Funkce

18. listopadu 2011

Pokud se bude v programu vyskytovat nějaký blok příkazů, který se vykonává často (většinou pro různé hodnoty proměnných), je vhodné ho definovat jako samostatnou funkci. I v případě, že k opakovanému volání nedochází, může použití funkcí sloužit alespoň k uspořádání kódu pro jeho větší přehlednost a čitelnost. Srozumitelné kódy mívají ve funkci main minimum řádků volajících jednotlivé funkce.

Funkce může mít dvě různá uplatnění. Buď může sloužit k výpočtu nějaké hodnoty nebo jen k uzavření nějaké sekvence příkazů jako např. při vypisování vektorů, matic atd. V tomto druhém případě funkce nevrací žádnou hodnotu.

Funkce definujeme vždy v úvodní části zdrojového kódu, mimo hlavní funkci `main`, poté co vložíme knihovny. Prvnímu řádku definice funkce se říká **hlavička funkce** a je vždy složena ze tří částí:

typ_návratové_hodnoty název_funkce (*výčet_parametrů*)

Za hlavičkou poté následuje **tělo funkce**, což je sekvence příkazů uzavřených ve složených závorkách.

Návratová hodnota je hodnota, která se ve funkci vypočítá a vrátí jako výsledek. Pokud definujeme funkci, která nám nemá nic vracet, uvedeme jako typ návratové hodnoty klíčové slovo `void`. V ostatních případech si musíme rozmyslet, jakého datového typu bude výsledek funkce a jako typ návratové hodnoty musíme uvést odpovídající datový typ: `int`, `double` atd. V těle funkce určíme návratovou hodnotu pomocí klíčového slova:

`return` *výraz* `;`

Tento příkaz je vždy posledním příkazem dané funkce. Příkazy uvedené za příkazem return se již neprovedou. V případě funkce bez návratové hodnoty je možné použít příkaz return také:

```
return;
```

a to k předčasnému ukončení funkce.

Název funkce může být libovolný souvislý řetězec znaků bez mezer.

Parametry funkce jsou jinými slovy proměnné dané funkce nebo také vstupní hodnoty funkce, se kterými poté funkce pracuje. V praxi to vypadá tak, že v okamžiku volání funkce si funkce vytvoří nové proměnné, do kterých se zkopírují hodnoty z proměnných, které do ní posíláme. Jinými slovy: do funkce vždy posíláme hodnoty, ne samotné proměnné. V hlavičce funkce jsou její parametry vždy uzavřeny v kulatých závorkách, vzájemně se oddělují čárkou a u každého je vždy nutné znovu uvést příslušný datový typ i v případě, že jsou všechny stejného typu.

Jako názornou ilustraci si ukážeme program na výpočet součtu dvou čísel, kde se ovšem výpočet součtu provede ve funkci. V tomto případě se tedy jedná o funkci s návratovou hodnotou. Volání funkce s návratovou hodnotou už znáte, protože jste ho používali kdykoliv jste potřebovali nějakou funkci z knihovny `math.h`.

```
#include<stdio.h>

double soucet (double b,double c)
{
  double d;
  d=b+c;
  return d;
}

main(void)
{
  double a, b, c;

  printf( "Zadej 1. cislo:\n" );
  scanf( "%lf", &a );

  printf( "Zadej 2. cislo:\n" );
  scanf( "%lf", &b );

  c = soucet(a,b);

  printf( "Soucet cisel %g + %g = %g.\n", a, b, c );

  return 0;
}
```

V uvedeném příkladu si všimněte, že ve funkci `main` i ve funkci `soucet` je použita proměnná `b`. Jelikož do funkce posíláme nejprve hodnotu proměnné `a`, tato hodnota se zkopíruje do prvního parametru funkce, tj. do proměnné `b` funkce `soucet`. Parametry funkce a všechny proměnné, které jsou definovány uvnitř funkce jsou tzv. **lokální proměnné**. To znamená, že existují pouze uvnitř dané funkce a zbytek programu je nezná. Pokud bychom tedy chtěli ve funkci `soucet` použít proměnnou `a`, která je však definovaná ve funkci `main`, překladač nám ohlásí chybu, totiž že daná proměnná je na tomto místě neznámá. Proto ačkoliv v obou funkcích existuje proměnná `b`, jedná se o dvě různé proměnné alokované v různých místech paměti.

Shrnutí některých podstatných detailů:

- V okamžiku volání funkce se alokuje paměť pro nové lokální proměnné, do kterých se pouze zkopíruje hodnota z proměnných, které jsou uvedeny v příkazu volajícím danou funkci.
- Funkce vrací vždy jen jednu hodnotu a to pomocí příkazu `return`. Změnou hodnoty proměnné definované jako parametr funkce se nijak nezmění hodnota proměnné, kterou jsme použili při volání funkce.

Jako příklad funkce bez návratové hodnoty si ukážeme program s funkcí na vypsání prvků řádkového vektoru.

```
#include<stdio.h>

void vypis_vektor (double b[], int n)
{
  int i;
  for (i=0;i<n; i++)
    printf( "%g ", b[i] );
}
```

```
main(void)
{
  double b[] = {4.8, 3.2, 6.4};

  printf( "Vektor b = ( " );
  vypis_vektor( b, 3 );
  printf( ")\n" );

  return 0;
}
```

Všimněte si rozdílného volání funkce bez návratové hodnoty. I tento způsob už ale znáte, např. při volání funkcí `printf()` či `scanf()`. Dále si povšimněte, že při volání funkce není vždy nutné používat jen proměnné, můžeme použít konstanty, jako v tomto případě nebo i složitější výrazy, funkce atd.

Kromě lokálních proměnných existují i **globální proměnné**, které jsou pak známé pro všechny funkce programu. Tyto proměnné se definují vně funkcí, obvykle po vložení knihoven před definicí první funkce. Pro používání globálních proměnných je však nutné mít dobrý důvod. Hodnota těchto proměnných může být změněna v libovolné funkci a ve složitějších programech není lehké v tom udržet pořádek. Studenti mají často sklon k tomu řešit některé algoritmické otázky právě použitím globálních proměnných. Jelikož se ale programovat učíme, budeme se prozatím globálním proměnným vyhýbat.