MQL4 COURSE

By Coders' guru

www.forex-tsd.com

-15-

Váš první Expert Advisor
Část 3

V předcházejících dvou částech této lekce jsme uvedli našeho expert advisora a prozkoumali jeho podstatu..
V příloze (***Appendix 2***) jsme prostudovali *Trading Functions* (obchodní funkce), které se nyní naučíme používat.
Dnes budeme také pokračovat v louskání zbývající části programového kódu našeho expert advisora.
Doufám tedy, že jste s čistou myslí připraveni na naši objevitelskou misi.

Náš stávající programový kód:

```
//+------------------------------------------------------------------+
//|                                                  My_First_EA.mq4 |
//|                                                      Coders Guru |
//|                                          http://www.forex-tsd.com |
//+------------------------------------------------------------------+
#property copyright "Coders Guru"
#property link http://www.forex-tsd.com

//---- vstupní parametry
extern double TakeProfit=250.0;
extern double Lots=0.1;
extern double TrailingStop=35.0;
//+------------------------------------------------------------------+
//| inicializace experta                                             |
//+------------------------------------------------------------------+
int init()
  {
//----
//----
return(0);
  }
//+------------------------------------------------------------------+
//| deinicializace experta                                           |
//+------------------------------------------------------------------+
int deinit()
  {
//----
//----
return(0);
  }
int Crossed (double line1 , double line2)
  {
static int last_direction = 0;
static int current_direction = 0;
```

```
if(line1>line2)current_direction = 1;     //nahoru
if(line1<line2)current_direction = 2;     //dolů
if(current_direction != last_direction)   //změna
  {
last_direction = current_direction;
return (last_direction);
  }
else
  {
return (0);
  }
  }
//+------------------------------------------------------------------+
//| funkce start experta                                             |
//+------------------------------------------------------------------+
int start()
  {
//----
   int cnt, ticket, total;
   double shortEma, longEma;

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

   shortEma = iMA(NULL,0,8,0,MODE_EMA,PRICE_CLOSE,0);
   longEma = iMA(NULL,0,13,0,MODE_EMA,PRICE_CLOSE,0);

   int isCrossed = Crossed (shortEma,longEma);

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

ticket=OrderSend(Symbol(),OP_BUY,Lots,Ask,3,0,Ask+TakeProfit*Point,
"My EA",12345,0,Green);
      if(ticket>0)
        {
         if(OrderSelect(ticket,SELECT_BY_TICKET,MODE_TRADES))
Print("BUY order opened : ",OrderOpenPrice());
        }
      else Print("Error opening BUY order : ",GetLastError());
      return(0);
        }
      if(isCrossed == 2)
        {
```

```
            ticket=OrderSend(Symbol(),OP_SELL,Lots,Bid,3,0,
Bid-TakeProfit*Point,"My EA",12345,0,Red);
            if(ticket>0)
              {
               if(OrderSelect(ticket,SELECT_BY_TICKET,MODE_TRADES))
Print("SELL order opened : ",OrderOpenPrice());
              }
            else Print("Error opening SELL order : ",GetLastError());
            return(0);
          }
         return(0);
       }
   for(cnt=0;cnt<total;cnt++)
       {
       OrderSelect(cnt, SELECT_BY_POS, MODE_TRADES);
       if(OrderType()<=OP_SELL &&  OrderSymbol()==Symbol())
         {
         if(OrderType()==OP_BUY)
// otevření pozice long
         {
// můžeme uzavřít?
         if(isCrossed == 2)
         {
         OrderClose(OrderTicket(),OrderLots(),Bid,3,Violet);
// uzavřít pozici
          return(0); // konec
         }
// prověřujeme trailing (posuvný) stop
       if(TrailingStop>0)
         {
          if(Bid-OrderOpenPrice()>Point*TrailingStop)
            {
             if(OrderStopLoss()<Bid-Point*TrailingStop)
               {
                OrderModify(OrderTicket(),OrderOpenPrice(),Bid-
Point*TrailingStop,OrderTakeProfit(),0,Green);
                return(0);
               }
             }
           }
         }
       else
// otevření pozice short
         {
// můžeme uzavřít?
          if(isCrossed == 1)
         {
          OrderClose(OrderTicket(),OrderLots(),Ask,3,Violet);
// uzavřít pozici
          return(0); // konec
         }
// prověřujeme trailing (posuvný) stop
          if(TrailingStop>0)
            {
             if((OrderOpenPrice()-Ask)>(Point*TrailingStop))
```

