Advisor, Custom Indicator, online obchodování

ABSTRACT

In this Bachelor thesis we familiarize ourselves with one of the most spread platforms for online business. We illustrate what types of programs we can create. These programs should help the trader to determine an evolution of the market or entirely automate the trading.

This Bachelor thesis consists from two parts. The theoretical part contains information about a trading platform and a manual containing basic rules and information about a language used in this platform. The practical part is presented by three simple programs with a different meaning in the trading platform. I have analysed these programs in my thesis and explained their functionality.

Keywords: MetaTrader 4, MQL4, Expert Advisor, Custom Indicator, online trading

Tímto bych chtěl poděkovat vedoucímu mé bakalářské práce panu Ing. Radku Matušovi Ph.D. za odborné vedení a podnětné připomínky udílené během vypracovávání této práce.

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- § že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- § že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....
podpis diplomanta

OBSAH

ÚVOD	9
I TEORETICKÁ ČÁST	10
1 ONLINE OBCHODOVÁNÍ	11
1.1 CFD	12
1.2 FUTURES.....	12
1.3 FOREX	13
2 OBCHODNÍ PLATFORMA METATRADER 4.....	14
2.1 KLIENTSKÝ TERMINÁL.....	15
2.2 METAQUOTES LANGUAGE 4.....	16
2.2.1 Expert Advisors	16
2.2.2 Custom Indicators.....	16
2.2.3 Scripts	16
2.2.4 Libraries	16
3 PROSTŘEDÍ METAEDITORU	17
4 ZÁKLADNÍ MANUÁL K METAQUOTES LANGUAGE 4.....	20
4.1 SYNTAXE.....	20
4.1.1 Komentáře.....	20
4.1.2 Identifikátory	20
4.1.3 Vyhrazená slova.....	21
4.2 DATOVÉ TYPY	21
4.2.1 Integer (int)	22
4.2.2 Boolean (bool).....	22
4.2.3 Literal (char).....	22
4.2.4 Strings (string).....	22
4.2.5 Floating-point number (double).....	23
4.2.6 Color (color)	23
4.2.7 Date and time (datetime).....	23
4.3 OPERACE A VÝRAZY	24
4.3.1 Aritmetické operace.....	24
4.3.2 Operace přiřazení.....	24
4.3.3 Porovnávací operace.....	25
4.3.4 Boolean operace	25
4.3.5 Bitové operace.....	25
4.4 OPERÁTORY.....	26
4.4.1 Operátor if.....	26
4.4.2 Operátor if-else.....	26
4.4.3 Operátor switch	26
4.4.4 Operátor while.....	27
4.4.5 Operátor for	27

4.5	FUNKCE.....	27
4.5.1	Definice funkcí.....	27
4.5.2	Volání funkce	28
4.5.3	Speciální funkce.....	28
II	PRAKTICKÁ ČÁST.....	30
5	INSTALACE METATRADER 4	31
6	POUŽITÍ PROGRAMŮ	34
6.1	POUŽITÍ INDIKÁTORŮ A SKRIPTŮ.....	34
6.2	POUŽITÍ A NASTAVENÍ STRATEGIÍ.....	36
6.3	TESTER STRATEGIE.....	39
6.3.1	Optimalizace strategie.....	42
7	CUSTOM INDICATOR	45
7.1	DEFINICE PARAMETRŮ PREPROCESORU A EXTERNÍCH PROMĚNNÝCH.....	45
7.2	FUNKCE INIT().....	46
7.3	FUNKCE START()	50
7.4	FUNKCE PRO VÝPOČET KLOUZAVÝCH PRŮMĚRŮ.....	52
8	EXPERT ADVISOR	55
8.1	DEFINICE PARAMETRŮ PREPROCESORU A EXTERNÍCH PROMĚNNÝCH.....	55
8.2	FUNKCE INIT().....	56
8.3	FUNKCE START()	58
8.4	VLASTNÍ FUNKCE	64
9	SCRIPT	70
9.1	DEFINICE PARAMETRŮ PREPROCESORU A FUNKCÍ KNIHOVNY	70
9.2	FUNKCE START()	71
	ZÁVĚR.....	75
	ZÁVĚR V ANGLIČTINĚ.....	76
	SEZNAM POUŽITÉ LITERATURY.....	77
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	78
	SEZNAM OBRÁZKŮ.....	79
	SEZNAM TABULEK	81
	SEZNAM PŘÍLOH.....	82

ÚVOD

S rozvojem počítačové techniky, ale především Internetu, se rozšířilo i online obchodování umožňující přístup na trh běžným lidem. Je potřeba pouze počítač, přístup k Internetu a účet u společnosti, která nám umožní plnění obchodních příkazů zadávaných prostřednictvím softwaru, poskytovaného těmito společnostmi.

Zprostředkovatelských společností je velká řada a je jen na člověku jakou si vybere. Při výběru je třeba zvážit výši poplatků, které se tyto firmy za svoje služby účtují, ale také software poskytnutý touto společností. Velmi rozšířenou platformou je MetaTrader 4. Hlavní výhodou tohoto softwaru je možnost naprogramovat si potřebné ukazatele pro analýzu trhu. Tento software nabízí i další možnosti a jednou z velmi zajímavých je možnost vytvoření vlastních automatických obchodních systémů, které vyhodnocovací proces trhu automatizují a není tak potřeba fyzické přítomnosti obchodníka.

V práci se budeme zabývat použitím jazyka MQL4 a tvorbou užitečných programů pro zmíněnou platformu.

I. TEORETICKÁ ČÁST

1 ONLINE OBCHODOVÁNÍ

Online obchodování (online trading) patří mezi zajímavé a ziskové příležitosti podnikání. Dřív než se člověk tomuto „řemeslu“ začne věnovat, je třeba získat potřebné vědomosti a zkušenosti. Tento způsob podnikání může přinést velké zisky, ale také velké ztráty, protože je velmi rizikový.

V dřívějších dobách se jednalo o parketové obchodování. Obchodovalo se přímo v několikapatrových budovách burzy, kde se nabídky a poptávky párovaly ručně. Do doby elektronického obchodování bylo parketové jediným typem obchodování, avšak jeho popularita postupně klesá a nahrazuje ho online obchodování, které je výhodnější. [1]



Obr. 1: Obrázek z burzy CBOT [2]

Některé výhody online obchodování:

Tab. 1: Rozdíly online a parketového obchodování

	Online obchodování	Parketové obchodování
Přímý přístup na trh	Ano	Ne
Komise (poplatek brokerovi)	Nízká	Vysoká
Okamžitá data z jakéhokoliv trhu	Ano	Ne
Využití pokročilých obchodních strategií	Ano	Ne

Obchodování na burze zprostředkovává tzv. broker, který za nás nakupuje a prodává kontrakty. Dříve se příkazy podávaly např. telefonicky a fyzická osoba (zastupující brokerskou společnost) za nás na burze nakoupila. [1]

Při online obchodování je to velmi podobné, rozdíl je v tom, že s naším brokerem komunikujeme pomocí online platform, které broker poskytuje. Vše se dnes děje elektronicky a to od podání pokynu k nákupu, až po samotný nákup kontraktu na burze. [1]

Online obchodování na burze nám umožňuje nakupování a prodávání vybraných aktiv (akcie, komodity, měnu), aniž bychom je fyzicky vlastnili. To nám umožňuje finanční a komoditní derivát Contract For Difference (CFD). Mezi další finanční deriváty patří např. Futures, Opce a další. [1]

1.1 CFD

CFD neboli kontrakty na vyrovnání rozdílů. Hlavní výhodou kontraktů CFD je, že investoři mohou obchodovat s měnami, komoditami, indexy a akciami, aniž by je skutečně vlastnili. Např. můžeme koupit kontrakt pro rozdíl na 100 akcií GOOG místo toho, abychom koupili 100 akcií GOOG na burze. Naším cílem je nakoupit CFD kontrakt a v budoucnu ho prodat za vyšší cenu, nebo nejprve prodat za vyšší cenu a v budoucnu nakoupit za cenu nižší. [1], [3]

Při otevření obchodu nám bude zablokována na našem účtu tzv. marže, která nám bude vrácena při uzavření pozice. Pokud byl náš obchod ziskový, budou k této částce přičteny další peníze. V opačném případě bude od marže naše ztráta odečtena a vrátí se nám rozdíl. Velikost marže se odvíjí od velikosti uzavřeného kontraktu (např. 3% hodnoty otevřené pozice, každý broker má jinou marži na různé instrumenty). [3], [7]

1.2 Futures

Futures představují smlouvu mezi dvěma stranami, které se zavazují ke koupi (prodeji) podkladového aktiva v den splatnosti. Místo dodání podkladového aktiva dochází většinou k peněžnímu vyrovnání. [4]

Futures mohou být kontrakty na dodávku akcii, měn, dluhopisů, kovů (zlato, stříbro), energii a dalších komodit. V čem se liší Futures od CFDs? Liší se především v tom, že

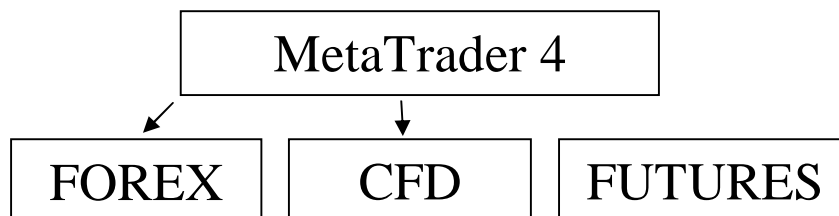
futures mají splatnost kontraktu (jedná se o expiraci, kdy dochází k vypořádání kontraktu mezi smluvními stranami), kdežto CFDs se uzavírají na dobu neurčitou. [4]

1.3 Forex

Foreign Exchange (Forex nebo FX) je největší finanční trh světa pro směnu měnových párů, který nemá vlastní burzu. Jedná se o mezinárodní obchodní systém, který propojuje banky, pojišťovny, brokerské společnosti (spojující individuální investory) a další. Na forexu se obchoduje 24 hodin denně 5 dní v týdnu. [5]

2 OBCHODNÍ PLATFORMA METATRADER 4

Jedná se o jednu z nejrozšířenějších obchodních online platforem, poskytující zprostředkovatelské služby na Forexových, CDFs a Futures trzích. [6]









Obr. 2: Makléřské služby poskytované platformou MetaTrader 4, [6]

MetaTrader 4 (MT4) je uživatelsky příjemný, nabízí živé grafy, poskytuje uživateli celou řadu nástrojů technické analýzy a široké spektrum technických indikátorů (ukazatelů).

Mezi konkurenční výhody MT4 patří pohodlné mobilní obchodování, inovativní automatické obchodování, robustní zabezpečení a funkce okamžitých a čekajících příkazů pro flexibilní management obchodů. Uživatel může také využít jazyk MetaQuotes Language 4 (MQL4) pro vytvoření vlastních skriptů, indikátorů, ale také k vytvoření programů zajišťujících automatické obchodování. [6]

Tab. 2: Tabulka některých Forex brokerů [9]

Logo	Název	Platforma	Země
	Alpari UK	MT4	Británie
	Brokerjet	SAXO TRADER	ČR
	Colosseum	MT4	ČR
	Interactive Brokers	TWS	USA
	Patria Forex	FXCM	ČR
	X-Trade Brokers	MT4	ČR

Brokerů využívající platformu MT4 je v dnešní době na 110.

2.1 Klientský terminál

Klientský terminál MT4 nabízí řadu analytických nástrojů. Pro každý obchodovaný instrument poskytuje devět časových rámců (time-frames). Je zde několik základních indikátorů a dalších více než 50 můžeme do platformy stáhnout z internetu, nebo si můžeme vytvořit vlastní. Tyto indikátory slouží ke zjednodušení analýzy trhu a určení trendu.

K určení trendu slouží fundamentální analýza a technická analýza.

- Fundamentální analýza (FA) – zabývá se zkoumáním vlivu celého trhu a je nutné se dokonale seznámit s obchodovaným instrumentem, např. vliv počasí na velikost sklizně u zemědělských komodit ovlivňuje velikost nabídky, nebo vliv úrokových měr jednotlivých států na cykličnost měn v období hospodářského růstu nebo krize. [1]
- Technická analýza (TA) – slouží k předpovídání budoucího vývoje trhu vyhodnocováním a analyzováním minulých a současných dat (ceny instrumentů, objemy) pomocí statistiky, grafů a indikátorů. TA přepokládá, že tržní cena odráží chování účastníků trhu a nebude se v krátkém období měnit, proto se využívá především pro krátkodobé obchody. [1]

V této práci s budeme zabývat tvorbou indikátorů pro TA.



Obr. 3: Klientský terminál

2.2 MetaQuotes Language 4

MQL4 je zabudovaný programovací jazyk pro tvorbu skriptů (Script), vlastních indikátorů (Custom Indicator), knihoven (Libraries) a programů pro automatické obchodování (Expert Advisor) v platformě MT4. Tím se výrazně rozšiřují možnosti této obchodní platformy. Jazyk MQL4 je syntaxí velmi podobný jazyku C. Ke tvorbě programů se používá prostředí MetaEditor 4. [8]

2.2.1 Expert Advisors

Expert Advisor (EA) je program, který umí automaticky provádět obchodní příkazy a podle námi vytvořené strategie (s využitím TA) sleduje vývoj na trhu. V okamžiku, kdy na základě strategie přijde od programu signál k provedení obchodu, se obchod automaticky uzavře. Je možné nastavit, aby nás pouze informoval o vstupu na trh, který program vyhodnotil. Pak se nám otevře okno, ve kterém stačí kliknout na tlačítko *koupit* nebo *prodat* a obchod se uzavře. Nemusíme už nastavovat stoploss (SL) ani takeprofit (TP), vše je už vyplněno na základě toho, jak je obchodní strategie naprogramovaná a nastavená. [8]

2.2.2 Custom Indicators

Custom Indicators (CI) umožňují vytvořit vlastní technické indikátory, které můžeme používat společně s již integrovanými technickými indikátory. Tyto programy nemůžou provádět obchody a slouží výhradně k analytickým účelům. Pokud je indikátor přidán na graf, tak se spouští v okamžiku každého cenového pohybu, tím reaguje na aktuální změnu, která se promítne v grafu indikátoru. [8]

2.2.3 Scripts

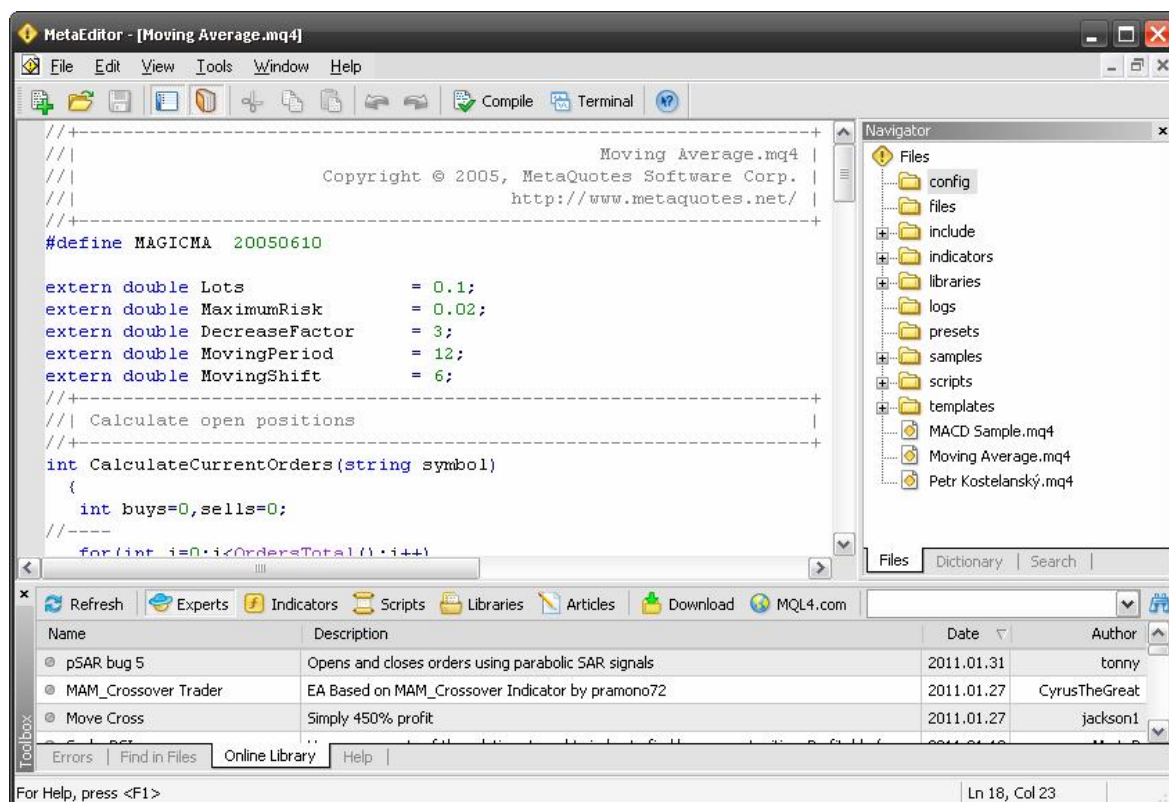
Používají se pro jednorázové provedení naprogramovaných úkonů. Spustí se pouze jednou a znovu se samy nespouští, tím se liší od CI a EA. Pomocí skriptu si můžeme zobrazovat informace o obchodovaných instrumentech jako např. spread, swap, nebo uzavírat několik obchodů najednou. Podstatou tedy je, že se nespouští opakovaně. [8]

2.2.4 Libraries

Knihovny, ve kterých jsou ukládány části uživatelských programů, využívané v jiných programech. [8]

3 PROSTŘEDÍ METAEDITORU

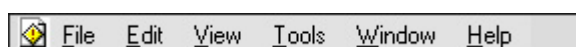
MetaEditor 4 IDE (Integrated Development Environment) je prostředí, které je součástí klientského terminálu a napomáhá a zjednodušuje orientaci ve zdrojovém kódu. Slouží k vytvoření, upravení nebo kompilaci vytvořeného programového kódu napsaném v jazyku MQL4. V MetaEditoru můžeme vytvořit Script, CI nebo EA, které pak můžeme použít v klientském terminálu. [8]



Obr. 4: Prostředí MetaEditoru

§ Panel nabídek

Nachází se v horní části obrazovky, je tvořen nabídkami (File, Edit, View, Tools, Window, Help), které fungují jako tzv. roletová nabídka s dalšími nabídkami možností. V roletové nabídce *File* je nejvýznamnější příkaz *Compile*, před každým použitím našeho programu je třeba ho nejdříve zkompileovat a poté je možné ho spustit v klientském terminálu.



