# MQL4 COURSE

By Coders' guru
www.forex-tsd.com

-11
## Váš první indikátor - část 2

Vítejte ve druhé části lekce **"Váš první indikátor"**.
V předchozí lekci jsme nezapsali žádný řádek s kódem, to proto, že pomocník „New Project Wizard" celý kód zapsal za nás. Děkujeme!

Dnes přidáme několik řádků ke kódu, který pro nás pomocník vygeneroval, aby byl pro nás program užitečnějším.
Poté si celý kód vysvětlíme, řádek po řádku.

Začněme tedy kódovat.

**Kódy, které jsme přidali:**

Kódy, které jsme přidali, jsou vyznačeny **tučně tmavě modře** k původnímu kódu:

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

//|                                                My_First_Indicator.mq4 |
//|                                                        Codersguru |
//|                                             http://www.forex-tsd.com |

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

#property copyright "Codersguru"
#property link "http://www.forex-tsd.com"


#property indicator_separate_window
#property indicator_buffers 1
#property indicator_color1 Red
```

```
//----vyrovnávací paměti

double ExtMapBuffer1[];


//+------------------------------------------------------------------+
//| Custom indicator – inicializační funkce |
//+------------------------------------------------------------------+
int init()

    {
//----indikátory

     SetIndexStyle(0,DRAW_LINE);

     SetIndexBuffer(0,ExtMapBuffer1);

     string short_name = "Your first indicator is running!";

     IndicatorShortName(short_name);

//----

     return(1);

    }
//+------------------------------------------------------------------+
//| Custom indicator – deinicializační funkce |
//+------------------------------------------------------------------+
int deinit()

    {
//----


//----

     return(0);

    }
//+------------------------------------------------------------------+
//| Custom indicator – funkce opakování |
//+------------------------------------------------------------------+
int start()

    {

     int counted_bars=IndicatorCounted();
```

```
//----kontrola možných chyb
    if (counted_bars<0) return(-1);
//----naposled počítaná svíce bude opět přepočítána
    if (counted_bars>0) counted_bars--;

    int pos=Bars-counted_bars;

    double dHigh , dLow , dResult;
    Comment("Hi! I'm here on the main chart windows!");
//----hlavní kalkulační smyčka
    while(pos>=0)
      {
          dHigh = High[pos];
          dLow = Low[pos];
          dResult = dHigh -dLow;
          ExtMapBuffer1[pos]= dResult ;
          pos--;
      }
//----
    return(0);
  }
//+------------------------------------------------------------------+
```

**Jak to bude fungovat?**

Zapíšeme řádek(ky) kódu, který si přejeme vyložit, ty budou vyloženy následovně, pokud se však nevyskytuje jiné téma, vysvětlíme řádky kódu přímo. Většinou se však budeme zastavovat, abychom si vysvětlovali některá obecná témata.

**Ohledně této metody bych rád slyšel vaše připomínky!**

Nyní si vysvětleme tento kód, řádek po řádku.

```
//+------------------------------------------------------------------+
//| My_First_Indicator.mq4 |
//| Codersguru |
//| http://www.forex-tsd.com |
//+------------------------------------------------------------------+
```

**Komentáře:**

Prvních 5 řádků kódu (v šedé barvě) jsou komentáře.
Komentáře používáte při zápisu řádků do vašeho kódu, které bude kompilátor ignorovat. Váš kód komentujete z mnoha důvodů:

- Pro objasnění
- K dokumentaci některých částí, jako např. copyrightu, vytvoření data apod.
- Aby byl srozumitelný.
- K vysvětlení, jak zapsaný kód pracuje.
- …

Komentáře můžete zapisovat dvěma způsoby:

**Jednořádkové komentáře**: Jednořádkový komentář začíná s "//" a končí novým řádkem.

**Víceřádkové komentáře**: Víceřádkové komentáře začínají s "/*" a končí s "*/" a můžete s nimi komentovat na více řádcích.

Do programu MQL4 pomocník vygeneroval z dat, která jsme vložili**, jméno** programu, **autora** a **odkaz** a zapsal je jako komentáře do horní části programu.