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

```
int cnt, ticket, total;
```

Na tomto řádku jsme deklarovali tři proměnné typu *integer* (celočíselné).
Použili jsme jeden řádek pro deklaraci všech tří, protože se jedná o proměnné stejného typu. (Pro proměnné různých typů nelze použít jednořádkovou deklaraci).

Poznámka: deklarace více proměnných na jedné řádce začíná **klíčovým slovem** které určuje typ proměnné za nímž následuje seznam **proměnných**, které jsou odděleny čárkou.

Místo jednořádkového zápisu lze použít pro každou proměnnou samostatný řádek:

```
int cnt;
int ticket;
int total;
```

Proměnnou *cnt* (counter) budeme používat jako čítač v programové smyčce (cyklu) která bude kontrolovat počet příkazů zadaných uvnitř smyčky.
Proměnnou *ticket* budeme používat pro uložení čísla štítku příkazu, které nám vrací funkce *OrderSend*.
Proměnnou *total* budeme používat pro uložení počtu zadaných příkazů.

```
double shortEma, longEma;
```

K deklaraci dvou proměnných typu *double* použit opět jeden řádek.
Tyto proměnné budeme používat pro uložení příslušných hodnot EMA.
Jak si jistě pamatujete z předchozích částí, používáme překřížení krátkého a dlouhého EMA jako podmínku pro vstup do pozice a výstup z pozice.

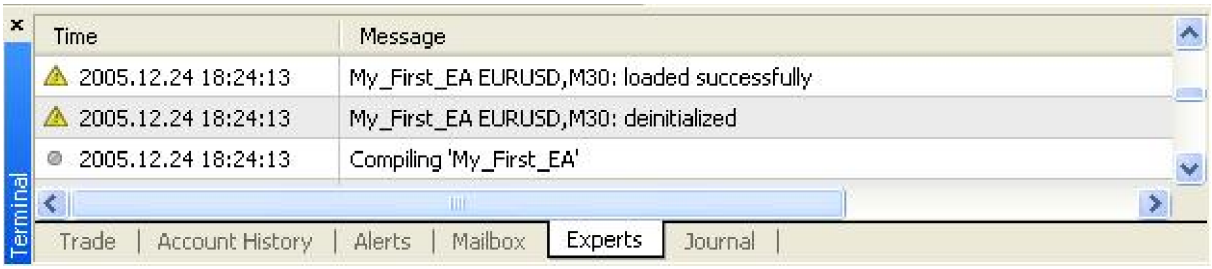
```
if(Bars<100)
  {
   Print("bars less than 100");
   return(0);
  }
```

Chceme pracovat s běžným grafem a proto předpokládáme, že bude mít více než 100 svící. To proto, že při nedostatečném počtu svící by náš EMA indikátor nepracoval správně.

Načetli jsme určitý počet svící pomocí funkce *Bars* a otestovali jsme, zda je jich méně než 100. Jestliže ano, uděláme dvě věci:

Sdělíme uživateli, co není v pořádku výpisem do záznamu Experta (Experts log):
"bars less than 100" (*Obrázek 1*).
Poté ukončíme činnost funkce *start* provedením řádku *retrun(0);*

To je tedy způsob, jak odmítneme pracovat, pokud je svící méně než 100.



Obrázek 1 – Experts log

```
if(TakeProfit<10)
      {
       Print("TakeProfit less than 10");
       return(0); // prověření proměnné TakeProfit
      }
```

Nechceme pracovat s nedostatečnou hodnotou proměnné *TakeProfit*.
TakeProfit je externí proměnná, to umožňuje uživateli měnit její hodnotu v okně **expert advisor properties**. Proto chceme uživatele chránit před zadáním nevhodně zvolených hodnot.
Předpokládáme, že hodnota proměnné *TakeProfit* menší než 10 je nevhodná, proto proměnnou prověřujeme.