Obr. 5: Prostředí MetaEditoru, panel nabídek

§ Panel nástrojů

Panel nástrojů je možné libovolně upravovat a přidat si na něj námi nepoužívanější nástroje, které MetaEditor nabízí.

Vlastní nástroje přidáme ve *View->Customiz...* a po otevření okna *Customizing toolbar* vybereme nástroje, které chceme aby se zobrazovaly v panelu nástrojů.

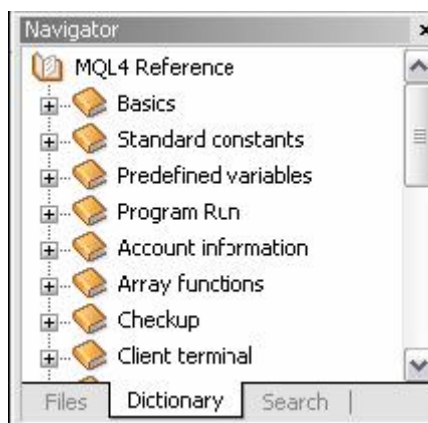


Obr. 6: Prostředí MetaEditoru, panel nástrojů

§ Navigator

Má tři záložky (Files, Dictionary, Search), mezi kterými se můžeme libovolně přepínat.

- Files – zobrazí adresář ve kterém se nachází zdrojové soubory s příponou *.mq4.
- Dictionary – zobrazí seznam příkazů, které je možné v MetaEditoru použít. Můžeme je tažením přetáhnout do pracovního okna nebo dvojklikem otevřít přímo v okně *Toolboxu* nápovědu pro daný příkaz.
- Search – slouží k vyhledávání příkazů, jejich nápověda se zobrazí v okně *Toolboxu*.



Obr. 7: Prostředí MetaEditoru, okno Navigator

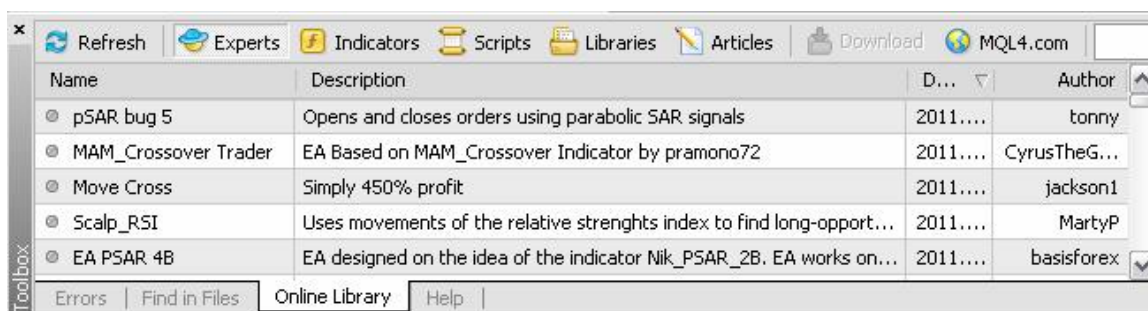
§ Toolbox



Obr. 8: Prostředí MetaEditoru, okno Toolboxu, záložka Help

Má čtyři záložky (Errors, Find in Files, Online Library, Help) mezi kterými se můžeme libovolně přepínat.

- Errors – zobrazuje chyby při kompilaci programu.
- Find in Files – zobrazuje výsledky hledání řetězce ve zvoleném souboru nebo ve zvolené složce. Hledání řetězce v programu vyvoláme v *Edit->Find in Files...*
- Online Library – připojuje se automaticky na server <http://www.mql4.com> a umožňuje stáhnout zdrojové soubory automatických obchodních programů, indikátorů, skriptů a knihoven viz. záložky na obrázku (Experts, Indicators, Scripts, Libraries).



Obr. 9: Prostředí MetaEditoru, okno Toolboxu, záložka Online Library

- Help – zobrazuje se zde nápověda k příkazům, které můžeme vyhledávat v okně *Navigator* v záložce *Search*. Informace k příkazům je možné zobrazit i bez připojení k Internetu.

4 ZÁKLADNÍ MANUÁL K METAQUOTES LANGUAGE 4

Při programování v jazyku MQL4 doporučuji se podívat na stránky s oficiální dokumentací k tomuto jazyku (<http://docs.mql4.com>).

4.1 Syntaxe

Syntaxe je velmi podobná jazyku C, liší se jen v několika částech a to, že

- nemá aritmetickou adresaci,
- nemá operátor *do ... while*,
- nemá operátor *goto ...*,
- nepodporuje operaci *[podmínka]?[1. výraz]:[2. výraz]*,
- nepodporuje složené datové typy (struktury),
- neumožňuje provádět složité operace např. *pom1 = pom2 = 10, pole[i++] = pom*, atd.

4.1.1 Komentáře

Víceřádkové komentáře nemůžou být vnořené. Přitom jednoduché komentáře můžou být vnořené do víceřádkových.

Příklad:

```
// jednoduchý komentář
/*      několika
        řádkový          // vnořený komentář
        komentář
*/
```

4.1.2 Identifikátory

Používají se k pojmenování proměnných, funkcí a typů dat. Délka identifikátorů může být maximálně 31 znaků.

Použitelné symboly jsou

- číslovky 0-9,

- malá a velká písmena latinky a-z a A-Z, rozlišuje se velikost písmen,
- symbol podtržení _.

Identifikátor nemůže začínat číslicí a nesmí kolidovat s vyhrazenými slovy.

Příklady identifikátorů:

PROMENNA1 promenna1 Pom_2

4.1.3 Vyhrazená slova

Jedná se o identifikátory, které jsou asociovány s určitými úkony a nemůžou být použity k jiným účelům.

Datové typy	Paměťové třídy	Operátory	Ostatní
bool	extern	break	false
color	static	case	true
datetime		continue	
double		default	
int		else	
string		for	
void		if	
		return	
		switch	
		while	

4.2 Datové typy

Mezi hlavní datové typy patří

- integer (int),
- boolean (bool),
- literals (char),
- strings (string),
- floating-point number (double),
- color (color),
- date and time (datetime).

4.2.1 Integer (int)

Čísla tohoto typu můžou nabývat hodnot od -2147483648 do 2147483647.

Decimální číslice od 0 do 9, nula by neměla být první číslicí.

Hexadecimální začínají 0x nebo 0X, číslice od 0 do 9, pro vyjádření hodnot 10 až 15 se používají písmena a-f nebo A-F.

Příklady:

```
0xa5, 0xB8, 0x32
```

4.2.2 Boolean (bool)

Nabývají hodnot 0 (false, FALSE) nebo 1 (true, TRUE).

Příklady:

```
bool promenna1 = true;  
bool promenna2 = 0;
```

4.2.3 Literal (char)

Prosté znaky nebo hexadecimální ASCII kódy znaku v apostrofech jsou typu celého čísla. Hodnoty jsou definované od 0 do 255, v případě překročení rozsahu je výsledek nedefinovaný.

Příklad:

```
int a = 'A';           // a = 65  
int b = '\xAE';        // b = 174
```

4.2.4 Strings (string)

Řetězce jsou ohraničeny uvozovkami. Délka řetězce je 0 až 255 znaků, v případě větší délky se přebývající řetězce zprava neberou v úvahu.

```
string a = "řetězec";   // výsledek: řetězec  
string b = "\xAE";      // výsledek: ®
```

Speciální znaky jako obrácené lomítko (\), uvozovky (") a řídicí znaky jsou uvozeny obráceným lomítkem (\):

```
Nový řádek           \n  
Tabulátor            \t  
Obrácené lomítko    \\    \
```

Apostrof	\'	'
Uvozovky	\"	"
Hexadecimální ASCII kód	\xAE	®

4.2.5 Floating-point number (double)

Číslo je dáno jako $n + r$, kde n je celočíselná část (integer part) a r je zlomková část (fractional part), ty jsou od sebe odděleny tečkou.

```
double cislo1 = 1.25;
double cislo2 = -22.00001452;
```

Zahrnují hodnoty v rozmezí -1.7 E^{-308} do 1.7 E^{308} . Pokud číslo přesahuje tento rozsah, je nedefinované.

4.2.6 Color (color)

Barva může být dána třemi způsoby:

- Znakem – tvořeným čtyřmi částmi. Konstantou C a třemi částmi ohraničenými v apostrofech (udávají hodnoty barevných složek v pořadí RGB, nabývají hodnot 0 až 255).

```
C'128,128,128' // šedá
C'0x00,0x00,0xFF' // modrá
```

- Číslem – v decimální nebo hexadecimální formě. Hexadecimální ve formě 0xBBGGRR (RR – velikost zastoupení červené složky, GG – zelené složky, BB – modré složky). V decimální formě nemají hodnoty přímo definované zastoupení v RGB, vychází z hexadecimální reprezentace.

```
0xFFFFFFFF // bílá
16777215 // bílá
```

- Jménem – jedná se o webové barvy.

```
Red
Black
```

4.2.7 Date and time (datetime)

Začínají konstantou D a hodnota uzavřená v apostrofech (') je tvořena šesti částmi (rok, měsíc, den, hodina, minuta, sekunda). Datový typ datetime, může nabývat hodnot od 1. ledna 1970 do 31. prosince 2037.

```
D'2000.05.20 12:30:05'  
D'20.05.2000 12:30:05'  
D'20.05.2010'           // odpovídá zápisu D'20.05.2010 00:00:00'  
D'12:30:05'
```

4.3 Operace a výrazy

Výraz je tvořen operandy a operačními symboly. Pokud za výraz napíšeme středník (;), mluvíme o operátoru. Na jednom řádku může být víc operátorů a jeden operátor může být rozdělen na několik řádků.

```
a++; a = 2;  
x = (a + b) /  
    (a + c) + 10;
```

4.3.1 Aritmetické operace

Patří sem operace jako sčítání (+), odčítání (-), násobení (*), dělení (/), zbytek po dělení (%), inkrementace (++) a dekrementace (--).

```
zbytek = 10 % 4;      // zbytek po dělení  
i++;                  // inkrementace  
j--;                  // dekrementace
```

Operace jako je inkrementace a dekrementace nesmí být použita v jiném výrazu.

```
a++;                  // správně  
int b = (a++)*3       // nelze
```

4.3.2 Operace přiřazení

Ve výrazu může být pouze jedna operace přiřazení.

```
y += x;               // přičtení x k y  
y -= x;               // odečtení x od y  
y *= x;               // vynásobení y proměnnou x  
y /= x;               // dělení y proměnnou x  
y %= x;               // celočíselné dělení y proměnnou x  
y >>= x;              // logický posun v y doprava v x-tém bitu  
y <<= x;              // logický posun v y doleva v x-tém bitu  
y &= x;               // bitová operace AND  
y |= x;               // bitová operace OR  
y ^= x;               // bitová operace XOR
```


4.3.3 Porovnávací operace

U porovnávání nelze porovnávat dvě desetinná čísla, musí se od sebe nejprve odečíst a pak porovnávat s nulou.

```
a == b;           // pokud a je rovno b, vrátí TRUE
a != b;           // pokud a není rovno b, vrátí TRUE
a < b;            // pokud a je menší než b, vrátí TRUE
a > b;            // pokud a je větší než b, vrátí TRUE
a <= b;           // pokud a je menší nebo rovno b, vrátí TRUE
a >= b;           // pokud je a je větší nebo rovno b, vrátí TRUE
```

4.3.4 Boolean operace

NOT – slouží k negaci výrazu.

```
if (!b);           // pokud b bude FALSE, výsledek bude TRUE
```

AND – se používá pro logický součin dvou proměnných.

```
if (a && b);        // pouze v případě že a i b budou TRUE, bude
                    // výsledek TRUE
```

OR – je pro logický součet dvou proměnných.

```
if (a || b);        // pokud bude alespoň jedna z proměnných a nebo b TRUE,
                    // bude výsledek TRUE
```

Použitím porovnávacích a boolean operací můžeme tvořit složitější podmínky:

```
if ((a == b) && (b < c) || (c != b));
```

4.3.5 Bitové operace

Bitové operace lze provádět jen s celými čísly.

Negace – hodnota výrazu a bude obsahovat 1 na všech pozicích kde b obsahuje 0. Ale také a bude obsahovat 0 všude kde b obsahuje 1.

```
a =~ b;
```

Bitový posun vpravo o y pozic – bity zleva budou vyplněny nulami.

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

Bitový posun vlevo o y pozic – bity zprava budou vyplněny nulami.

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

Bitový AND – výsledek výrazu je roven 1 na těch pozicích, kde x **a současně** y je rovno 1. Na ostatních pozicích je pak rovno nule.

$$a = b \ \& \ c;$$

Bitový OR – výsledek výrazu je roven 1 na těch pozicích, kde x **nebo** y je rovno 1. Na ostatních pozicích je pak rovno nule.

$$a = b \ | \ c;$$

Bitový XOR – výsledek výrazu je roven 1 na těch pozicích, kde x **má jinou hodnotu** než y .

$$A = b \ ^ \ c;$$

4.4 Operátory

Operátory if, if-else, switch, for, while, můžou být do sebe navzájem vkládány.

4.4.1 Operátor if

```
if (vyraz)
    operator;
```

pokud je *vyraz* TRUE, vykoná se *operator*.

4.4.2 Operátor if-else

```
if (vyraz)
    operator1;
else
    operator2;
```

Pokud je *vyraz* pravdivý (TRUE), vykoná se *operator1*. Pokud není pravdivý (FALSE), vykoná se *operator2*.

4.4.3 Operátor switch

```
switch (vyraz){
    case konstanta1:
        operator1_1;
        operator1_2;
        break;
    case konstanta2:
        operator2_1;
        operator2_2;
        break;
```

```
...
default:
    operator;
}
```

Dochází k porovnávání *vyrazu* s konstantami. V případě shody se vykonají operátory, pod označením dané konstanty. V případě, že *vyrazu* neodpovídá žádná konstanta, vykonají se operátory pod označením *default*. Pokud se *default* nevyskytuje, tak se nic neprovede a program pokračuje.

Za operátory musí být umístěn operátor *break*, jinak by program pokračoval na operátory odpovídající hodnotě další konstanty.

Jednotlivé konstanty nesmí obsahovat proměnné např. „case a+b:“, musí být dány konstanty např. „case 2:“ nebo „case 1+2:“.

4.4.4 Operátor while

```
while (vyraz)
    operator;
```

Dokud platí, že je hodnota *vyrazu* TRUE, operátor se neustále vykonává. V okamžiku, kdy bude mít *vyraz* hodnotu FALSE, cyklus končí.

4.4.5 Operátor for

```
for (vyraz1; vyraz2; vyraz3)
    operator;
```

Vyraz1 je inicializován na začátku cyklu. *Vyraz2* se při každém cyklu testuje, zda daná podmínka ještě platí. *Vyraz3* je proveden při každém cyklu, používá se jako čítač cyklu.

4.5 Funkce

Funkce může být volána z jakékoliv části programu. Funkce je tvořena blokem operací, které mohou mít návratovou hodnotu, nebo taky nemusí.

4.5.1 Definice funkcí

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

Neboli:

```
datový_typ_návratové_hodnoty název_funkce (dat_typ parametr1, dat_typ
parametr1){
    lokální_proměnné;
    příkazy_vykonávané_uvnitř_fce;
    return (návratová_hodnota);
}
```

Pokud by tělo funkce neobsahovalo operátor *return*, byl by datový typ funkce *void*, tedy funkce by nevracela žádnou hodnotu, pouze by vykonala operace obsažené v jejím těle.

4.5.2 Volání funkce

Volání funkce s návratovou hodnotou:

```
double a = fce1 (3.2, 4.5);
```

Neboli:

```
dat_typ proměnná = název_funkce (parametr1, parametr2);
```

Volání funkce bez návratové hodnoty a bez vstupních parametrů:

```
fce2();
```

4.5.3 Speciální funkce

- Init()

Tato funkce je volána při zapnutí klientského terminálu, při změně periodicity grafu a změně finančního instrumentu. Používají se v ní např. fce pro vytvoření objektů, které budou v grafu používány. Pokud není na začátku programu potřebná inicializace, může zůstat tato funkce prázdná. [8]

- Start()

V případě, že tato funkce v našem programu chybí, nemůže být náš program spuštěn.