```
#property copyright "Codersguru"
#property link "http://www.forex-tsd.com"


#property indicator_separate_window
#property indicator_buffers 1
#property indicator_color1 Red
```

Příkaz Property:

Jak jste zaznamenali, všechny tyto řádky začínají slovem **(#property**). To proto, že jsou druhem **Preprocesorových příkazů** nazývaných **property directives**.

**Preprocesors** jsou instrukce, které dáváte kompilátoru k provedení před spuštěním (zpracováním) vašeho kódu.
**Příkazy property** jsou předdefinované konstanty zvané "**Controlling Compilation**" zabudované

v jazyce MQL4; jejich úkolem je nastavení vlastností vašeho programu. Např.: Vyobrazí se váš indikátor v hlavním okně grafu nebo v okně zvláštním? Kdo je autorem programu?

***Poznámka**:Řádky preprocesorů končí prováděcím znakem return (nový řádek), nikoliv středníkem.*

Zde se pokusíme prodiskutovat příkazy property, které jsou k dispozici v MQL4.

**link:**
Je odkaz na vaše webové stránky, o který jste si požádali v kroku 2
pomocníka Expert Advisoru (nahlédněte do předchozí lekce).
Datový typ této vlastnosti je string.

**copyright:**
Je jméno autora programu, které jste si vyžádali rovněž v kroku 2 pomocníka Expert Advisoru.

Datový typ této vlastnosti je string.

**stacksize:**
Je hodnota celého čísla nastavená v paměti pro každý proces, výchozí hodnota je 16384.
Datový typ této vlastnosti je integer.

**indicator_chart_window:**
Když nastavíte tuto funkci, váš indikátor bude vykreslen v hlavním okně grafu (obr. 1). Můžete si vybrat jednu ze dvou možných voleb pro vyobrazení vašich indikátorů, a to buď v hlavním okně grafu použitím této funkce, nebo vykreslením ve zvláštním okně volbou funkce **the indicator_separate_window**. Nemůžete použít oba dva najednou.
Datový typ této vlastnosti je void, což znamená, že nevrací žádnou hodnotu.

**indicator_separate_window:**
Pokud použijete tuto volbu, indikátor bude vykreslen v odděleném okně (obr.1). Rozměr pro toto okno můžete nastavit pomocí dvou funkcí **indicator_minimum** pro minimální hodnotu a **indicator_maximum** pro maximální hodnotu velikosti.
Úroveň vašich indikátorů můžete nastavit prostřednictvím funkce **indicator_levelN,** kde **N** je číslem indikátoru.
Obě funkce **indicator_chart_window** a **indicator_separate_window** jsou typu void, což znamená, že nevracejí hodnotu a pouze je zapíšete. V našem programu vykreslíme indikátor ve zvláštním okně:
#property indicator_separate_window


*Figure 1*

**indicator_minimum:**

Za pomocí této funkce nastavíme minimální hodnotu velikosti odděleného okna, která se nachází v jeho spodní části. Např.:

*#propery indicator_minimum 0 #propery indicator_maximum 100*

Zde jsme nastavili spodní ohraničení okna na 0 a vrchní ohraničení na hodnotu 100 (viz. indicator_ maximum), pak jsme nastavili hodnoty v rozmezí 0 až 100 v našem odděleném okně, ve kterém vykreslujeme náš indikátor.

Datový typ této funkce je integer.

**indicator_maximum:**

Za pomocí této funkce nastavujeme maximální hodnotu velikosti oddělených oken, tedy jejich vrchní ohraničení. Tato hodnota musí být vyšší než hodnota **indicator_minimum**.

Datový typ této funkce je integer.

**indicator_levelN:**

Za pomocí této funkce nastavíme úroveň rozměru indikátoru, kterou jsme vytvořili ve funkcích **indicator_minimum** a **indicator_maximum**, tato hodnota musí být vyšší než **indicator_minimum** a nižší než **indicator_maximum**.

**N** je číslo, které nastavíme pro úroveň indikátoru, její rozsah musí být v rozmezí 1 až 8 (protože máme k dispozici pro indikátory pouze 8 vyrovnávacích pamětí v programu, takže můžeme nastavit **indicator_level** pro každý z nich použitím jeho čísla).

Např.:

```
#propery indicator_minimum      0
#propery indicator_maximum      100
#propery indicator_level1         10      //nastavení úrovně první vyrovnávací paměti indikátoru
#propery indicator_level2         65.5    //nastavení druhé vyrovnávací paměti indikátoru
```

Datový typ této funkce je double.

**indicator_buffers:**

Za pomocí této funkce nastavujeme počet míst (polí) v paměti, alokovaných pro vytvoření našeho řádku(ů). Při určení tohoto čísla (v rozsahu od **1** do **8**) sdělujeme MQL4: "Prosím o alokování prostoru v paměti pro mě k vykreslení řádku indikátoru".

V našem programu jsme použili pouze jednu paměť.