Jestliže je hodnota skutečně menší než 10, informujeme uživatele zprávou
"TakeProfit menší než 10", a ukončíme činnost funkce *start* provedením řádku *retrun(0);*

```
shortEma = iMA(NULL,0,8,0,MODE_EMA,PRICE_CLOSE,0);
longEma = iMA(NULL,0,13,0,MODE_EMA,PRICE_CLOSE,0);
```

Tak tedy nyní máme načteno více než 100 svící a hodnota proměnné *TakeProfit* je nastavena na více než 10.

Nyní chceme vědět, jak vypočítat long a short EMA pro aktuální svíci.
Použijeme funkci **iMA**, která je do MQL4 vestavěna a slouží pro výpočet klouzavých průměrů.

Na tomto místě se na chvíli zastavím, abych vysvětlil některé podrobnosti o funkci **iMA**.

---

iMA:

Syntaxe:

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

Popis:

Funkce *iMA* vypočítá indikátor klouzavého průměru a vrátí jeho hodnotu (datový typ `double`).

Poznámka: Klouzavý průměr je průměrná cena určité měny v určitém časovém intervalu (dnů, hodin, minut atd) vyhodnocená pro námi sledované období (např. do podoby grafu).

Parametry:

Tato funkce má 7 parametrů:

`string` **symbol**:
Symbol měnového páru, který obchodujete (příklad: `string` EURUSD `and` USDJPY).
Pokud chceme použít aktuální symbol, zadáme hodnotu parametru NULL.

`int` **timeframe:**
Časový rámec, který chceme použít pro výpočet klouzavého průměru.

MT4 umožňuje použít některou z následujících konstant s těmito hodnotami:

Konstanta	**hodnota**	**popis**
PERIOD_M1	1	1 minuta
PERIOD_M5	5	5 minut
PERIOD_M15	15	15 minut
PERIOD_M30	30	30 minut
PERIOD_H1	60	1 hodina
PERIOD_H4	240	4 hodiny
PERIOD_D1	1440	1 den (kalendářní)
PERIOD_W1	10080	týden
PERIOD_MN1	43200	měsíc (30 dní)
0 (zero)	0	Časový rámec, použitý na aktuálním grafu

Chceme li použít časový rámec aktuálního grafu, použijeme hodnotu parametru 0.

Poznámka: Pro zadání hodnoty periody můžeme použít její číselnou hodnotu nebo název konstaty:

Například, zápis:
```
iMA(NULL,PERIOD_H4,8,0,MODE_EMA,PRICE_CLOSE,0);
```
je ekvivalentní zápisu
```
iMA(NULL,240,8,0,MODE_EMA,PRICE_CLOSE,0);
```
Nicméně pro přehlednost zápisu se doporučuje používat názvy konstant..

`int` **period:**
Počet dní, které chceme použít pro výpočet klouzavého průměru.
`int` **ma_shift:**
Počet svící (posun), o který je graf klouzavého průměru pozadu (napřed) oproti aktuální svíci.

0 znamená žádný posun (*Obrázek 2*)
Kladná hodnota – posun doprava (*Obrázek 3*)
Záporná hodnota – posun doleva (*Obrázek 4*)

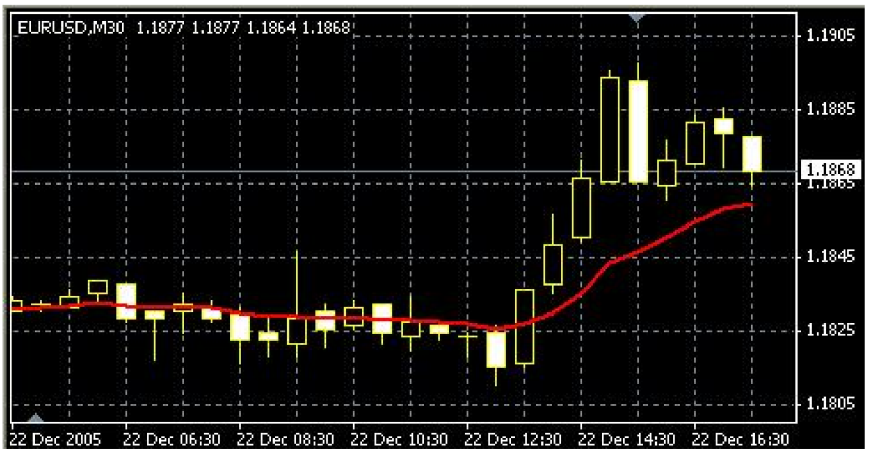
`int` **ma_method**:
Metoda výpočtu klouzavého průměru. Může nabývat následujících hodnot:

