Funkce v jazyce C

Petrásek Jan

Funkcionální dekompozice

- Jde o rozdělení aplikace na logické celky, kdy každý celek realizuje nějaká (sada) funkce
- Funkcím se jinak také říká podprogramy nebo procedury
- U velkých aplikací sdružujeme funkce do modulů (knihoven)
- Formálně je procedura takový podprogram, který nemá návratovou hodnotu, jen vykoná nějakou akci
- Jako funkci pak označujeme podprogram, který má návratovou hodnotu
- Deklarace funkce: <datový typ> název(<parametry>)
- Definice funkce: <datový typ> název(<parametry>){<tělo>}

Odbočka k proměnným

- Proměnné deklarované uvnitř funkce (včetně main) josou lokální
- Proměnní deklarované nad funkcemi pod direktivami a makry jsou globální
- Globální proměnné jsou přímo dostupné kdekoliv v kódu
- Lokální proměnné jsou dostupné pouze uvnitř funkce
- Pokud je u lokální proměnné užito klíčové slovo static, pak je proměnná i její hodnota uchována po skončení funkce (reálně je proměnná uložena v globální paměti)

Odbočka k proměnným

```
//direktivy
#include <stdio.h>
#include <stdlib.h>
//makra
#define DELKA 10
int opakovat = 0; //Globální proměnná
//Globální prostor

//funkce
void main(){
    int pocet = 0; //lokální proměnná
    //tělo funkce
}
```

Tvorba funkcí

- Funkci deklarujeme v globálním prostoru (nad main)
- Funkce musí být deklarována výše, nežli je volána!
- Funkce může být definována až za svým voláním, pokud je řádně deklarována
- Z názvu funkce musí být vidět, co je její úkol
- <Datový typ návratové hodnoty> název(<parametry>){<tělo>}
- void pozdrav(){
  printf("Ahoj, vřele tě tu vítám!\n");
  }
- Pokud se jedná o proceduru, je datový typ návratové hodnoty void a funkce neobsahuje příkaz return

Tvorba funkcí – návratová hodnota

- Návrat hodnoty realizuje příkaz return

```
//globální promenne
int delka = 10;
int vyska = 5;
//funcke
int ObsahObdelnika()
{
    int s = delka * vyska;
    return s; //vrátí vypočtený obsah – tedy návratová hodnota
}
void main()
{
    //tělo funkce
}
```

Tvorba funkcí – Parametry

- Obecně funkce může mít libovolně parametrů
- Parametry označujeme jako argumenty
- Parametry oddělujeme čárkou
- U každého parametru udáváme datový typ

```
int ObsahObdelnika(int vyska, int delka)
{
    int s = delka * vyska;
    return s; //vrátí vypočtený obsah
}
```

Volání funkcí

- Volat lze jen výše v kódu existující funkci
- Volat funkci je možné i uvnitř příkazu print
- Funkci voláme pomocí jejího jména a parametrů

```
int ObsahObdelnika(int vyska, int delka) //funke vvpoctu S
{
    int s = delka * vyska;
    return s; //vrátí vypočtený obsah
}
void main()
{
    printf("Obsah obdelnika 10x5 je " ObsahObdelnika(5,10));
}
```

MVC

- Jde o návrhový vzor cloudových aplikací
- Je uplatnitelný i u dalších typů aplikací
- Umožňuje, aby různé části aplikace využívaly různé technologie
- Pro nás může sloužit jako vodítko k základnímu logickému členění aplikace
- Controler odbavuje požadavky klienta – řeší vstup
- Model se stará o logiku a práci s daty – jádro aplikace
- View se stará o zobrazení výsledku – výstup