```
#property indicator_buffers 1
```

To proto, že vykreslujeme pouze jeden řádek.

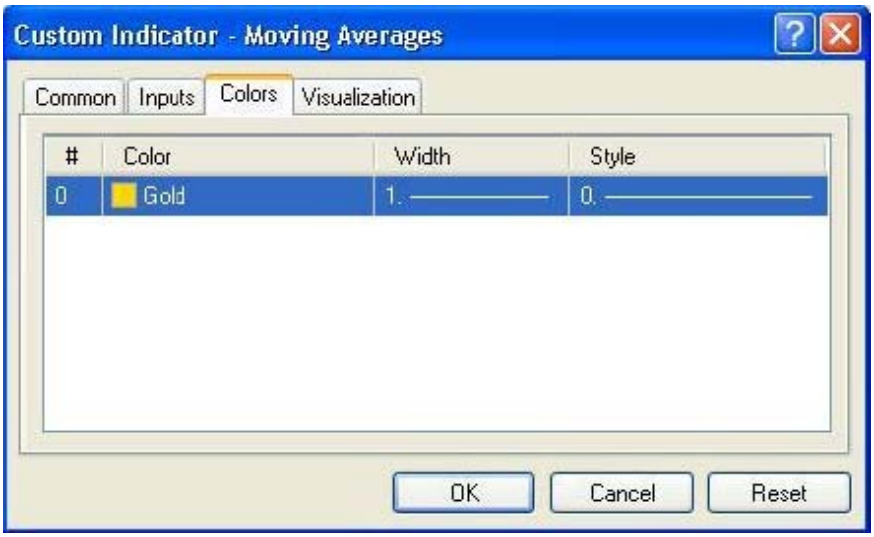
**indicator_colorN:**

V indikátoru můžeme použít až 8 řádků, můžete nastavit barvu každého z nich použitím funkce **indicator_colorN** , přičemž **N** je číslo řádku, které je definováno pamětmi **indicator_buffers**.

Uživatel vašeho indikátoru může měnit barvu z dialogu properties vašeho indikátoru (obr. 2).
V našem programu bude barva řádku indikátoru červená.

```
#property indicator_color1 Red
```

Datový typ této funkce je color.




*Obr. 2*

```
double ExtMapBuffer1[];
```

**Pole:**

Stejné předměty obvykle seskupujeme do jednotek, stejně tak je tomu u programování, kdy také potřebujeme seskupit dohromady data stejného typu. Pro tento účel používáme funkci Array (pole). Pole jsou velmi podobná tabulkám seznamů. Do tabulek seskupujete položky a přiřazujete jim číslo řady. Řady v polích se nazývají indexy.

Pro deklarování pole se používá takovýto kód:
**int my_array[50];**
Zde jste deklarovali pole typu integer, které může obsahovat až 50 položek.

Zpřístupnit každou položku v poli můžete použitím jejího indexu, jak je tomu zde:
**My_array[10] = 500;**
Zde jste vložili položku číslo 10 do pole 500.

Pole můžete inicializovat ve stejném řádku, v jakém bylo provedeno deklarování:

**int my_array[5] = {1,24,15,66,500};**

V našem programu použijeme tento řádek kódu:
`double ExtMapBuffer1[];`

Zde jsme deklarovali pole typu double. Pole použijeme pro výpočet našich hodnot, které vykreslíme v grafu.

```
int init()

{
}
```

**Zvláštní funkce:**

Funkce jsou bloky kódů, které, stejně jako stroje, přijímají vstupy a vrací výstupy (viz. lekce 7 – Funkce). V MQL4 se vyskytují 3 zvláštní funkce

**init():**

Každý program spustí tuto funkci před jakoukoliv další funkcí, do níž vkládáte inicializační hodnoty vašich proměnných.

**start():**

Zde probíhá většina práce. Pokaždé, když je přijata do vašeho programu nová cenová nabídka, je vyvolána tato funkce.

**deinit():**

Tato poslední funkce programu bude vyvolána před jeho uzavřením, můžete zde vložit přesuny, jaké si přejete.