V EA je funkce volána v okamžiku obdržení nové hodnoty obchodovaného instrumentu. [8]

U programu indikátoru je tato funkce volána v okamžiku, kdy je indikátor přidán do grafu, při otevření klientského terminálu a také v okamžiku, kdy dojde ke změně hodnoty obchodovaného instrumentu. [8]

V případě skriptu je tato funkce volána při spuštění skriptu. [8]

- Deinit()

Funkce volaná v okamžiku uzavření klientského terminálu. Slouží např. pro odstranění již vytvořených objektů. Pokud bychom v této funkci dříve vytvořené objekty neodstranili, zůstaly by na grafu i když bychom náš indikátor nebo EA z grafu odstranili. [8]

II. PRAKTICKÁ ČÁST

5 INSTALACE METATRADER 4

Instalace a nastavení obchodní platformy MetaTrader 4 je velmi jednoduchá a popíšeme si ji v jednotlivých bodech:

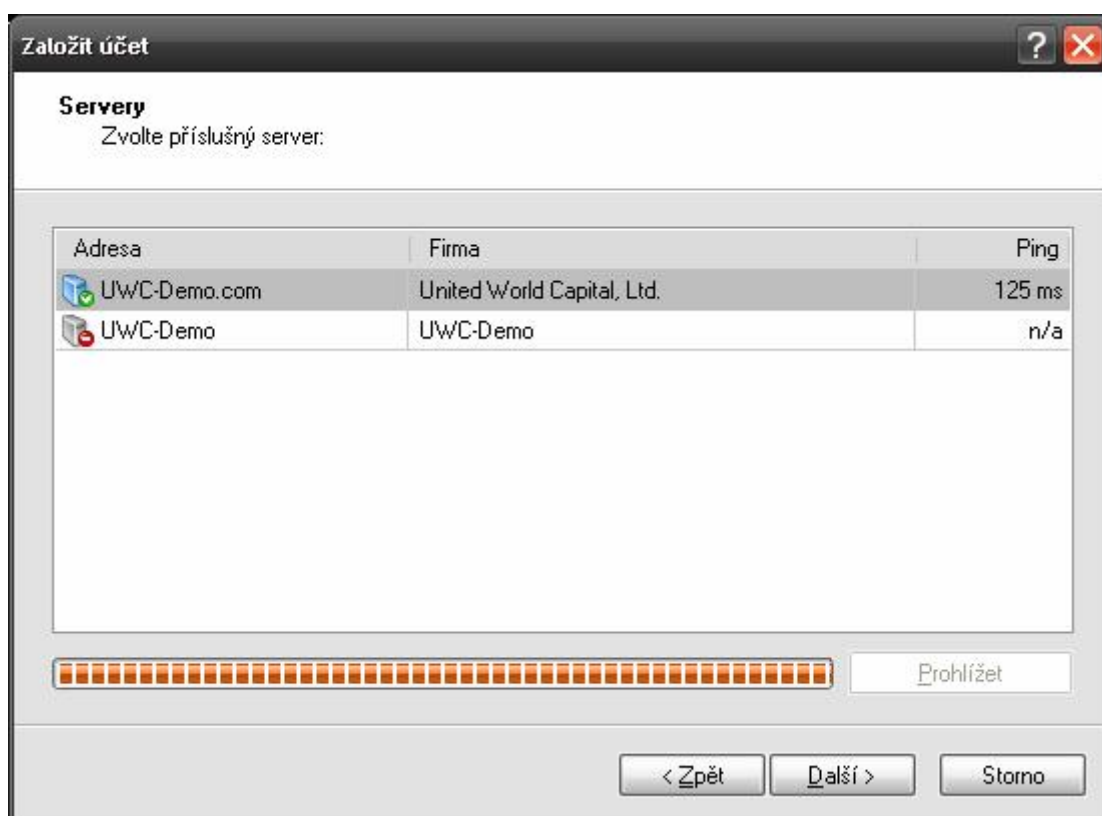
Přejdeme na adresu http://www.metaquotes.net/en/metatrader4/trading_terminal, kde se mimo jiné můžeme dočíst informace o této platformě. V případě, že nás tyto informace moc nezajímají, nebo je už víme, tak se posuneme až dolů na konec této stránky, kde je odkaz na stáhnutí tohoto softwaru. Přímý odkaz na tento software je zde <http://files.metatrader4.com/mt4setup.exe>. Velikost je 5,5Mb.

Po stáhnutí programu provedeme instalaci, stačí když všude klikneme na tlačítko *Další*, protože se nás Wizard, kromě volby kam chceme software nainstalovat, na nic důležitého neptá.

Při prvním spuštění MT4 se zobrazí následující dialogové okno:

Obr. 10: Dialogové okno Založit účet, Osobní údaje

Název označuje název účtu. Typ účtu můžete ponechat forex-usd. Vyplňte zbývající políčka a pokračujte na další okno:



Obr. 11: Dialogové okno Založit účet, Servery

V tomto okně klikněte na tlačítko *Prohlížeč*, které vám zobrazí dostupné servery. Vyberte si některý z dostupných serveru a pokračujte na další okno:

Obr. 12: Dialogové okno Založit účet, Registrace

Klikněte na tlačítko *Dokončit* a ihned by se vám měly načíst data do grafů. Pokud by se tak nestalo, stačí se přihlásit v *Soubor→Přihlásit*, kde se vyplní přihlašovací jméno, heslo a z nabídky se vybere server z kterého nám budou chodit data.

V tomto okamžiku máme nainstalovaný terminál MT4, do kterého se z Internetu stahují aktuální data. K dispozici budou pouze data měnových párů.

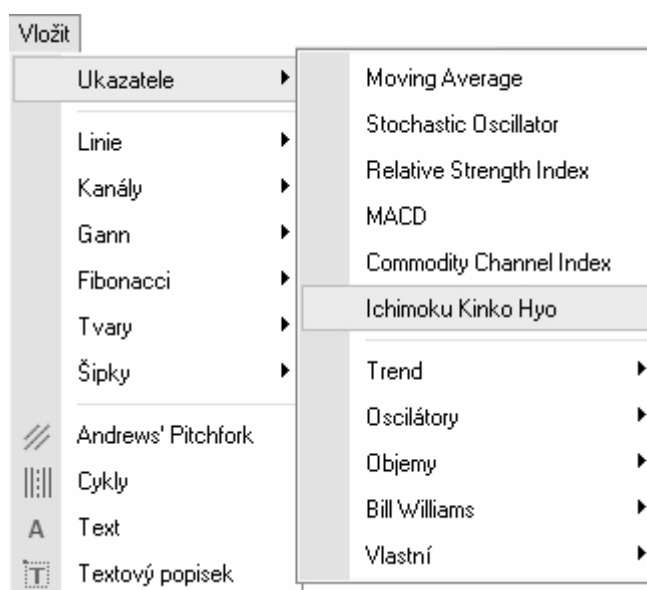
V případě, že bychom chtěli obchodovat komodity, akcie, aj., je nutné si otevřít účet u brokerské společnosti, některé jsem uváděl na začátku mé práce. V takovém případě se budeme muset u této společnosti zaregistrovat přes Internet a stáhnout si MT4 z jejich webu. Na email nám budou zaslány přihlašovací údaje, které vyplníte v *Soubor→Přihlásit*. Pokud bychom měli s otevřením účtu nějaký problém, je nejlepší se obrátit přímo na tuto společnost a oni s námi problém vyřeší. Na webech těchto společností také najdeme návody, které popisují založení účtu, podle kterých můžeme postupovat.

6 POUŽITÍ PROGRAMŮ

Programy, které si vytvoříme, se velmi snadno aplikují na daný graf. Začneme od těch nejjednodušších což jsou indikátory a skripty až po ty složitější programy. Předtím než použijeme některý EA je třeba zkontrolovat nastavení, jinak nám v místě označovaném jako *Deník* začnou naskakovat errorry. Předtím než nějaký EA použijeme, je dobré ho nejdříve otestovat na historických datech a k tomu slouží *Tester strategií*.

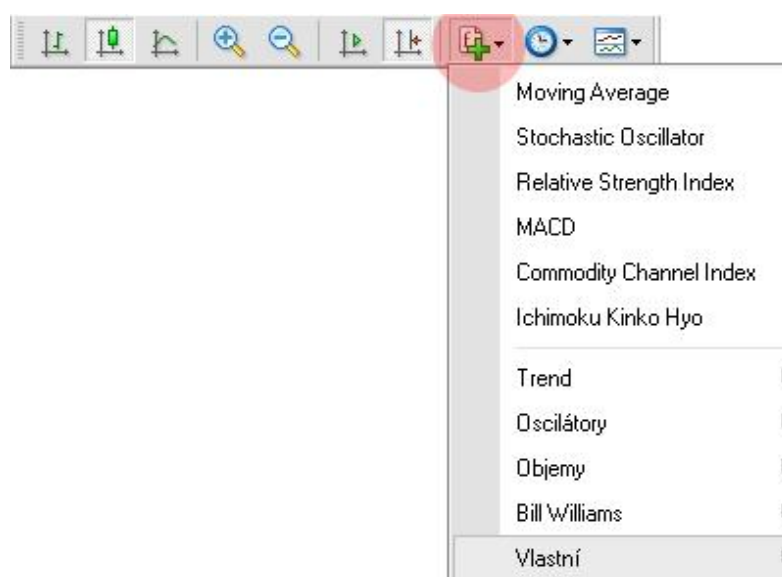
6.1 Použití indikátorů a skriptů

Indikátory můžeme vložit do grafu třemi způsoby. Prvním způsobem je vložení z nabídky *Vložit→Ukazatele*, kde se nacházejí indikátory, které obsahuje každý MT4 již od nainstalování. Vlastní ukazatele najdeme v *Vložit→Ukazatele→Vlastní*.



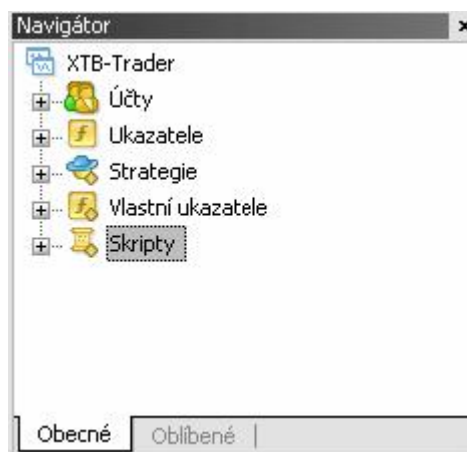
Obr. 13: Ukázka vložení indikátoru z roletové nabídky *Vložit*

Druhým způsobem vložení je vložení z panelu nástrojů z nabídky *Grafy*. V případě, že tento nástroj nemáme dostupný, necháme si ho zobrazit zatržením *Grafy* v *Pohled→Nástroje→Grafy*.



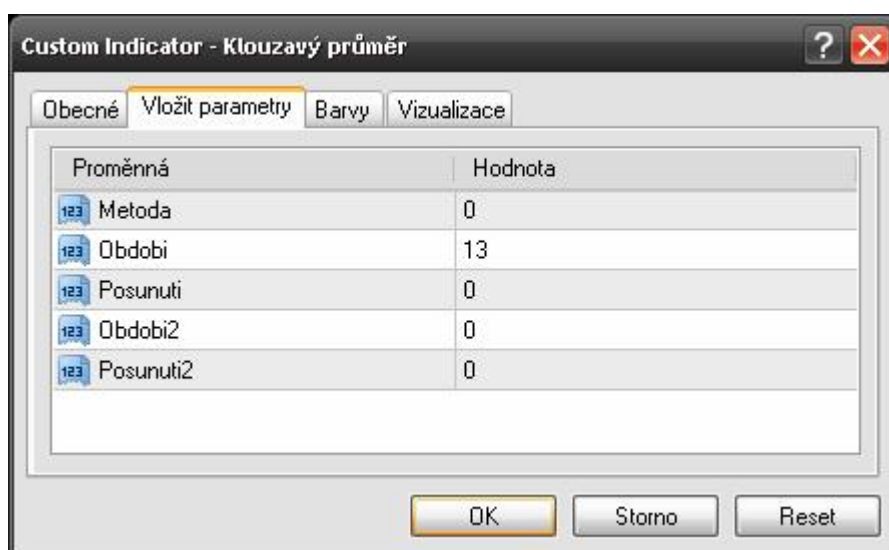
Obr. 14: Ukázka vložení indikátoru z panelu nástrojů

Třetím způsobem vložení indikátoru, který je taky jediným způsobem pro použití skriptu, je spuštění indikátoru (skriptu, EA) z okna *Navigátor* na záložce *Obecné*. Spuštění provedeme dvojklikem na vybraný program, nebo jeho přetažením z okna *Navigátoru* do grafu, ve kterém chceme program spustit.



Obr. 15: Okno Navigátor

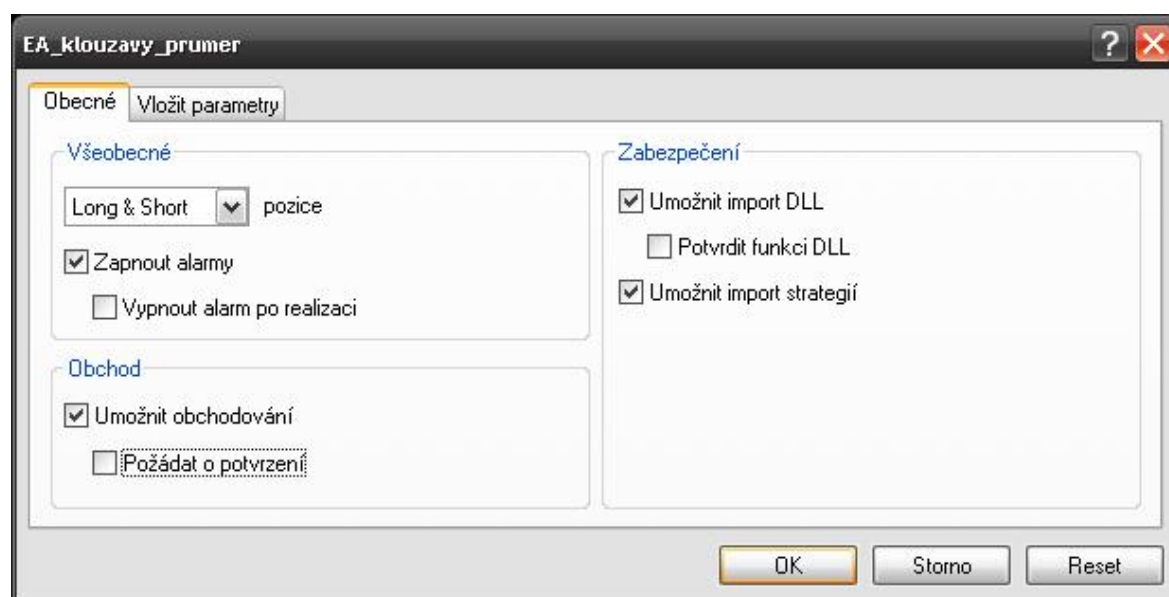
Po vložení zvoleného indikátoru (skriptu, EA) do grafu se vyvolá dialogové okno, které souvisí s nastavením indikátoru (skriptu, EA). Dialogová okna vlastních indikátorů (skriptů, EA) se trochu od zabudovaných indikátorů liší, především na záložce pro nastavení parametrů daného indikátoru. Zde je pro ukázkou dialogové okno pro nastavení parametrů u vlastního indikátoru:



Obr. 16: Dialogové okno pro nastavení parametrů indikátoru

6.2 Použití a nastavení strategií

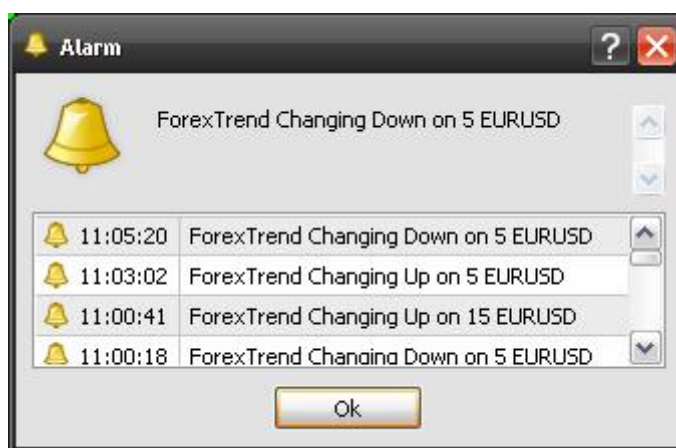
Strategii vybereme z okna *Navigátor* a vložíme ji do grafu. Tím strategií aplikujeme na náš graf a vyvolá se dialogové okno s nastavením. Na záložce *Obecné* nastavíme požadavky, které jsou pro všechny strategie stejné. Na záložce *Vložit parametry* nastavíme parametry strategie. Tyto parametry jsou pro každou strategií odlišné a je na autorovi naprogramované strategie, jaké parametry nás nechá nastavit.