Konstanta	Hodnota	Popis
MODE_SMA	0	Simple moving average
MODE_EMA	1	Exponential moving average
MODE_SMMA	2	Smoothed moving average
MODE_LWMA	3	Linear weighted moving average

`int` **applied_price**:
Cena, kterou chceme použít pro výpočet klouzavého průměru. Může nabývat následujících hodnot:

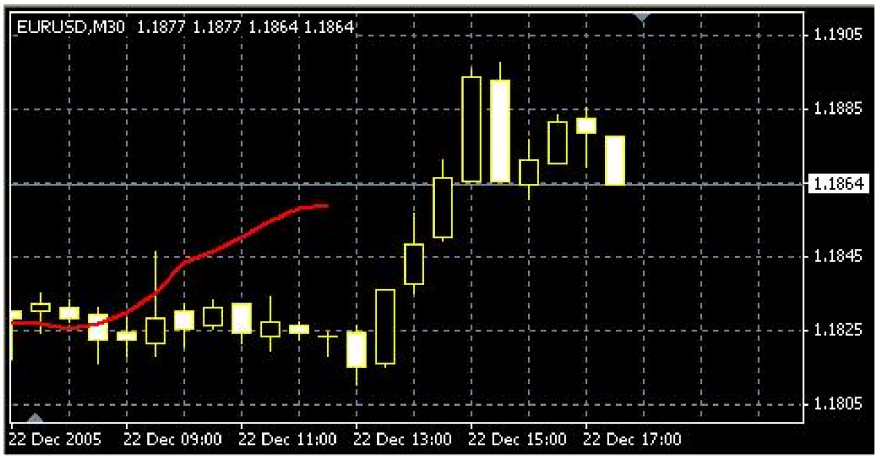
Konstanta	Hodnota	Popis
PRICE_CLOSE	0	Close price.
PRICE_OPEN	1	Open price.
PRICE_HIGH	2	High price.
PRICE_LOW	3	Low price.
PRICE_MEDIAN	4	Median price, (high+low)/2.
PRICE_TYPICAL	5	Typical price, (high+low+close)/3.
PRICE_WEIGHTED	6	Weighted close price, (high+low+close+close)/4.

`int` **shift**:
Počet svící (vzhledem k aktuální svíci) který chceme použít k výpočtu klouzavého průměru. Pro aktuální svíci použijeme hodnotu **0**.



Obrázek 2 : ma_ shift = 0

Obrázek 3: ma_shift = 10



Obrázek 4: ma_shift = -10

```
shortEma = iMA(NULL,0,8,0,MODE_EMA,PRICE_CLOSE,0);
longEma = iMA(NULL,0,13,0,MODE_EMA,PRICE_CLOSE,0);
```
Nyní tedy víme, co znamenají tyto dva řádky programového kódu..

Přiřadili jsme proměnné *shortEma* hodnotu:
8 denního exponenciálního klouzavého průměru (EMA) na základě uzavírací (close) ceny aktuální svíce.
Stručně ji můžeme nazývat 8EMA.

A proměnné *longEma* jsme přiřadili hodnotu:
13 denního exponenciálního klouzavého průměru (EMA) na základě uzavírací (close) ceny aktuální svíce.
Stručně ji můžeme nazývat 13EMA.

```
int isCrossed = Crossed (shortEma,longEma);
```

Poznámka: Funkce *Crossed* načítá dvě hodnoty typu `double` jako parametry a vrací hodnotu integer.

První parametr udává aktuální hodnotu první funkce(v našem případě short EMA) a druhý parametr aktuální hodnotu druhé funkce (long EMA).
Funkce *Crossed* monitoruje hodnoty obou vstupních funkcí pokaždé, když ji voláme, a současně ukládá jejich směrnice (slope) do statických proměnných po dobu mezi opakovaným voláním funkce.