```
SetIndexStyle(0,DRAW_LINE);
SetIndexBuffer(0,ExtMapBuffer1);

string short_name = "váš indikátor běží!";
IndicatorShortName(short_name);
```

**Custom indicator - funkce:**

Nemůžu vám v této lekci popsat funkce všech indikátorů. Podrobněji se jim však budeme věnovat v dalších lekcích. Zde si tedy nastudujeme funkce, týkající se našeho programu.

**SetIndexStyle:**

```
void SetIndexStyle( int index, int type, int style=EMPTY, int width=EMPTY, color clr=CLR_NONE)
```

Tato funkce nastaví střih vykreslení řádku.

**Index** parametr této funkce se pohybuje v rozsahu od 1 do 7 (to proto, že indexování pole začíná hodnotou 0 a je omezeno na 8 řádků). A indikuje, u kterého řádku si přejeme střih nastavit.

**Typ** parametru je typ vzhledu řádku a může odpovídat jedné z následujících typových konstant:
**DRAW_LINE** (vykreslení řádku)
**DRAW_SECTION** (vykreslení sekce)
**DRAW_HISTOGRAM** (vykreslení histogramu)
**DRAW_ARROW** (vykreslení ukazatele)
**DRAW_NONE** (žádné vykreslení)

**Styl** parametru je střih vykreslování řádku a může být jednou z těchto konstant:

**STYLE_SOLID** (použití pevného pera)

**STYLE_DASH** (použití přerušovaných čar)

**STYLE_DOT** (tečkování)

**STYLE_DASHDOT** (čárkování a tečkování)

**STYLE_DASHDOTDOT** (čárkování a dvojité tečkování)

Nebo může být **EMPTY** (výchozí), což znamená, že ve vykreslení nebude provedena žádná změna.

Parametr **width** je šířkou řádku a jeho rozsah se pohybuje v rozsahu hodnot 1 až 5. Případně může být **EMPTY**
(výchozí), takže se šířka nemění.

**Clr** parametr je barva řádku. Může to být jakákoliv proměnná typu color. Výchozí hodnota je **CLR_NONE,** což znamená, že stav barev je prázdný.

V našem řádku kódu:

```
SetIndexStyle(0,DRAW_LINE);
```

Nastavili jsme index na 0, což znamená, že budeme pracovat s prvním řádkem (který je zároveň jediným). Typ vzhledu jsme nastavili jako DRAW_LINE, protože si přejeme vykreslit řádek v grafu. U ostatních parametrů jsme ponechali výchozí hodnoty.

**SetIndexBuffer:**

```
bool SetIndexBuffer( int index, double array[])
```

Tato funkce nastaví pole, kde přiřadíme hodnotu našeho indikátoru k vyrovnávací paměti indikátoru, která bude vykreslena.

Funkce přebere index vyrovnávací paměti, kde 0 je první vyrovnávací pamětí, 1 je druhou atd. Pak si přebere jméno pole.
Pokud funkce uspěje, vrací hodnotu **true,** jinak vrací hodnotu **false**.
V našem programu je pole, které obsahuje vypočtené hodnoty, ExtMapBuffer1.
A máme zde pouze jednu vyrovnávací paměť indikátoru (#property indicator_buffers 1), takže jí bude vyrovnávací paměť přiřazena.

**IndicatorShortName:**

```
void IndicatorShortName( string name)
```

Tato funkce určuje text, který bude vyobrazen v levém horním rohu okna grafu (obr.3). V našem programu jsme deklarovali proměnnou string a přiřadili k ní hodnotu "Váš první indikátor běží", kterou jsme poté předali funkci **IndicatorShortName**.

```
string short_name = "Váš první indikátor běží!";
IndicatorShortName(short_name);
```


*Figure 3*

```
return(0);
```

Toto je vratná hodnota funkce **init(),** která funkci ruší a předává program k provedení další funkce **start().**

```
int deinit()

    {
//----


//----

      return(0);

    }
```

K funkci **deinit()** není nic nového, co by se dalo říct.

Se zbytkem kódů budeme pokračovat v další lekci. Doufám, že se vám lekce líbila a uvítám jakékoliv otázky a připomínky.

S pozdravem

**Coders' Guru**
06-11-2005