Obr. 17: Dialogové okno pro obecné nastavení aplikované strategie

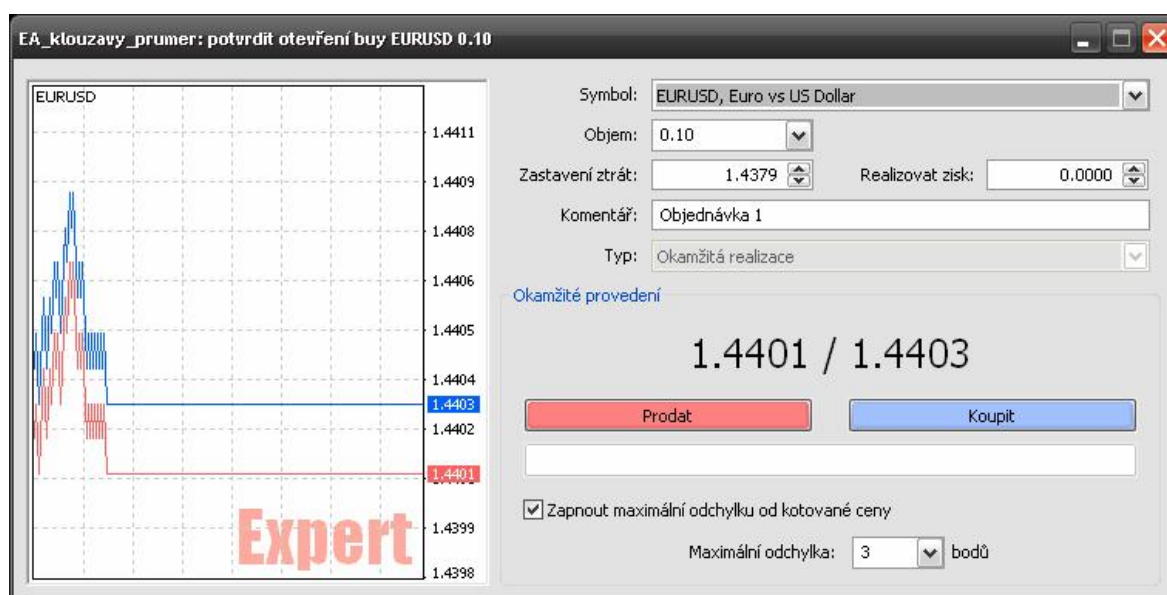
V tomto okně můžeme nastavit zda se má strategie použít jen na *Only Long* (nákup) , *Only Short* (prodej) nebo *Long & Short* obchodování. Pokud zatrhneme *Zapnout alarmy*, tak umožníme, aby se zobrazovaly alarmy, které v kódu představují funkce `Alert()`.

V části *Zabezpečení* znamená *Umožnit import DLL*, že je umožněn import DLL knihoven, které používají některé strategie a *Umožnit import strategii* umožňuje import strategii do investiční platformy.



Obr. 18: Dialogové okno Alarm

Nejdůležitější je zatrhnout *Umožnit obchodování* v části *Obchod*, tím povolíme uzavírání obchodů na základě naprogramované strategie. V případě, že chceme být před každým uzavřeným obchodem informováni zatrhneme *Požádat o potvrzení*, tím si zabezpečíme, že se bez našeho souhlasu neuzavře žádný obchod, protože se před každým obchodem zobrazí dialogové okno, ve kterém objednávku potvrdíme kliknutím na tlačítko *koupit* nebo *prodat*. Parametry jako jsou *Symbol*, *Objem*, *Zastavení ztrát* (neboli SL), *Realizovat zisk* (neboli TP), *Komentář*, *Typ* a *Maximální odchylka* (slippage), jsou automaticky přednastavené podle naprogramované strategie, ale můžeme si je před uzavřením obchodu upravit podle naší potřeby.



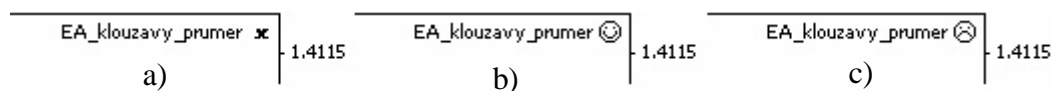
Obr. 19: Dialogové okno pro potvrzení obchodu u spuštěné strategie

Pokud při zavádění strategie provedeme potřebné nastavení, které jsme zde zmínili, aplikuje se strategie na náš graf. To ještě ale neznamená, že je strategie spuštěná. Spuštění strategie provedeme tlačítkem *Zapnout strategii*, které najdeme na panelu nástrojů.



Obr. 20: Panel nástrojů Standard s tlačítkem „Zapnout strategii“

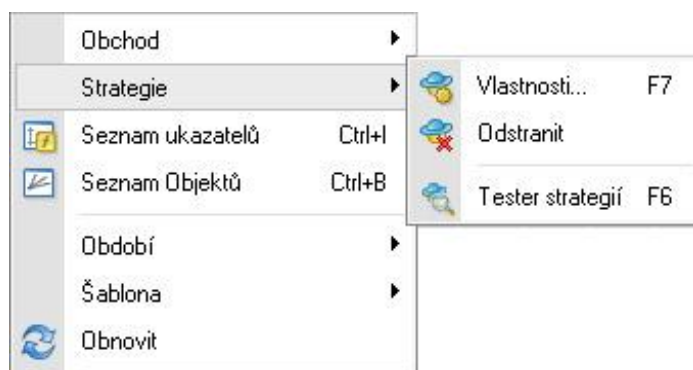
Stav spuštěné strategie se nám zobrazuje v pravém horním rohu grafu a zobrazuje nám název aplikované strategie a hned za ním symbol označující stav strategie.



Obr. 21: Symboly označující stavy strategie

- a) Strategie, která je na graf aplikovaná, ale není spuštěná.
- b) Spuštěná strategie u které je umožněno obchodování.
- c) Spuštěná strategie u které není povolené obchodování.

V případě, že potřebujeme aplikovanou strategii modifikovat nebo odstranit, stačí kliknout pravým tlačítkem myši kdekoli v grafu a zvolit volbu *Strategie→Vlastnosti...* nebo *Strategie→Odstranit*.



Obr. 22: Kontextová nabídka pro změnu vlastností
nebo odstranění strategie z grafu

6.3 Tester strategie

Před každým použitím strategie označované v MT4 jako EA, je vhodné si tuto strategii nejprve otestovat. Je jedno jestli je to váš vlastní výtvar, nebo je strategie stažená z internetu, vždy je dobré ji otestovat na historických datech. K tomu slouží v MT4 tzv. Tester strategie.

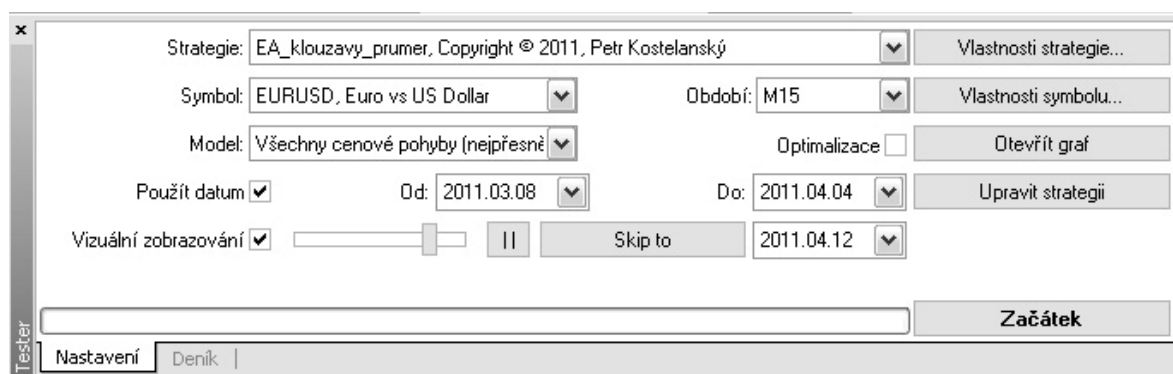
Tester strategie vyvoláme z roletové nabídky *Pohled* → *Tester strategií*. Pod grafem se nám zobrazí okno *Tester* se záložkami *Nastavení* a *Deník*.

V políčku *Strategie* vybere zvolenou strategii. V případě, že se nám v seznamu nezobrazuje, je nutné její kód nejprve zkompileovat v MetaEditoru a pak se nám v seznamu zobrazí. V *Symbol* vybereme na jakém instrumentu chceme strategii testovat.

V *Model* máme na výběr z:

- Všechny cenové pohyby
- Kontrolní body
- Pouze otevírací ceny

Nejlepší a nejpresnější způsob je zvolit *Všechny cenové pohyby*.



Obr. 23: Okno Tester, záložka Nastavení

Dále zvolíme datum, který omezuje interval historických dat, na kterých chceme strategii testovat. Můžeme si nechat zobrazit vizualizaci strategie na grafu. Na posuvníku si nastavíme rychlost průběhu vizualizace. Pomocí tlačítka *Skip to* můžeme za běhu testování vynechat oblast od aktuální testované hodnoty po nastavené datum, od kterého bude strategie znovu pokračovat. Dále je důležité nastavit vlastnosti strategie. Kliknutím na *Vlastnosti strategie...* se vyvolá okno, které obsahuje záložky *Testování*, *Vložit parametry* a *Optimalizace*. Kde na záložce *Vložit parametry* nastavujeme parametry strategie, které jsou vstupními parametry pro danou strategii. Pod tlačítkem *Vlastnosti symbolu...* se ukrývají informace o instrumentu na kterém budeme strategii testovat, ukrývá informace jako spread, limitní odstup, atd.

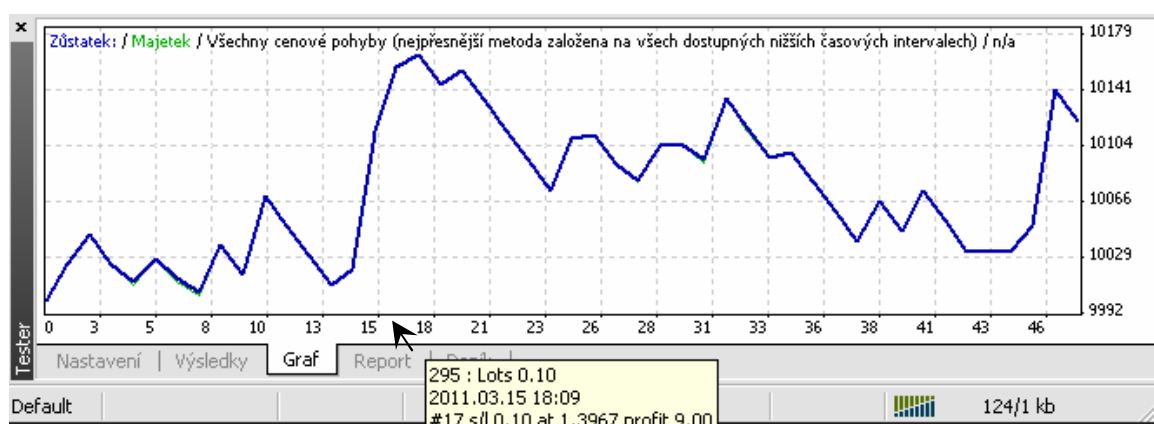
Po stisku tlačítka *Začátek* otestujeme strategii na historických datech. Jakmile test doběhne, zobrazí se nám dole v okně *Tester* tyto záložky:

- **Nastavení** – záložka, kterou jsme si už rozebrali, slouží pro nastavení testeru strategie.
- **Výsledky** – zobrazuje volané příkazy, příkazy pro nákup (buy), prodej (sell), úpravu příkazu při posunutí stoplossu (modify) a zavření pozice díky stoplossu (s/l) nebo vystoupení z pozice pomocí takeprofit (t/p).

#	Čas	Typ	Pokyn	Lotů	Cena	Zastavení ztrát	Realizovat zisk	Zisk	Zůstatek:
554	2011.04.08 09:28	modify	46	0.10	1.4309	1.4396			
555	2011.04.08 09:29	modify	46	0.10	1.4309	1.4397			
556	2011.04.08 09:29	modify	46	0.10	1.4309	1.4398			
557	2011.04.08 09:29	modify	46	0.10	1.4309	1.4399			
558	2011.04.08 09:45	modify	46	0.10	1.4309	1.4400			
559	2011.04.08 10:06	s/l	46	0.10	1.4400	1.4400		90.63	10141.36
560	2011.04.11 14:45	sell	47	0.10	1.4427	1.4447			
561	2011.04.11 15:44	s/l	47	0.10	1.4447	1.4447		-20.00	10121.36

Obr. 24: Okno Tester, záložka Výsledky

- Graf – zobrazuje průběh stavu našeho účtu, kde zobrazuje zůstatek a majetek. Vyjadřuje závislost stavu účtu na prováděných obchodech. Při najetí na osu x se nám zobrazí bublina, která obsahuje informace o uzavřeném obchodu. Pokud v tomto místě provedeme dvojklik levým tlačítkem myši, tak se přesuneme na záložku *Výsledky* přímo na konkrétní řádek dané objednávky. To samé platí pokud dvojklikneme na nějakou křivku v grafu.



Obr. 25: Okno Tester, záložka Graf

- Report – na této záložce se zobrazuje zpráva o výsledku obchodování strategie.

Testování sloupco...	3857	Značky namodelovány	989532	Modelování kvality	n/a
Mismatched chart...	403				
Počáteční depozit	10000.00				
Celkový čistý zisk	121.36	Hrubý zisk	578.27	Hrubá ztráta	-456.91
Ziskový faktor	1.27	Předpokládaný zisk	2.58		
Absolutní pokles	3.34	Maximální pokles (%)	170.81 (1.6...	Relativní pokles	1.68% (170...
Transakce celkem	47	Krátké pozice (výhra %)	24 (50.00%)	Dlouhé pozice (výhra %)	23 (43.48%)
		Ziskové obchody (% z ...	22 (46.81%)	Ztrátové obchody (% ...	25 (53.19%)
	Největší	ziskový obchod	91.89	ztrátový obchod	-20.37
	Průměr	ziskový obchod	26.29	ztrátový obchod	-18.28
	Maximum	návazné výhry (finanč...	4 (154.89)	návazné prohry (finan...	4 (-80.00)
	Maximální	návazný zisk (count of ...	154.89 (4)	návazná ztráta (počet ...	-80.00 (4)
	Průměrný	návazné výhry	2	návazné prohry	2

Obr. 26: Okno Tester, záložka Report

- Deník – zobrazuje informace o veškerých prováděných procesech, zprávy sem vkládá sám terminál a informuje nás o uzavřených obchodech. Zobrazuje zde veškeré chybové hlášky např. při neotevření požadované pozice. Informace sem může vypisovat i náš program pomocí příkazu Print().

Čas	Zpráva
2011.04.13 11:3...	2011.04.11 15:44 Tester: stop loss #47 at 1.4447 (1.4446 / 1.4448)
2011.04.13 11:3...	2011.04.11 14:45 EA_klouzavy_prumer EURUSD,M15: Alert: Objednávka 47 provedena n v čase: 2...
2011.04.13 11:3...	2011.04.11 14:45 EA_klouzavy_prumer EURUSD,M15: Objednávka provedena
2011.04.13 11:3...	2011.04.11 14:45 EA_klouzavy_prumer EURUSD,M15: open #47 sell 0.10 EURUSD at 1.4427 sl: 1....
2011.04.13 11:3...	2011.04.08 10:06 Tester: stop loss #46 at 1.4400 (1.4400 / 1.4402)
2011.04.13 11:3...	2011.04.08 09:45 EA_klouzavy_prumer EURUSD,M15: Úprava: 0-46 na SL: 1.44 TP: 0
2011.04.13 11:3...	2011.04.08 09:45 EA_klouzavy_prumer EURUSD,M15: modify #46 buy 0.10 EURUSD at 1.4309 sl: ...

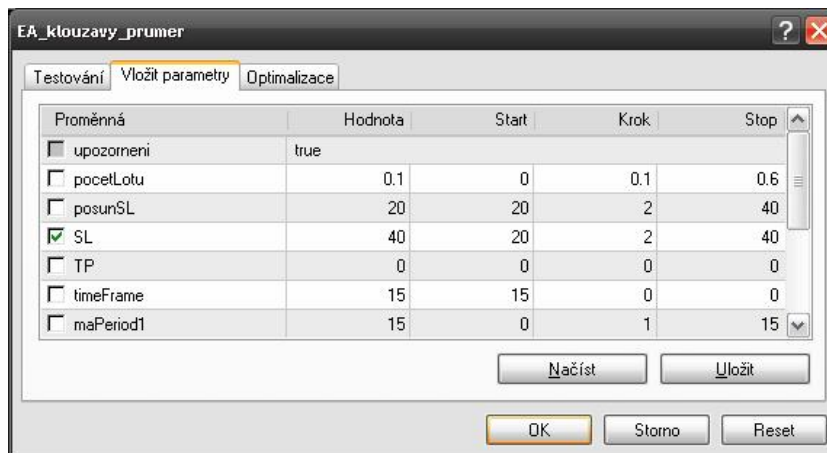
Obr. 27: Okno Tester, záložka Deník

6.3.1 Optimalizace strategie

Aby se při testování strategie provedla optimalizace a bylo nám zobrazeno vyhodnocení optimalizace, je potřeba:

- Zaškrtnout možnost *Optimalizace* v okně *Tester* na záložce *Nastavení*.
- Nastavit parametry strategie, které se mají optimalizovat a to kliknutím na *Vlastnosti strategie...* v okně *Tester* na záložce *Nastavení*.