Funkce vrací **0** jestliže nenastala změna oproti uloženým hodnotám.

Funkce vrací **1** jestliže se směr změnil (grafy se překřížily) a první je nad druhým

Funkce vrací **2** jestliže se směr změnil (grafy se překřížily) a druhý je nad prvním

Zde tedy deklarujeme proměnnou *isCrossed* pro uložení vrácené hodnoty funkce *Crossed*. Tuto hodnotu budeme používat k otevírání a uzavírání pozic.

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

Hodnotu, vrácenou funkcí *OrdersTotal*, přiřazujeme do proměnné total.

Poznámka: Funkce *OrdersTotal* vrací počet zadaných příkazů, tj. počet otevřených pozic a budoucích (podmíněných) vstupů do pozice (entry, pending order). Jestliže je toto číslo 0, znamená to, že nejsou zadány žádné příkazy.

Podrobněji viz appendix 2

Větev `if` programového kódu je spuštěna pouze tehdy, je-li hodnota total menší než 1, tedy není-li zadán žádný příkaz.

```
if(isCrossed == 1)
      {
ticket=OrderSend(Symbol(),OP_BUY,Lots,Ask,3,0,Ask+TakeProfit*Point,
"My EA",12345,0,Green);
         if(ticket>0)
           {
            if(OrderSelect(ticket,SELECT_BY_TICKET,MODE_TRADES))
Print("BUY order opened : ",OrderOpenPrice());
           }
         else Print("Error opening BUY order : ",GetLastError());
         return(0);
      }
```

V případě, že graf *shortEma* překříží *longEma* a *shortEma* je nad *longEma*, zadáme pokyn pro nákup. Pro zadání příkazu **buy** (nákup) použijeme funkci *OrderSend.*

Poznámka: Funkce *OrderSend* se používá pro zadání příkazu buy/sell (nákup/prodej) nebo k zadání budoucího (podmíněného) vstupu do pozice (entry, pending order).
Funkce vrací číslo štítku (ticket number) zadaného příkazu, pokud byl úspěšně proveden, v případě chyby vrací hodnotu **-1**.

Syntaxe:

`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)

Parametry:

symbol:
funkce *Sybmol* načítá název aktuální měny a předává jej funkci *OrderSend*.
cmd:
typ příkazu, v našem případě *OP_BUY* , protože chceme otevřít pozici buy (nákup).
volume:
We used the *Lots* value that the use has been supplied.
price:
Používáme funkci *Ask* k načtení aktuální *Ask* price (nákupní ceny) a předání funkci OrderSend
slippage:
Nastavujeme povolenou hodnotu *slippage* na **3** (pips, ticks, points).
stoploss:
Hodnotu *stoploss* nastavujeme na **0**, to znamená žádný *stoploss.*
takeprofit:
Násobíme hodnotu *TakeProfit* , zadanou uživatelem, hodnotou funkce *Point* a výsledek přičítáme k hodnotě *Ask* price (nákupní ceny).

Poznámka: funkce *Point* vrací velikost 1 point (pip, tick) aktuální měny
Příklad: obchodujeme-li EURUSD hodnota 1 point = .0001 ,obchodujeme-li EURJPY hodnota 1 point by měla být .01
To je tedy důvod, proč je nutné převádět hodnoty *stoploss* a *takeprofit* na points dříve, než je použijeme uvnitř funkcí *OrderSend* nebo *OrderModify*.

comment:
komentář, v našem programovém kódu řetězec *"My EA"*
magic:
v našem programovém kódu číslo *12345* pro příkaz *magic number*.
expiration:
datum ukončení platnosti příkazu, v našem případě **0**.
arrow_color:
barvu šipky indikátoru máme *Green* (Protože máme rádi peníze a peníze jsou zelené)

Funkce *OrderSend* vrací číslo štítku příkazu (pokud je úspěšná), možný výskyt chyby testujeme pomocí následujícího příkazu:

```
if(ticket>0)
```

Použili jsme funkci *OrderSelect* k výběru příkazu dle čísla štítku a to před použitím funkce *OrderOpenPrice*, která vrací otevírací cenu vybraného příkazu. Vše je OK, funkce *OrderSend* vrátila správné číslo štítku (větší než 0) a funkce *OrderSelect* úspěšně vybrala příkaz. A to příležitost sdělit tuto dobrou správu uživateli odesláním zprávy *"BUY order opened : "* a zobrazením ceny v otevřené pozici.