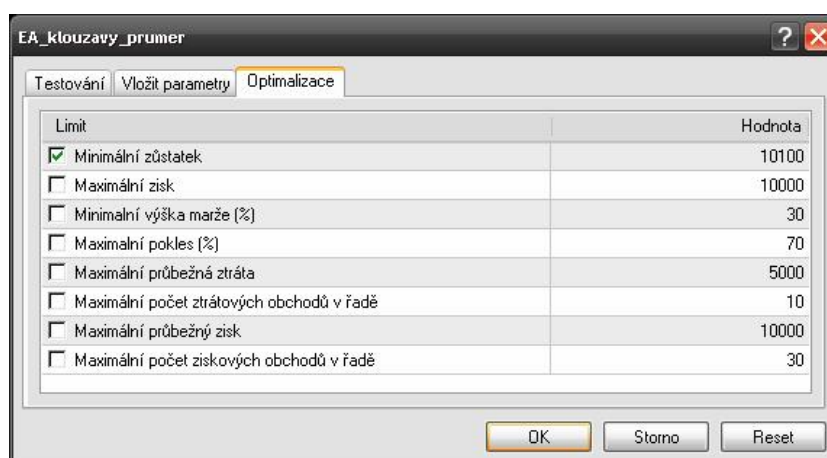
Pro kliknutí na *Vlastnosti strategie...* se otevře dialogové okno pro danou strategii.



Obr. 28: Okno pro nastavení parametrů testované strategie, záložka *Vložit parametry*

Každý řádek na záložce *Vložit parametry* představuje parametry strategie. U každého parametru můžeme nastavit *Hodnotu* se kterou se počítá, pokud parametr neoptimalizujeme. *Start*, *Krok* a *Stop* představují hodnoty, které se využijí pro výpočet optimalizace a berou se v úvahu pokud vybraný parametr zaškrtneme. Tím také určíme parametry, které budeme optimalizovat.

Na záložce *Optimalizace* máme možnost dalšího nastavení optimalizace strategie. Stačí zatrhnout zvolenou možnost a nastavit hodnotu. Tyto nastavení slouží k eliminaci výsledků optimalizace, tak aby se nám zobrazily pouze takové výsledky, které splňují požadavky nastavené na záložce *Optimalizace*.



Obr. 29: Okno pro nastavení parametrů testované strategie, záložka *Optimalizace*

Po stisku tlačítka *Začátek* otestujeme strategii a současně se provedou výpočty optimalizace. Jakmile test doběhne, zobrazí se v okně *Tester* záložky optimalizace:

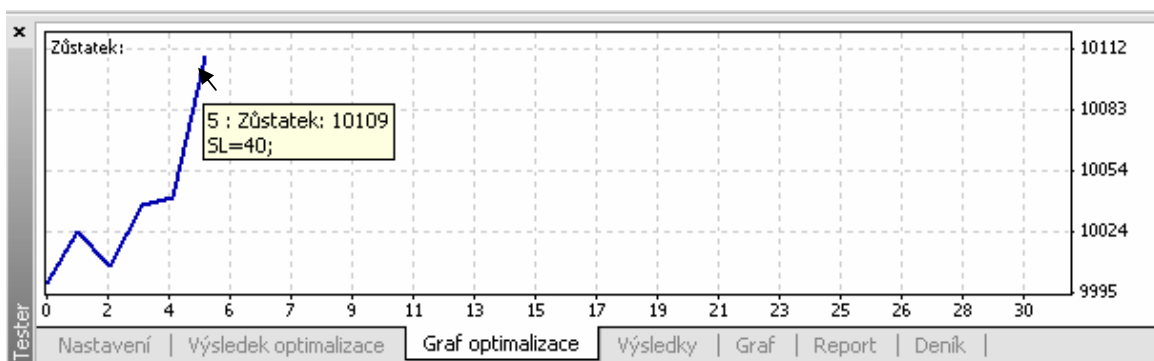
- Výsledek optimalizace – zobrazí několik sloupců, přitom zobrazuje pouze takové možnosti nastavení strategie, které jsou ziskové. Nezisková nastavení jsou skrytá, ale je možné si je nechat vypsat pokud odtrhneme *Přeskočit nepoužitelné výsledky* v kontextové nabídce po kliknutí pravým tlačítkem myši v tabulce výsledků optimalizace.

Test	Zisk	Transakce celkem	Faktor zisku	Přepokládaný zisk	Pokles \$	Pokles %	Vložit parametry
1	24.23	36	1.07	0.67	140.33	1.39	SL=32; pocetLotu=0.1; posun...
2	8.23	36	1.02	0.23	146.33	1.45	SL=34; pocetLotu=0.1; posun...
3	37.10	36	1.11	1.03	152.33	1.50	SL=36; pocetLotu=0.1; posun...
4	41.10	36	1.12	1.14	154.33	1.52	SL=38; pocetLotu=0.1; posun...
5	108.97	36	1.36	3.03	147.33	1.44	SL=40; pocetLotu=0.1; posun...

Obr. 30: Okno *Tester*, záložka *Výsledek optimalizace*

Ve sloupci *Zisk* vidíme, která možnost nastavení strategie přinesla největší zisk a ve sloupci *Vložit parametry* jsou parametry, které byly pro test strategie použity. Poklikáním na některý řádek, nastavíme do testeru strategie parametry strategie, které jsou na daném řádku ve sloupci *Vložit Parametry* a současně budeme přesměrování na záložku *Nastavení*, abychom mohli strategii s těmito parametry otestovat.

- Graf optimalizace – zobrazuje jednotlivé ziskové nastavení strategie a velikost dosaženého zisku. Místo použitého spojnicového grafu by byl názornější histogram, .



Obr. 31: Okno *Tester*, záložka *Graf optimalizace*

7 CUSTOM INDICATOR

Jako CI, tedy indikátor sloužící k technické analýze vývoje trhu, jsem vytvořil indikátor klouzavý průměr.

7.1 Definice parametrů preprocesoru a externích proměnných

```
1 #property copyright "Copyright © 2011, Petr Kostelanský"
2 #property link      ""
3 #property indicator_chart_window
4 #property indicator_buffers 2
5 #property indicator_color1 Red
6 #property indicator_color2 Lime
7
8 extern int maPeriod = 13;
9 extern int maMethod = 0;
10 extern int maShift = 0;
11
12 extern int maPeriod2 = 0;
13 extern int maMethod2 = 0;
14 extern int maShift2 = 0;
15
16 double Buffer1[];
17 double Buffer2[];
18
19 int poz = 0;
20 int poz2 = 0;
21 int spocitanychSvic = 0;
22 double soucet = 0;
23 double soucet2 = 0;
```

Všechny řádky začínající slovem *#property* jsou preprocesorové příkazy, což jsou příkazy pro kompilátor, které se vykonají před spuštěním kódu. Ukončovacím symbolem není středník (;).

#property copyright "" – obsahuje jméno autora, nebo společnosti. Datový typ string.

#property link "" – pro vložení odkazu na autorovi stránky, nebo společnosti. Datový typ string.

Oba uvedené preprocesory je možné vyplnit při zakládání nového programu, kde nás průvodce vyzve k jejich zadání.

#property indicator_chart_window – oznamuje kompilátoru, že vykreslovaný indikátor bude umístěn v grafu zobrazovaného instrumentu. Opačnou možností je *indicator_separate_window*, tato volba vykreslí indikátor pod grafem ve vlastním oknu. Oba mají datový typ void.

#property indicator_buffers 2 – oznamujeme tím, kolik vyrovnávacích pamětí pro vykreslování do grafu budeme používat, maximálně však 8.

#property indicator_color1 Red – označuje barvu, kterou budeme při vykreslování používat, přípustné varianty jsou *indicator_color1* až *indicator_color8*.

Preprocesorový příkaz *#property* umožňuje používat i jiné parametry, ty naleznete v oficiální dokumentaci MQL4 (www.mql.com).

Kromě příkazu *#property* je možné použít i *#define*, *#include* a *#import*, ty mají však jiný význam.

Zápis **extern datový_typ proměnná = hodnota** slouží k vytvoření externích proměnných představujících parametry indikátoru, které bude moci uživatel nastavovat ve vlastnostech indikátoru, aniž by musel zasahovat do zdrojového kódu. V případě, že bychom náš program chtěli někomu poskytnout a nechtěli bychom, aby měl možnost náš kód jakkoliv editovat, poskytneme mu pouze soubor s příponou *.ex4. Tím poskytneme zkompilovanou verzi programu, kde použitím externích proměnných umožníme uživateli, aby mohl chování indikátoru ovlivňovat podle svých potřeb.

double Buffer1[] – představuje definici proměnné pole, datového typu double.

Dále definujeme globální proměnné a to především ty proměnné, které budeme chtít využívat ve více funkcích např. v *init()*, *deint()*, *start()* anebo v našich vlastních funkcích.

7.2 Funkce *init()*

Jak už bylo zmíněno, jedná se o funkci, která proběhne pouze v případech, kdy indikátor vložíme do grafu nebo když změníme vlastnosti indikátoru. V jiných případech se funkce nespouští.

```
25     string nazevMetody2;  
26     IndicatorBuffers(2);
```

Ve funkci `init()` si nejprve vytvoříme dvě proměnné *nazevMetody* a *nazevMetody2*. Bude se jednat o textové řetězce, tedy datový typ `string`. Do těchto proměnných si posléze uložíme názvy metod klouzavých průměrů, které zobrazíme na grafu, aby měl uživatel přehled, jaká metoda klouzavého průměru je na graf aplikovaná.

IndicatorBuffers(počet) - je funkce která slouží k alokaci paměti, ze které se budou vykreslovat data do grafu

```
27     if(maPeriod2 == 0){  
28         SetIndexBuffer(0,Buffer1);  
29         SetIndexStyle(0,DRAW_LINE, STYLE_SOLID, 1, Red);  
30         SetIndexShift(0,maShift);  
31         SetIndexDrawBegin(0,NULL);  
32     }else if(maPeriod == 0){  
33         SetIndexBuffer(1,Buffer2);  
34         SetIndexStyle(1,DRAW_LINE, STYLE_SOLID, 1, Red);  
35         SetIndexShift(1,maShift2);  
36         SetIndexDrawBegin(1,NULL);  
37     }else{  
38         SetIndexBuffer(0,Buffer1);  
39         SetIndexStyle(0,DRAW_HISTOGRAM, EMPTY, 2, Red);  
40         SetIndexShift(0,maShift);  
41         SetIndexDrawBegin(0,NULL);  
42  
43         SetIndexBuffer(1,Buffer2);  
44         SetIndexStyle(1,DRAW_HISTOGRAM, EMPTY, 2, Lime);  
45         SetIndexShift(1,maShift2);  
46         SetIndexDrawBegin(1,NULL);  
47     }
```

Nejprve si řekněme co tato část kódu dělá. Jedná se test, zda uživatel zadal periodu pro druhý (první) klouzavý průměr, tedy jestli v grafu budou vykresleny dva nebo jen jeden. V případě, že *maPeriod2* bude nula a *maPeriod* bude mít nastavenou hodnotu, bude vykreslen pouze první klouzavý průměr, křivka bude mít tvar čáry. Pokud uživatel vloží do proměnné *maPeriod2* nějakou hodnotu, vytvoří se v grafu oblast mezi těmito křivkami, která bude vyšrafovaná (za použití histogramu). Z kódu je jasné, že to platí i obráceně,

pokud *maPeriod2* bude mít hodnotu a *maPeriod* bude nula vykreslí se křivka (bude mít pouze jinou barvu).

Dále si popíšme jednotlivé fce:

bool SetIndexBuffer(int index, double array[])

Je funkce, pomocí které danému poli přiřadíme index. *Index* může nabývat hodnot 1 až 8.

void SetIndexStyle(int index, int type, int style, int width, color clr)

Touto funkcí přiřadíme danému indexu typ čáry. V prvním případě jsme použili `DRAW_LINE`, představující klasickou čáru. V druhém případě používáme `DRAW_HISTOGRAM`, který vytváří mezi pomyslnými křivkami histogram.

Styl čáry máme nastavený na `STYLE_SOLID`, tedy nepřerušovanou čáru. V druhém případě na `EMPTY`, protože styl čáry je možné nastavovat pouze pro typ `DRAW_LINE`.

Parametr *width* určujeme tloušťku čáry a pomocí *color* barvu čáry.

Parametr *style* a *width* můžou mít vlastnost `EMPTY`. Ostatní parametry musí být vždy nastavené.

void SetIndexShift(int index, int shift)

Používá se pro posun vykreslovaných dat doprava o velikost *shift*. *Index* označuje buffer, který jsme si na začátku nastavili.

void SetIndexDrawBegin(int index, int begin)

Nastavuje počet svíček od kterých má indikátor začít vykreslovat graf. Počet svíček se udává od začátku grafu (ne od konce). *Index* označuje pole, které obsahuje data pro vykreslení do grafu např. 1 pro *Buffer1[]*. Můžeme totiž chtít vykreslit graf až od určitého místa, protože graf indikátoru před tímto místem nedává vhodné výsledky např. protože má nedostatek dat k výpočtu.

```
48  switch(maMethod){
49      case 0:
50          nazevMetody = "SMA"; break;
51      case 1:
52          nazevMetody = "EMA"; break;
53      case 2:
54          nazevMetody = "SMMA";break;
```



```
55     default :
56         maMethod = 0;
57         nazevMetody = "SMA";
58     }
59
60     switch(maMethod2){
61         case 0:
62             nazevMetody2 = "SMA"; break;
63         case 1:
64             nazevMetody2 = "EMA"; break;
65         case 2:
66             nazevMetody2 = "SMMA"; break;
67         default :
68             maMethod2 = 0;
69             nazevMetody2 = "SMA";
70     }
```

Zde jsme použili operátory switch. Operátor switch převezme parametr metody klouzavého průměru, který si může uživatel v nastavení indikátoru nastavit. Pomocí operátoru switch si uložíme do proměnné *nazevMetody* a *nazevMetody2* řetězec odpovídající volbě uživatele.

```
71 if(maPeriod2 == 0){
72     IndicatorShortName(nazevMetody + "(" + maPeriod + ")");
73     Comment("\n" + nazevMetody + "(" + maPeriod + ")");
74 }else if(maPeriod == 0){
75     IndicatorShortName(nazevMetody2 + "(" + maPeriod2 + ")");
76     Comment("\n" + nazevMetody2 + "(" + maPeriod2 + ")");
77 }else{
78     IndicatorShortName(nazevMetody + "(" + maPeriod + ")" +
79         nazevMetody2 + "(" + maPeriod2 + ")");
80     Comment("\n" + nazevMetody + "(" + maPeriod + ")" +
81         nazevMetody2 + "(" + maPeriod2 + ")");
82 }
83
84 return(0);
```

Pokud máme nastavenou pouze jednu z period *maPeriod*, nebo *maPeriod2* bude vypadat výstup např. SMA(13), kde 13 označuje zvolenou periodu. Pokud budou nastavené obě periody bude výstup např. SMA(8)SMMA(30).

void IndicatorShortName(string name)

Slouží k nastavení jména indikátoru, který se zobrazuje v kontextové nabídce indikátoru nebo v *Okně údajů*. Pro řetězení se používá symbol „+“ (plus).

void Comment(...)

Komentáře se zobrazují v grafu vlevo nahoře. V tomto případě informuje uživatele o volbě metod a period pro klouzavý průměr (klouzavé průměry). Pro řetězení se používá symbol „+“ (plus) nebo „,“ (čárka). Tato funkce může zobrazit pouze 64 parametrů, tedy 64 slov oddělených mezerou.

return(0)

Označuje skončení funkce *init()*.

7.3 Funkce start()

Funkce *start()* se spouští při každém cenovém pohybu, v okamžiku kdy je indikátor přidán do grafu, nebo při otevření klientského terminálu. Jedná se o hlavní část programu, který vykonává samotné výpočty.

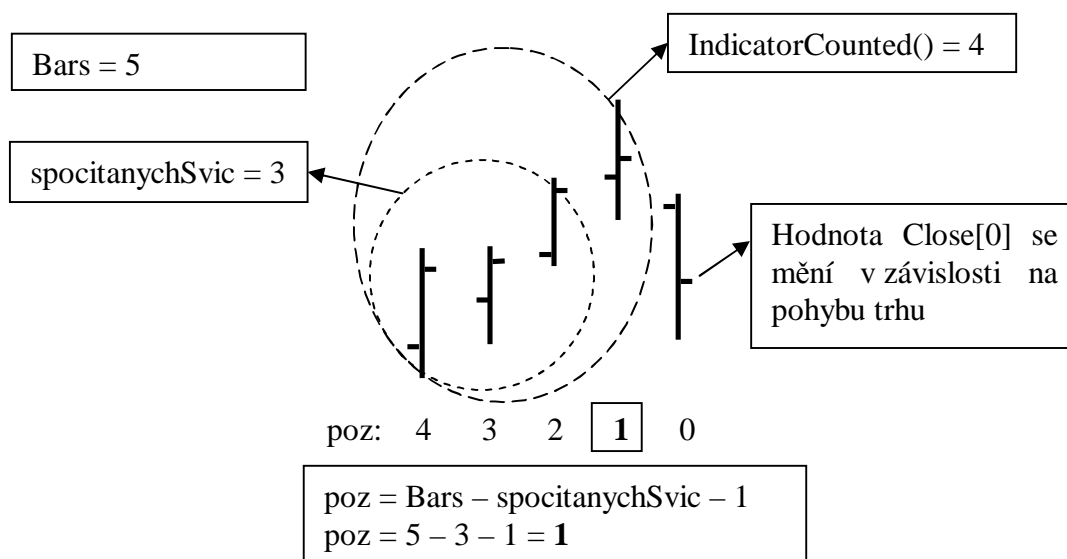
```
83     spocitanychSvic = IndicatorCounted();
84     if(Bars <= 10) return(0);
85
86     if (spocitanychSvic < 0) return(-1);
87
88     if(spocitanychSvic > 0) spocitanychSvic--;
89
90     poz = Bars-spocitanychSvic-1;
91     poz2 = poz;
```

int IndicatorCounted()

Vrací počet svíček, které byly ve výpočtech použity. Tuto hodnotu si uložíme do proměnné *spocitanychSvic*. Pomocí této funkce kontrolujeme, zda není spočítaných svíček méně než nula. V takovém případě by se jednalo o chybu a v *return* bychom vrátili -1 viz. *if(spocitanychSvic < 0) return(-1)*.

V případě, že program už svíčky propočítal, odečteme od tohoto počtu jedničku a tím si zajistíme, že při dalším vstupu programu do funkce `start()`, bude poslední svíčka znovu přepočítána viz. `if(spocitanychSvic > 0) spocitanychSvic--`.

Do proměnné `poz`, což je proměnná indexu pod kterým budeme ukládat data do pole, uložíme tuto hodnotu: `Bars-spocitanychSvic-1`.



Obr. 32: Výpočet pozice pro přepočet poslední uzavřené svíčky

Z obrázku plyne, že budeme přepočítávat poslední uzavřenou svíčku s pozicí `poz = 1`, protože svíčka s pozicí `poz = 0` je aktuální svíčka, která není uzavřená (hodnota `Close` na pozici `poz = 0`, tedy `Close[0]` se neustále mění podle vývoje trhu).

int Bars

Jedná se o předdefinovanou proměnnou vracující počet svíček, které aktuální graf instrumentu obsahuje. V našem případě provádíme tento test: `if(Bars <= 30) return(0)`. Pokud je počet svíček menší nebo rovno 30, tak výpočet ukončíme, protože je k výpočtu velmi málo dat.

```

92 if(maPeriod != 0){
93     switch(maMethod){
94         case 0:
95             sma(); break;
96         case 1:
97             ema(); break;
98         case 2:
99             smma(); break;

```

```
100     default:
101         sma();
102     }
103 }
104 if(maPeriod2 != 0){
105     switch(maMethod2){
106         case 0:
107             sma2(); break;
108         case 1:
109             ema2(); break;
110         case 2:
111             smma2(); break;
112         default:
113             sma2();
114     }
115 }
116 return(0);
```

Opět využijeme operátor switch, kde podle volby metody klouzavého průměru voláme funkce, ve kterých probíhají výpočty hodnot klouzavého průměru.

7.4 Funkce pro výpočet klouzavých průměrů

Nejedná se už o funkce, které jsou nutné pro samotný chod programu. Jedná se o funkce, kterých si může programátor vytvořit libovolný počet.

```
117 void sma(){           //Simple Moving Average
118     soucet = 0;
119     int i;
120
121     if(poz < maPeriod)
122         poz = maPeriod;
123
124     for(i = 1; i < maPeriod; i++, poz--){
125         soucet += Close[poz];
126     }
127
128     while(poz >= 0){
129         soucet += Close[poz];
130         Buffer1[poz] = soucet/maPeriod;
```

```

131         soucet -= Close[maPeriod+poz-1];
132         poz--;
133     }
134 }

```

Algoritmus jednoduchého klouzavého průměru (SMA) je velmi jednoduchý a je dán následujícími vzorci:

$$SMA = \frac{h_m + h_{m-1} + h_{m-2} + \dots + h_{m-n}}{n} \quad (1)$$

$$SMA_{k+1} = SMA_k - \frac{h_{m-n}}{n} + \frac{h_m}{n} \quad (2)$$

SMA Simple Moving Average

h..... představuje určitou hodnotu (low, high, open, close atd.)

n..... perioda (určuje období se kterým SMA počítáme)

```

135 void ema() { //Exponential Moving Average
136     double vyFaktor = 2.0/(maPeriod+1);
137
138     if(spcitanychSvic < 2)
139         poz = Bars-2;
140
141     while(poz >= 0)
142     {
143         if(poz == (Bars-2)) {
144             Buffer1[poz+1] = Close[poz+1];
145         }
146         Buffer1[poz] = Close[poz]*vyFaktor + Buffer1[poz+1]*(1-
vyFaktor);
147         poz--;
148     }
149 }

```

Výpočet:

$$EMA_{k+1} = EMA_k + a(h_k - EMA_k) \quad (3)$$

Po úpravě:

$$EMA_{k+1} = a \cdot h_k + EMA_k(1-a) \quad (4)$$

Vyhlažovací faktor určíme jako:

$$a = \frac{2}{n+1} \quad (5)$$

EMA..... Exponential Moving Average

α vyhlazovací faktor z intervalu 0 až 1

h představuje určitou hodnotu (low, high, open, close atd.)

```

150 void smma(){           //Smoothed Moving Average
151     int i;
152
153     while(poz >= 0){
154         if(poz > Bars-maPeriod){
155             for(i = 0;i < maPeriod;i++,poz--)
156                 soucet += Close[poz];
157         }else{
158             soucet = Buffer1[poz+1]*(maPeriod-1)+Close[poz];
159         }
160         Buffer1[poz] = soucet/maPeriod;
161         poz--;
162     }
163 }
```

Výpočet:

$$SMMA = \frac{h_m + h_{m-1} + h_{m-2} + \dots + h_{m-n}}{n} \quad (6)$$

$$SMMA_{k+1} = SMMA_k - \frac{SMMA_k + h_k}{n} \quad (7)$$

SMMA..... Smoothed Moving Average

h představuje určitou hodnotu (low, high, open, close atd.)

n perioda (určuje období se kterým SMMA počítáme)

8 EXPERT ADVISOR

Jako obchodní strategii jsem vytvořil strategii, která bude využívat klouzavých průměrů. K výpočtu klouzavých průměrů použijeme funkce technických ukazatelů, které jsou v jazyku MQL obsaženy.

8.1 Definice parametrů preprocesoru a externích proměnných

```
1 #property copyright "Copyright © 2011, Petr Kostelanský"
2 #property link      ""
3 #include <WinUser32.mqh>          // knihovna pro funkce MessageBox()
4
5 extern bool upozorneni = true;
6 extern double pocetLotu = 0.1;
7 extern int posunSL = 20;
8 extern int SL = 20;
9 extern int TP = 0;
10 extern int timeFrame = 15;
11 extern int maPeriod1 = 10;
12 extern int maShift1 = 0;
13 extern int maPeriod2 = 30;
14 extern int maShift2 = 6;
15 extern int maMethod = 2;
16
17 double casNakup, casProdej;
18 int i;
```

O preprocesorech jsem se zmiňoval u Custom Indicatoru. Zmíním pouze preprocesor *#include*. Používá se pro vkládání knihovny do našeho programu. V našem případě se jedná o knihovnu *WinUser32.mqh*, která je nutná pro práci s dialogovými okny (pro funkci *MessageBox()*).

Dále následuje seznam externích proměnných označených klíčovým slovem **extern**, pomocí kterých si uživatel danou strategii nastaví podle svých potřeb.

Po externích proměnných následují globální proměnné jako *casNakup*, *casProdej*. Jejich význam si vysvětlíme později.

8.2 Funkce init()

Funkce init() se spouští pouze v okamžiku, kdy strategii přidáme na graf.

```

19     double stopLevel = MarketInfo(Symbol(),MODE_STOPLEVEL);
20     double minLot = MarketInfo(Symbol(),MODE_MINLOT);
21     string nazevMetody;
22
23     if(SL==0){
24         SL = 100;
25         MessageBox("Není nastavený StopLoss!!!\nStopLoss bude nastaven
na "+SL+" bodů ", "Nastavení StopLoss", MB_OK|MB_ICONEXCLAMATION);
26     }else if(SL<stopLevel){
27         SL = stopLevel;
28         MessageBox("StopLoss musí mít odstup minimálně
"+DoubleToStr(stopLevel,0)+" bodů\nStopLoss bude nastaven na
"+DoubleToStr(stopLevel,0)+" bodů ", "Nastavení StopLoss",
MB_OK|MB_ICONEXCLAMATION);
29     }
30
31     if(pocetLotu < minLot){
32         pocetLotu = minLot;
33         MessageBox("Minimální velikost lotu je a bude nastavena na
"+DoubleToStr(pocetLotu,1)+" lotů ", "Nastavení lotů",
MB_OK|MB_ICONEXCLAMATION);
34     }

```

double MarketInfo(string symbol, int type)

Jedná se funkci, která vrací informace o obchodovaném instrumentu v závislosti na volbě typu *type*. Pro typ `MODE_STOPLEVEL` vrací nejmenší možný odstup SL od nabídky (popř. poptávky) a pro `MODE_MINLOT` vrací nejmenší velikost lotu, který je možné obchodovat.

Dále provádíme základní porovnání, zda uživatel nastavil korektní údaje. V opačném případě vyvoláme dialogové okno pomocí *MessageBox()*.

int MessageBox(string text=NULL, string caption=NULL, int flags=EMPTY)

Parametry:

- **Text** obsahuje zprávu, kterou chceme uživateli sdělit. Můžeme ho nastavit na NULL, tedy bez zprávy anebo vložit prázdný řetězec "".
- **Caption** vkládá řetězec do záhlaví. Řetězec může zůstat prázdný.

- **Flags** slouží k nastavení zobrazovaných tlačítek, kde v závislosti na stisknutí tlačítka „OK“, „CANCEL“, atd., vrací kód typu `int`, jehož otestováním zjistíme jakou volbu uživatel zvolil. Součástí *flags* může být i nastavení obrázku u dialogového okna např. volba `MB_ICONEXCLAMATION` zobrazující upozornění.



Obr. 33: Dialogové okno vyvolané funkcí
MessageBox()

string DoubleToStr(double value, int digits)

Převádí číslo typu `double` na textový řetězec. Parametr *value* obsahuje převáděné číslo a parametr *digits* počet desetinných míst, které se mají zobrazovat.

```

35 switch(maMethod)
36     {
37         case 0:
38             nazevMetody = "SMA"; break;
39         case 1:
40             nazevMetody = "EMA"; break;
41         case 2:
42             nazevMetody = "SMMA"; break;
43
44         default :
45             maMethod = 0;
46             nazevMetody = "SMA";
47     }
48
49     Comment("\nNastavení:",
50         "\n-----",
51         "\nLotů: ",pocetLotu,
52         "\nSpread: ",MarketInfo(Symbol(),MODE_SPREAD),
53         "\nStopLoss: ",SL,
54         "\nTakeProfit: ",TP,
55         "\n-----",

```

```

56         "\nMA1 perioda: ",maPeriod1,
57         "\nMA1 posunutí: ",maShift1,
58         "\nMA2 perioda: ",maPeriod2,
59         "\nMA2 posunutí: ",maShift2,
60         "\nMA metoda: ",nazevMetody
61     );
62
63     return(0);
64 }

```

Pomocí operátoru switch testujeme, kterou volbu uživatel zadal, abychom mu jeho volbu mohli zobrazit na grafu.

void Comment(...)

Funkce *Comment()* zobrazuje na graf textový řetězec s maximálním možným počtem parametrů 64. Pro řetězení se používá symbol „+“ (plus) nebo „,“ (čárka).

Výstup této funkce vypadá následovně:

```

Nastavení:
-----
Lotů: 0.1
Spread: 2
StopLoss: 20
TakeProfit: 0
-----
MA1 perioda: 10
MA1 posunutí 0
MA2 perioda: 30
MA2 posunutí 6
MA metoda: SMMA

```

Obr. 34: Výstup funkce

Comment() u EA

return(0)

Ukončuje funkci *init()* a vrací návratovou hodnotu.

8.3 Funkce start()

V EA je funkce volána v okamžiku obdržení nové hodnoty obchodovaného instrumentu, není tedy volána při vložení strategie do grafu.

```

65     int pocetByk = 0;
66     int pocetMedved = 0;
67     bool podNakup, podProdej;

```

```
68
69     double ma1_1 =
       iMA(0,timeFrame,maPeriod1,maShift1,maMethod,PRICE_CLOSE,1);
70     double ma1_3 =
       iMA(0,timeFrame,maPeriod1,maShift1,maMethod,PRICE_CLOSE,3);
71     double ma2_1 =
       iMA(0,timeFrame,maPeriod2,maShift2,maMethod,PRICE_CLOSE,1);
72     double ma2_3 =
       iMA(0,timeFrame,maPeriod2,maShift2,maMethod,PRICE_CLOSE,3);
73
74     podNakup = ma2_3>ma1_3 && ma2_1<ma1_1;
75     podProdej = ma2_3<ma1_3 && ma2_1>ma1_1;
```

double iMA(string symbol, int timeframe, int period, int ma_shift, int ma_method, int applied_price, int shift)

Jedná se o funkce technického indikátoru, které jsou v jazyku MQL4 již vytvořeny. Na základě zadaných parametrů vrací tato funkce hodnotu klouzavého průměru (Moving Average).

Parametry:

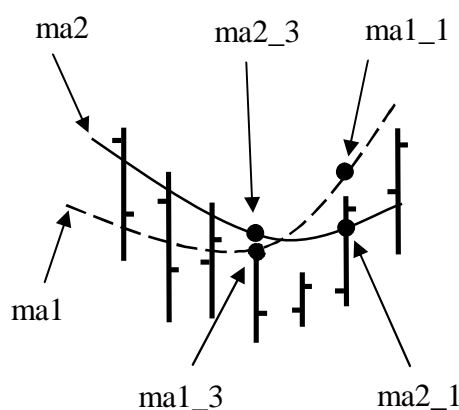
- **Symbol** slouží k nastavení instrumentu na který má být tato funkce použita, např. hodnota 0 (nula) nebo NULL označuje, že chceme tuto funkci použít pro výpočet klouzavého průměru instrumentu, na jehož grafu byla strategie aplikována, tedy na aktuální instrument. Pokud bychom chtěli hodnotu klouzavého průměru na páru EURUSD, zadáme jako parametr "EURUSD" (včetně uvozovek, jedná se o řetězec typu string).
- **Timeframe** označuje časový rámec grafu, z něhož má být hodnota klouzavého průměru získána např. z M5 (pětiminutového grafu), H1 (hodinového grafu).
- **Period** slouží k nastavení období, které má být pro výpočet klouzavého průměru použito.
- **Ma_shift** slouží k nastavení posunu vypočítané křivky doprava.
- **Ma_method**, tedy metody pro výpočet klouzavého průměru např. SMA, EMA, atd. Metodu nastavíme buď pomocí jednoho z klíčových slov MODE_SMA, MODE_EMA, nebo celočíselné hodnoty 0 (odpovídá metodě SMA), 1 (odpovídá metodě EMA).

- **Applied_price** označuje hodnotu, která se má použít pro klouzavý průměr např. uzavírací cena (CLOSE), otevírací cena (OPEN), nejvyšší cena (HIGH) atd. Hodnotu nastavíme klíčovým slovem PRICE_CLOSE, PRICE_HIGH atd., nebo celočíselnou hodnotou 0 (pro hodnotu CLOSE), 2 (pro hodnotu HIGH).
- **Shift** určuje, kterou hodnotu klouzavého průměru chceme získat. Parametr zadáváme jako celočíselnou hodnotu např. 0 (nula) označující, že chceme hodnotu klouzavého průměru pro aktuální svíčku. Hodnota 5 říká, že chceme pátou hodnotu od aktuální svíčky, tedy hodnotu odpovídající páté svíčce od aktuální.

podNakup = ma2_3 > ma1_3 && ma2_1 < ma1_1

Uvedený příkaz testuje, zda se do proměnné *podNakup* (podmínka pro nákup) vloží hodnotu TRUE, nebo FALSE.

Testování překřížení:



Obr. 35: Podmínka k nákupu u dvou klouzavých průměrů

podProdej = ma2_3 < ma1_3 && ma2_1 > ma1_1

Do proměnné *podProdej* (podmínka pro prodej) se vloží hodnota TRUE, nebo FALSE podle toho, jestli je splněna podmínka k prodeji.

```

76 for(i=0;i<OrdersTotal();i++){
77     OrderSelect(i, SELECT_BY_POS, MODE_TRADES);
78     if(OrderType() == OP_BUY)
79         pocetByk++;
80     if(OrderType() == OP_SELL)
81         pocetMedved++;

```

```
82     }
```

Pomocí cyklu `for` prohledáváme otevřené pozice a testujeme u nich o jaký typ objednávky se jedná. Podle typu objednávky inkrementujeme jednu z proměnných *pocetByk*, *pocetMedved*. Tyto proměnné nám budou určovat, zda je už otevřená long (nákupní), nebo short (prodejní) pozice. Tím se vyhneme situaci, kdy by se nám ve velmi krátkém okamžiku otevřelo např. 100 long pozic.

bool OrderSelect(int index, int select, int pool=MODE_TRADES)

Parametry:

- **Index** označuje index objednávky z množiny pozic.
- **Select** je parametr, který může nabývat dvou hodnot:

`SELECT_BY_POS` – budeme vybírat objednávku podle indexu z množiny pozic,

`SELECT_BY_TICKET` – budeme vybírat objednávku podle specifického čísla objednávky (každá objednávka má vlastní číslo).