Poznámka: Více podrobností o funkcích *OrderSelect* a *OrderOpenPrice* viz appendix 2

Pokud funkce *OrderSend* vrátí **-1** , to znamená že nastala chyba při provádění příkazu, musíme tuto špatnou zprávu oznámit uživateli odesláním zprávy: *"Error opening BUY order : "*, a zobrazením čísla chyby vrácené funkcí *GetLastError*.
A v tomto případě musíme ukončit funkci start příkazem *return(0).*

```
if(isCrossed == 2)
      {
         ticket=OrderSend(Symbol(),OP_SELL,Lots,Bid,3,0,
Bid-TakeProfit*Point,"My EA",12345,0,Red);
          if(ticket>0)
            {
             if(OrderSelect(ticket,SELECT_BY_TICKET,MODE_TRADES))
Print("SELL order opened : ",OrderOpenPrice());
            }
          else Print("Error opening SELL order : ",GetLastError());
          return(0);
      }
```

Zde je opačný scénař, kdy *shortEma* překříží *longEma* směrem dolů.
Použijeme funkci *OrderSend* k zadání příkazu Sell (prodej). Rozdíl mezi parametry Buy (nákup) a Sell (prodej) funkce *OrderSend* jistě již tušíte.

Tyto parametry se nemění:
symbol
volume
slippage
stoploss
comment
magic
expiration

Tyto parametry se mění:
cmd:
Použijeme *OP_SELL* protože chceme otevřít pozici Sell (prodej).
price:
Použijeme funkci *Bid* k načtení *Bid* price (prodejní ceny) a předání hodnoty funkci *OrderSend*

takeprofit:
Násobíme hodnotu *TakeProfit* , zadanou uživatelem, hodnotou funkce *Point* a výsledek odečítáme od hodnoty *Bid* price (prodejní ceny).
arrow_color:
barvu šipky indikátoru máme *Red* (Protože máme rádi peníze a peníze jsou zelené, ale potřebujeme odlišnou barvu pro pozici Sell)

Nyní si můžeme stručně zopakovat co nastane po volání *OrderSend* s výše uvedenými parametry:

Funkce *OrderSend* vrátí číslo štítku. Prověříme, zda je toto číslo větší než **0** (indikuje že nedošlo k chybě). Použili jsme *OrderSelect* k výběru typu příkazu před použitím *OrderOpenPrice*.
Vše je OK, oznámíme tuto dobrou zprávu uživateli odesláním zprávy *"Sell order opened : "* a zobrazením ceny v otevřené pozici.
Pokud funkce *OrderSend* vrátí **-1** oznámíme tuto špatnou zprávu uživateli odesláním zprávy: *"Error opening SELL order : "*, plus číslo chyby a ukončíme funkci *start* provedením příkazu *return(0).*

- Počkejte chvilku! (říkáte).
- Za posledním blokem je ještě jeden řádek.
- Kde? (odpovídám).
- Tady:

```
return(0);
```

Následující úsek kódu ukazuje pozici tohoto příkazu ve struktuře našeho kódu:

```
if(total < 1)
{
if(isCrossed == 1)
{
.....
}
if(isCrossed == 2)
{
.....
}
return(0);  ← zde
}
```

Jestliže *shortEma* překříží *longEma* směrem nahoru zadáme příkaz Buy (nákup).
Jestliže *shortEma* překříží *longEma* směrem dolů zadáme příkaz Sell (prodej).

Pokud se v zadaném časovém rámci nepřekříží, je činnost funkce start ukončena řádkem return(0); (v případě, že *isCrossed* není rovno 1 nebo 2).

Nyní jsme tedy připraveni otevírat pozice Buy (nákup) a Sell (prodej).

V následující lekci přelouskáme zbytek programového kódu.