- **Pool** označuje množinu pozic, ze které chceme vybírat. Použije se pouze pro volbu `SELECT_BY_POS`. Možnosti parametru *pool* jsou následující:

`MODE_TRADES` – jedná se o výchozí možnost. Tato množina obsahuje otevřené nebo čekající objednávky.

`MODE_HISTORY` – množina obsahující zavřené nebo zrušené pozice.

int OrderType()

Objednávku nejprve vybereme pomocí funkce *OrderSelect()* a následně můžeme určit typ objednávky pomocí funkce *OrderType()*.

```
83 if(podNakup && (pocetByk==0) && casNakup!=Time[0]){
84     if(AccountFreeMarginCheck(Symbol(),OP_BUY,pocetLotu)<=0 ||
      GetLastError()==134)
85         Print("Není dostatek finančních prostředků pro tuto
      objednávku");
86     else
87         Objednavka("koupit");    //koupit
88 }
89 if(podProdej && (pocetMedved==0) && casProdej!=Time[0]){
90     if(AccountFreeMarginCheck(Symbol(),OP_SELL,pocetLotu)<=0 ||
      GetLastError()==134)
```

```

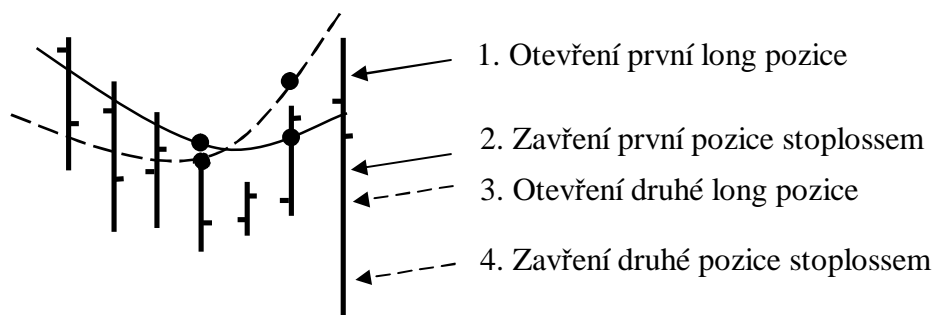
91         Print("Není dostatek finančních prostředků pro tuto
objednávku");
92     else
93         Objednavka("prodat");    //prodat
94     }

```

Máme zde dvě podmínky *if*, které mají totožnou strukturu a řeší jestli vstoupit do pozice long (nákup), nebo short (prodej). Jednu z nich si pojďme rozebrat.

podNakup && (pocetByk==0) && casNakup!=Time[0]

Argument podmínky *if* testuje, zda je proměnná *podNakup* rovna TRUE (musí být splněna podmínka, daná překřížením klouzavých průměrů). Proměnná *pocetByk* musí být na hodnotě 0 (v tomto okamžiku nesmí být otevřená jiná long pozice) a proměnná *casNakup* (obsahuje čas otevření ceny svíčky, kdy byla pozice otevřena, ale taky nemusela být otevřena) se nesmí rovnat času otevření ceny svíčky, ve které byla pozice otevřena a to z následujícího důvodu:



Obr. 36: Obrázek chybného vstupu do pozice long

double AccountFreeMarginCheck(string symbol, int cmd, double volume)

Tato funkce spočítá, zda bude na účtu dostatek prostředků k otevření pozice.

Parametry:

- **Symbol** určuje na jaký instrument se funkce použije. Funkce *Symbol()* vrací aktuální symbol (symbol, na kterém je strategie použita).
- **Cmd** může nabývat hodnot OP_BUY (pro pozici long) a OP_SELL (pro pozici short).
- **Volume** označuje počet lotů, které chceme obchodovat.

Pokud je na účtu dostatek finančních prostředků volá se funkce *Objednavka()*. Jedná se vlastní funkci o které bude řeč v další podkapitole.

```
95 if(OrdersTotal() != 0){
96     UpravitObjednavku();
97 }
```

OrdersTotal()

Vrací počet otevřených pozic. Pokud jsou některé pozice otevřené, voláme funkci *UpravitObjednavku()*, která posunuje SL.

```
98 if(podNakup || podProdej){
99     for(i=0;i<OrdersTotal();i++){
100         OrderSelect(i, SELECT_BY_POS, MODE_TRADES);
101
102         if(podProdej){
103             if(OrderType()==OP_BUY) // long pozice je otevřena
104                 OrderClose(OrderTicket(),OrderLots(),Bid,3,Green);
105         }
106
107         if(podNakup){
108             if(OrderType()==OP_SELL) // short pozice je otevřena
109                 OrderClose(OrderTicket(),OrderLots(),Ask,3,Red);
110         }
111     }
112 }
113 return(0);
```

Pokud dojde k překřížení klouzavých průměrů, znamená to také, že by se měla pozice uzavřít. Pomocí funkce *OrderSelect()* vybíráme otevřené pozice a testujeme, jestli by se neměly uzavřít. Uzavření pozice se provede funkcí *OrderClose()*.

bool OrderClose(int ticket, double lots, double price, int slippage, color Color=CLR_NONE)

Parametry:

- **Ticket** označuje specifické číslo objednávky, kterým jsou objednávky odlišeny.
- **Lots** označuje počet lotů uzavírané pozice.

- **Price** označuje cenu, za kterou má být pozice uzavřena. *Price* může obsahovat proměnnou *Bid*, pokud chceme vystoupit z long (nákupní) pozice, nebo *Ask* pro vystoupení z short (prodejní) pozice.
- **Slippage** označuje skluz od uzavírací ceny, za kterou může být pozice ještě uzavřena např. pokud by v okamžiku uzavírání pozice došlo k cenovému pohybu.
- **Color** je barva, kterou bude vyplněn uzavírací symbol zobrazený na grafu.

int OrderTicket()

Vrací číslo aktuální objednávky.

double OrderLots()

Vrací počet lotů aktuální objednávky.

double Bid

Předdefinovaná proměnná, jejíž hodnota je rovna aktuální nabídce na trhu.

double Ask

Předdefinovaná proměnná, jejíž hodnota je rovna aktuální poptávce na trhu.

8.4 Vlastní funkce

```
114 void Objednavka(string typObjednavka){
115     int ticket = 0;
116     double takeProfit;
117
118     RefreshRates();
119     takeProfit = TakeProfit(typObjednavka, TP);
120
121     if(typObjednavka == "koupit"){
122         ticket = OrderSend(Symbol(), OP_BUY, pocetLotu, Ask, 3, Ask-
            SL*Point, takeProfit, "Objednávka 1", 16384, 0, Green);
123         if(ticket > 0)
124             casNakup = Time[0];
125     }
126
127     if(typObjednavka == "prodat"){
```



```

128         ticket =
OrderSend(Symbol(),OP_SELL,pocetLotu,Bid,3,Bid+SL*Point,takeProfit,"
Objednávka 1",16384,0,Red);
129         if(ticket > 0)
130             casProdej = Time[0];
131     }
132
133     if(ticket > 0){
134         if(OrderSelect(ticket,SELECT_BY_TICKET,MODE_TRADES)){
135             if (upozorneni==true){
136                 Alert("Objednávka ",ticket,
137                     "\nStoploss: ", OrderStopLoss(),
138                     "\nTakeProfit: ", OrderTakeProfit(),
139                     "\nOtevírací cena: ", OrderOpenPrice()
140                 );
141             }
142         }else{
143             Print("Error, nebyla nalezena aktuální objednávka :
",GetLastError());
144             return(0);
145         }
146     }else{
147         Print("Error, otevreni pozice se nezdarilo :
",GetLastError());
148         return(0);
149     }
150 }

```

void Objednavka(string typObjednavka)

Jedná se o vytvořenou funkci, která přebírá hodnotu parametru *typObjednavka*. Na základě hodnoty "koupit", nebo "prodat" se vstoupí do long, nebo short pozice.

bool RefreshRates()

Slouží k znovu načtení dat z grafu, abychom získali aktuální hodnotu *Bid*, nebo *Ask* a mohlo dojít k otevření pozice. Používá se tam, kde se před funkcí pro otevření pozice nachází dlouhé výpočty, které by mohly program zpozdít.

double TakeProfit(string typObjednavka, int TP)

Tato funkce určí hodnotu takeprofit na základě dvou parametrů. *typObjednavka* určuje, jestli se jedná o nákup, nebo prodej. Parametr *TP* nese informaci o počtu bodů (point), které určují vzdálenost takeprofitu od aktuální nabídky, nebo poptávky.

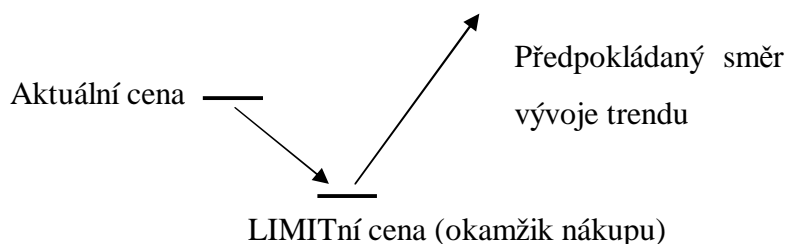
int OrderSend(string symbol, int cmd, double volume, double price, int slippage, double stoploss, double takeprofit, string comment=NULL, int magic=0, datetime expiration=0, color arrow_color=CLR_NONE)

Funkce se používá k uzavírání obchodů. Její návratovou hodnotou je číslo ticketu (číslo provedené objednávky). Pokud se objednávka neprovede vrací -1.

Parametry *slippage*, *arrow_color* (*color*) jsme už zmiňovali u funkce *OrderClose()* a mají stejný význam jako zde.

- **Symbol** určuje na jaký instrument se funkce použije. Funkce *Symbol()* vrací aktuální symbol (symbol na kterém je strategie použita).
- **Cmd** může nabývat hodnot OP_BUY (pro pozici long) a OP_SELL (pro pozici short). Následující typy se většinou používají v kombinaci s datem expirace, aby mohl být čekající pokyn (pending order) zrušen. Jedná se tedy o typy, kterými se tvoří čekající pokyny. Čekajícím pokynem je objednávka buď uzavřena, nebo vyprší doba expirace a čekající pokyn je zrušen.

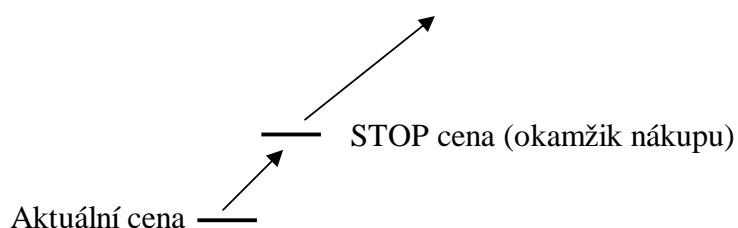
OP_BUYLIMIT – objednávka je umístěná na limitní cenu za kterou chceme nakoupit. Předpokládáte, že cena klesne na limitní cenu a směr se otočí do byčího trendu.



Obr. 37: Objednávka BUY LIMIT

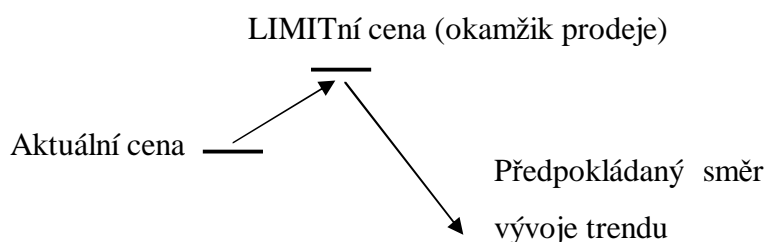
OP_BUYSTOP – objednávka, kterou umístíme nad aktuální cenu trhu.

Předpokládaný směr
vývoje trendu



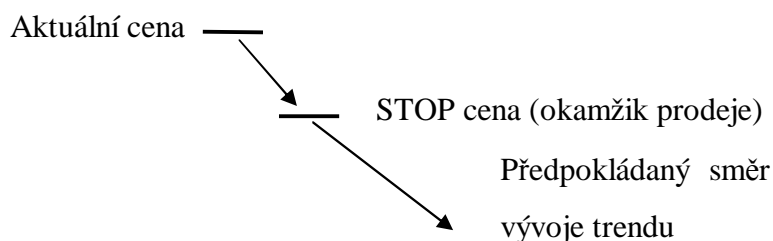
Obr. 38: Objednávka BUY STOP

OP_SELLLIMIT – snažíme se prodat za cenu vyšší než aktuální cena trhu.



Obr. 39: Objednávka SELL LIMIT

OP_SELLSTOP – objednávka na prodej je umístěná pod aktuální cenu.



Obr. 40: Objednávka SELL STOP

- **Volume** označuje počet lotů, které chceme obchodovat.
- **Price** může obsahovat proměnnou *Bid* v případě, že chceme vstoupit do short (prodejní) pozice, nebo *Ask* pokud vstupujeme do long (nákupní) pozice.
- **Stoploss** určuje cenu na kterou nastavíme SL. V případě nákupu nastavíme SL o 20 bodů pod hodnotu *Ask* např. $Ask - 20 * Point$, kde předdefinovaná proměnná *double Point* vrací hodnotu jednoho bodu.
- **Takeprofit** určuje zisk, po jehož dosažení chceme pozici uzavřít.
- **Comment** vyjadřuje komentář k objednávce.

- **Magic** představuje specifický identifikátor (číslo).
- **Expiration** určuje čas, kdy vyprší čekající pokyn. Hodnota expirace může mít následující tvar: $TimeCurrent() + PERIOD_M5 * 60 * 5$ a znamená, že pokyn bude čekat pět svíček od uzavření čekajícího pokynu na pětiminutovém grafu. Expirace se používá pro typy objednávek `OP_BUYLIMIT`, `OP_SELLLIMIT`, `OP_BUYSTOP`, `OP_SELLSTOP`.

V programu dále testujeme hodnotu ticketu. Pokud byla objednávka uzavřena, je hodnota ticketu kladné číslo a vypíšeme informace o objednávce pomocí fce *Alert()*. Dojde-li při objednávce k chybě, bude hodnota ticketu -1. Chybu si necháme vytisknout do *Deníku* funkcí *Print()* za pomoci fce *GetLastError()*.

int GetLastError()

Tato funkce vrací číslo poslední chyby. Informaci o chybě najde v oficiální dokumentaci (<http://docs.mql4.com/constants/errors>).

```
151 double TakeProfit(string typObjednavka, int TP){
152     if(TP!=0){
153         if(typObjednavka == "koupit")
154             return(Ask+TP*Point);
155         else
156             return(Bid-TP*Point);
157     }else
158         return(0);
159 }
```

double TakeProfit(string typObjednavka, int TP)

Funkce vrací hodnotu na kterou se takeprofit nastaví. Parametry funkce jsou *typObjednavka*, který říká pro jaký typ objednávky bude takeprofit vypočítán a *TP*, udávající vzdálenost takeprofit od otevírací ceny v bodech.

```
160 void UpravitObjednavku(){
161     RefreshRates();
162
163     for(i=0;i<OrdersTotal();i++){
164         OrderSelect(i, SELECT_BY_POS, MODE_TRADES);
165
166         if(OrderType() == OP_BUY){
```

```
167         if((Bid-OrderOpenPrice())>Point*posunSL){
168             if(OrderStopLoss()<Bid-Point*posunSL){
169                 OrderModify(OrderTicket(), OrderOpenPrice(), Bid-
posunSL*Point, OrderTakeProfit(), 0, Blue);
170             }
171         }
172     }
173
174     if(OrderType() == OP_SELL){
175         if((OrderOpenPrice()-Ask)>Point*posunSL){
176             if(OrderStopLoss()>Ask+Point*posunSL){
177                 OrderModify(OrderTicket(), OrderOpenPrice(),
Ask+posunSL*Point, OrderTakeProfit(), 0, Blue);
178             }
179         }
180     }
181
182     }
183 }
```

void UpravitObjednavku()

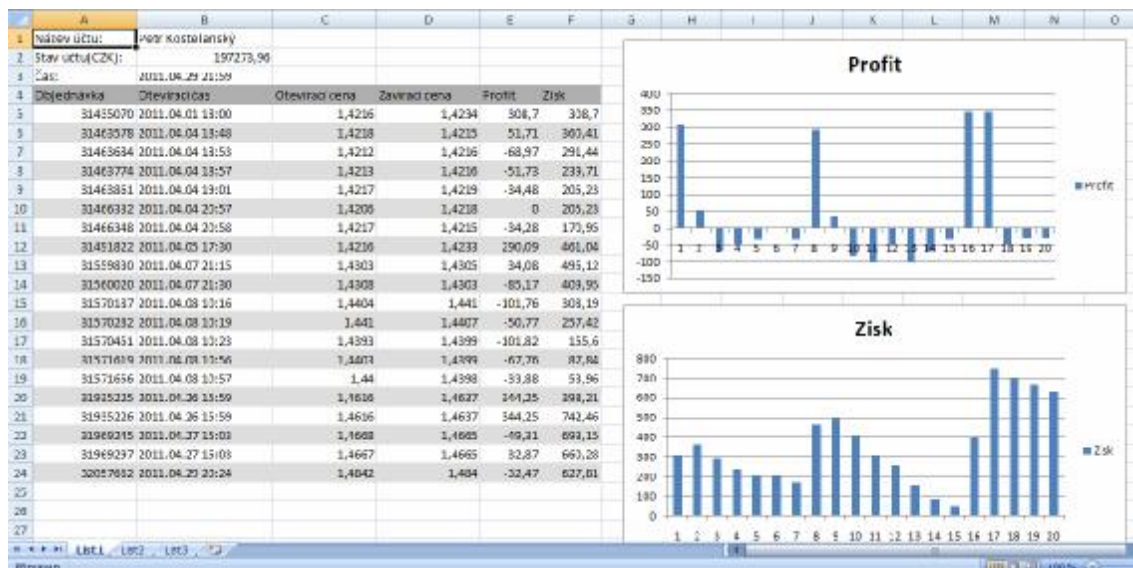
Slouží k modifikaci objednávky a to především k posunu SL. Upravuje objednávky long i short pozic.

bool OrderModify(int ticket, double price, double stoploss, double takeprofit, datetime expiration, color arrow_color=CLR_NONE)

Funkce se v parametrech podobá funkci *OrderSend()*. Jedním z nejdůležitějších parametru je zde *stoploss*, kterým posunujeme SL u provedené objednávky. Jako parametr *ticket* nastavíme *OrderTicket()*, který vrátí číslo vybrané objednávky. Parametr *price* je nastaven na *OrderOpenPrice()*, vracející otevírací cenu vybrané objednávky. Parametr *takeprofit* nastavíme na *OrderTakeProfit()*, který vrátí nastavený TP.

9 SCRIPT

Jako script jsem vytvořil program, prohledávající objednávky, které byly uzavřené v historii. Za použití knihovny *mt4excel.dll*, umožňující práci s Excelem, uložíme informace o objednávkách do Excelu a necháme vykreslit graf.



Obr. 41: Výstup ze scriptu

9.1 Definice parametrů preprocesoru a funkcí knihovny

```

1 #property copyright "Copyright © 2011, Petr Kostelanský"
2 #property link      ""
3 #include <WinUser32.mqh>
4 #import "mt4excel.dll"
5
6 bool ExcelOpen();
7 bool ExcelClose();
8 bool ExcelSaveAs(string FileName);
9 bool ExcelSetDiagramma(int TypeD,string Title,string XRange,string
   YRange,int Left,int Top,int Riht,int Bottom);
10 bool ExcelSetTextCell(int X,int Y,string Value);
11 bool ExcelSetRange(string Range);
12 bool ExcelRangeColumnWidth(int Width);
13 bool ExcelRangeInteriorColor(int Color);
14 string ExcelGetLastErrorText();

```

U tohoto programu jsme využili preprocesor *#import* pro načtení knihovny. Knihovna musí být umístěna ve složce *\experts\libraries*.

Dále jsme museli definovat funkce z knihovny *mt4excel.dll*, které budeme využívat.

9.2 Funkce start()

```
15 if (ExcelOpen())
16     Print("Excel byl úspěšně otevřen");
17 else
18     Print("Nepodařilo se otevřít Excel:",ExcelGetLastErrorText());
```

bool ExcelOpen()

Funkce slouží k otevření programu Excel. Funkce je datového typu boolean, proto testujeme jestli byl Excel otevřen.

```
19 ExcelSetTextCell(4,1,"Objednávka");
20 ExcelSetTextCell(4,2,"Otevírací čas");
21 ExcelSetTextCell(4,3,"Otevírací cena");
22 ExcelSetTextCell(4,4,"Zavírací cena");
23 ExcelSetTextCell(4,5,"Profit");
24 ExcelSetTextCell(4,6,"Zisk");
```

bool ExcelSetTextCell(int X,int Y,string Value)

Funkce nastavuje textový řetězec, na souřadnice dané parametry X (představuje čísla řádků) a Y (čísla sloupců).

```
25 ExcelSetRange("A1");
26 ExcelRangeColumnWidth(15);
27 ExcelSetRange("B1");
28 ExcelRangeColumnWidth(20);
29 ExcelSetRange("C1");
30 ExcelRangeColumnWidth(15);
31 ExcelSetRange("D1");
32 ExcelRangeColumnWidth(15);
33 ExcelSetRange("A4:F4");
34 ExcelRangeInteriorColor(0xAFAFAF);
```

bool ExcelSetRange(string Range)

Do parametru této funkce vložíme rozsah buněk, pro které budeme nastavovat další vlastnosti. Tyto vlastnosti nastavíme pomocí jiných funkcí.

bool ExcelRangeColumnWidth(int Width)

Pro vybraný rozsah buněk nastavíme šířku sloupců.

bool ExcelRangeInteriorColor(int Color)

Slouží k nastavení barvy pozadí pro rozsah buněk, určený funkcí *ExcelSetRange("A4:F4")*.

```
35     int xSouradnice,i;
36     double zisk = 0;
37     string konecRozsahu;
38     string zacatekRozsahu;
39     string rozsah;
40
41     for(i=0;i<OrdersHistoryTotal() ;i++){
42         OrderSelect(i, SELECT_BY_POS, MODE_HISTORY);
43         xSouradnice = i+5;
44         zisk += OrderProfit();
45         if(xSouradnice%2==0){
46             rozsah = StringConcatenate("A", xSouradnice, ":F",
xSouradnice);
47             ExcelSetRange(rozsah);
48             ExcelRangeInteriorColor(0xDEDEDE);
49         }
50         ExcelSetTextCell(xSouradnice,1,OrderTicket());
51         ExcelSetTextCell(xSouradnice,2,TimeToStr(OrderOpenTime()));
52         ExcelSetTextCell(xSouradnice,3,OrderOpenPrice());
53         ExcelSetTextCell(xSouradnice,4,OrderClosePrice());
54         ExcelSetTextCell(xSouradnice,5,OrderProfit());
55         ExcelSetTextCell(xSouradnice,6,zisk);
56         if(i==0)
57             int zacXSouradnice = xSouradnice-1;
58         int konecXSouradnice = xSouradnice;
59     }
```

Cyklem for budeme procházet množinu objednávek, které byly zadány v minulosti. Funkci *OrderSelect(i, SELECT_BY_POS, MODE_HISTORY)* budeme jednotlivé objednávky vybírat a budeme u nich získávat informace jako:

- *OrderTicket()* – číslo objednávky.
- *OrderOpenTime()* – otevírací čas objednávky.
- *OrderOpenPrice()* – kurz za kterou jsme pozici otevřeli.
- *OrderClosePrice()* – kurz za kterou jsme pozici zavřeli.

- OrderProfit() – zisk (ztráta) z objednávky.

Funkce *OrderSelect()* je vysvětlena u programu Expert Advisor.

string StringConcatenate(...)

Funkce slouží ke spojování řetězců. Vytvoříme řetězec (např. A5:F5 – ten bude představovat rozsah buněk, pro který budeme nastavovat odlišnou barvu pozadí), který bude představovat vstupní parametr do funkce *ExcelSetRange()*. Funkcí *ExcelRangeInteriorColor()* nastavíme barvu pozadí. Abychom tak nastavili každý sudý řádek, vytvoříme jednoduchou podmínku *if(xSouradnice%2==0)*, pro porovnávání zbytku po celočíselném dělení.

```
60     string rozsahE = StringConcatenate("E", zacXSouradnice, ":E",  
    konecXSouradnice);  
61     string rozsahF = StringConcatenate("F", zacXSouradnice, ":F",  
    konecXSouradnice);  
62  
63     ExcelSetDiagramma(60,"Profit","E4:E5",rozsahE,470,10,400,200);  
64     ExcelSetDiagramma(65,"Zisk","F4:F5",rozsahF,470,220,400,200);
```

Do proměnných *rozsahE* a *rozsahF* si uložíme řetězce, představující sloupce, které vykreslíme do grafů.

```
65     ExcelSetTextCell(1,1,"Název účtu:");  
66     ExcelSetTextCell(1,2,AccountName());  
67  
68     ExcelSetTextCell(2,1,"Stav účtu(CZK):");  
69     ExcelSetTextCell(2,2,DoubleToStr(AccountFreeMargin(),2));  
70  
71     ExcelSetTextCell(3,1,"Čas:");  
72     ExcelSetTextCell(3,2,TimeToStr(TimeCurrent()));  
73  
74     int ret = MessageBox("Zavřít Excel?", "",  
    MB_YESNO|MB_ICONQUESTION);  
75     if(ret==IDYES){  
76         ExcelSaveAs("c:\\stav_uctu.xls");  
77         ExcelClose();  
78     }  
79  
80     return(0);
```

Na závěr zobrazíme uživateli dialogové okno s dotazem, jestli se má Excel zavřít. Vlastnosti funkce *MessageBox()* je popsán u programu Expert Advisor. Výsledek po kliknutí na jedno z tlačítek *Ano*, *Ne* si uložíme do proměnné *ret* a následně ho porovnáme s možností IDYES (uživatel kliknul na *Ano*).

bool ExcelSaveAs(string FileName)

Pokud uživatel klikne na tlačítko *Ano*, uložíme soubor do kořene disku pod názvem *stav_uctu.xls*.

bool ExcelClose()

Na závěr soubor uzavřeme.

ZÁVĚR

Hlavním cílem bylo seznámit se s online obchodováním a se softwarem, který tyto možnosti poskytuje. V dnešní době existuje řada programů, poskytující online data z trhů. My jsme se seznámili s platformou MetaTrader 4, který poskytuje řada brokerských společností a je velmi rozšířen. Jeho největší výhodou je zabudovaný jazyk, umožňující tvorbu vlastních programů.

V první části jsme se seznámili s obchodní platformou a uvedli jsme si její přednosti a možnosti využití v praxi. Praktická část práce obsahuje tři programy: skript, ukazatel a program automatizující obchodní transakce. Hlavním cílem bylo ukázat si funkcionality těchto programů a možnosti, které nám jazyk MLQ nabízí. Podrobně jsme si popsali použití programů a také možnost testování obchodní strategie.

Mezi nejdůležitější vlastnosti této obchodní platformy patří možnost vytvořit si automatický obchodní systém, který by automatizoval celý obchodní proces včetně rozhodování a uzavírání obchodu. Díky tomu by už nebyla třeba neustálé fyzické přítomnosti obchodníka u počítače. Je třeba mít na paměti, že k vytvoření úspěšného automatického obchodního systému je třeba spousta zkušeností a znalostí. A i v případě vytvoření takového systému, je třeba provádět neustálé modifikace, tak aby mohl správně reagovat na měnící se trh.

ZÁVĚR V ANGLIČTINĚ

The main objective was to learn about online businesses and about software that provides these options. There are many programs today that provide online data from the markets. We familiarized ourselves with the MetaTrader 4 platform, which provides a number of brokerage companies and which is very widespread. Its biggest advantage is the built-in language that allows a creation of custom programs.

In the first part, we acquainted with the trading platform and we introduced its advantages and possibilities of practical use. The theoretical part contains three programs: the script, an indicator and a program automatizing business transactions. Our main purpose was to demonstrate the functionality of these programs and options that MLQ language offers to us. We described in the detail the application of the programs and the possibility of testing of trading strategies.

One of the most important features of this trading platform is the ability to create an automatic trading system that automatizes the entire business process including decision-making and business closure. Thanks this property the continuous physical presence of the trader by the computer would not be necessary. It should be kept in the mind that for creating of a successful automatic trading system is lot of experience and knowledge necessary. And even in case of creating of such a system it is necessary to carry out continuous modifications, so that it can properly respond to the market changing for ever.

SEZNAM POUŽITÉ LITERATURY

- [1] *Financnik.cz - komodity, akcie, burza, forex* [online]. c2005 [cit. 2011-05-01]. Dostupné z WWW: <<http://www.financnik.cz/>>.
- [2] *Burza-knihy.cz* [online]. c2008 [cit. 2011-05-23]. Burza - jak na obchodování. Dostupné z WWW: <<http://www.burza-knihy.cz/burza-fotografie.php>>.
- [3] Contract for difference#CFDs compared to other products. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, , last modified on 19 April 2011 [cit. 2011-05-01]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Contract_for_difference#CFDs_compared_to_other_products>.
- [4] Futures#Futures. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, , last modified on 1. 4. 2011 [cit. 2011-05-01]. Dostupné z WWW: <<http://cs.wikipedia.org/wiki/Futures#Futures>>.
- [5] Forex. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, , last modified on 11. 4. 2011 [cit. 2011-05-01]. Dostupné z WWW: <<http://cs.wikipedia.org/wiki/Forex>>.
- [6] *Metaquotes.net* [online]. c2000 [cit. 2011-05-23]. MetaTrader 4 Trading Platform. Dostupné z WWW: <<http://www.metaquotes.net/en/metatrader4>>.
- [7] *X-Trade Brokers* [online]. 2002 [cit. 2011-05-01]. Dostupné z WWW: <<http://www.xtb.cz/>>.
- [8] *MQL4 - automated forex trading, custom indicators and strategy tester* [online]. c2000 [cit. 2011-05-01]. Dostupné z WWW: <<http://www.mql4.com/>>.
- [9] *Fxstreet.cz* [online]. c2009 [cit. 2011-05-23]. Srovnání a hodnocení FOREX brokerů. Dostupné z WWW: <<http://www.fxstreet.cz/srovnani-a-hodnoceni-brokeru.html>>.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

CFD	Contract For Difference
FOREX	Foreign Exchange
MT4	MetaTrader 4
MQL4	MetaQuotes Language 4
EA	Expert Advisor
CI	Custom Indicator
FA	Fundamentální analýza
TA	Technická analýza
SL	Stoploss
TP	Takeprofit

SEZNAM OBRÁZKŮ

Obr. 1: Obrázek z burzy CBOT [2].....	11
Obr. 2: Makléřské služby poskytované platformou MetaTrader 4, [6]	14
Obr. 3: Klientský terminál.....	15
Obr. 4: Prostředí MetaEditoru	17
Obr. 5: Prostředí MetaEditoru, panel nabídek	17
Obr. 6: Prostředí MetaEditoru, panel nástrojů.....	18
Obr. 7: Prostředí MetaEditoru, okno Navigator	18
Obr. 8: Prostředí MetaEditoru, okno Toolboxu, záložka Help	19
Obr. 9: Prostředí MetaEditoru, okno Toolboxu, záložka Online Library.....	19
Obr. 10: Dialogové okno Založit účet, Osobní údaje	31
Obr. 11: Dialogové okno Založit účet, Servery	32
Obr. 12: Dialogové okno Založit účet, Registrace.....	33
Obr. 13: Ukázka vložení indikátoru z roletové nabídky Vložit.....	34
Obr. 14: Ukázka vložení indikátoru z panelu nástrojů	35
Obr. 15: Okno Navigátor.....	35
Obr. 16: Dialogové okno pro nastavení parametrů indikátoru.....	36
Obr. 17: Dialogové okno pro obecné nastavení aplikované strategie	36
Obr. 18: Dialogové okno Alarm.....	37
Obr. 19: Dialogové okno pro potvrzení obchodu u spuštěné strategie	38
Obr. 20: Panel nástrojů Standard s tlačítkem „Zapnout strategie“.....	38
Obr. 21: Symboly označující stavy strategie	38
Obr. 22: Kontextová nabídka pro změnu vlastností nebo odstranění strategie z grafu	39
Obr. 23: Okno Tester, záložka Nastavení.....	40
Obr. 24: Okno Tester, záložka Výsledky.....	41
Obr. 25: Okno Tester, záložka Graf.....	41
Obr. 26: Okno Tester, záložka Report	42
Obr. 27: Okno Tester, záložka Deník.....	42
Obr. 28: Okno pro nastavení parametrů testované strategie, záložka Vložit parametry	43
Obr. 29: Okno pro nastavení parametrů testované strategie, záložka Optimalizace	43
Obr. 30: Okno Tester, záložka Výsledek optimalizace	44
Obr. 31: Okno Tester, záložka Graf optimalizace.....	44

Obr. 32: Výpočet pozice pro přepočet poslední uzavřené svíčky	51
Obr. 33: Dialogové okno vyvolané funkcí MessageBox()	57
Obr. 34: Výstup funkce Comment() u EA	58
Obr. 35: Podmínka k nákupu u dvou klouzavých průměrů	60
Obr. 36: Obrázek chybného vstupu do pozice long	62
Obr. 37: Objednávka BUY LIMIT	66
Obr. 38: Objednávka BUY STOP	67
Obr. 39: Objednávka SELL LIMIT	67
Obr. 40: Objednávka SELL STOP	67
Obr. 41: Výstup ze scriptu	70

SEZNAM TABULEK

Tab. 1: Rozdíly online a parketového obchodování.....	11
Tab. 2: Tabulka některých Forex brokerů [9].....	14

SEZNAM PŘÍLOH

P I CD-ROM s textem práce a zdrojovými soubory