Základy programování v C++ - 5. a 6. cvičení Podprogramy, rekurze, rozdělení programu do více souborů

Zuzana Petříčková

9. října 2019

Přehled

Už jsme probírali

- Vytvoření konzolové aplikace
- Proměnné a základní datové typy
- Příkazy pro řízení běhu programu
  - podmínky (IF)
  - cykly (WHILE, DO-WHILE, FOR)
  - skoky (BREAK, CONTINUE, GOTO)
- Ošetření chyb uživatele

Úvod do podprogramů

Podprogram

- část programu, kterou je možné volat (opakovaně) z různých míst kódu
- může mít parametry a vracet nějakou hodnotu

Funkce

- podprogram, který vrací nějakou hodnotu a volá se ve výrazu

Procedura

- podprogram, který nevrací hodnotu a volá se jako příkaz
- v C++ nejsou, používá se funkce bez návratové hodnoty

Úvod do podprogramů

Význam podprogramů

- zpřehlednění programu
- kód mohu používat na více místech

Podprogramy v C++ ... funkce

```
int vynasob(int a, int b)
{
    return a * b ;
}
```

- **hlavička funkce:**
  - návratová hodnota (typ funkce) nebo **void** (funkce nic nevrací)
  - název funkce (identifikátor, jméno)
  - seznam parametrů (včetně typů)
- **tělo funkce:**
  - ukončení funkce a návrat hodnoty: příkaz **return**

Úvod do podprogramů

definiční deklarace funkce

- hlavička a tělo

```
int vynasob(int a, int b)
{
    return a * b;
}
```

- deklarace funkce musí předcházet jejímu použití

Volání funkce v programu:

```
int c = vynasob(x,6);
```

Úvod do podprogramů

```
#include<iostream>
using namespace std;

// definicni deklarace
int vynasob(int a, int b)
{
    return a * b ;
}

int main()
{
    int vysledek = vynasob(3,5);
    vysledek += vynasob(4,1);
    cout << "Vysledek operace je " << vysledek << endl;
    return 0;
}
```

Úvod do podprogramů ... funkce bez parametrů typu void

```
#include<iostream>
using namespace std;

// definicni deklarace
void pozdrav()
{
    cout << "Ahoj lidi!" << endl;
}

int main()
{
    // volani funkce
    pozdrav();
    return 0;
}
```

Více funkcí se stejným jménem, ale s jinými parametry (tzv. přetěžování funkcí)

```
double vydel(double a, double b)
{
    // cout << "delim racionalni cisla" << endl;
    return a / b;
}
int vydel(int a, int b)
{
    // cout << "delim cela cisla" << endl;
    return a / b;
}
int main()
{
    cout << vydel(3, 4) << endl;
    cout << vydel(3.0, 4.0) << endl;
    //cout << vydel(3, 4.0) << endl; //chyba pri prekladu
    cout << vynasob(3.0, 4.0) << endl; // varovani pri prekla
}
```

Úvod do podprogramů

- Funkce, která vrátí maximum ze tří čísel:

```
int maximum(int a, int b, int c)
{
    if (a >= b && a >= c)
        return a;
    if (b >= a && b >= c)
        return b;
    return c;
}

int main()
{
    int a, b, c;
    ... // nacteni cisel
    cout << "maximim z cisel je " << maximum(a,b,c) << endl;
    return 0;
}
```

Úvod do podprogramů

deklarace funkce

- definiční ... hlavička a tělo

```
int vynasob(int a, int b)
{
    return a * b;
}
```

- informační (prototyp) ... pouze hlavička se středníkem

```
int vynasob(int a, int b);
```

- deklarace funkce musí předcházet jejímu použití
- pro překlad programu stačí prototyp, definice může být později nebo v jiném souboru

Volání funkce v programu:

```
int c = vynasob(a,6);
```

Úvod do podprogramů

```
#include<iostream>
using namespace std;

// informativni deklarace (prototyp funkce)
int vynasob(int a, int b);

int main()
{
    int vysledek;
    vysledek = vynasob(3,5);
    cout << "Vysledek testu je " << vysledek << endl;
    return 0;
}
int vynasob(int a, int b) // definicni deklarace
{
   return a * b ;
}
```

Proměnné a datové typy

Druhy proměnných

- lokální proměnná ... deklarována na úrovni funkce / bloku
- globální proměnná ... deklarována na úrovni programu
- konstanty

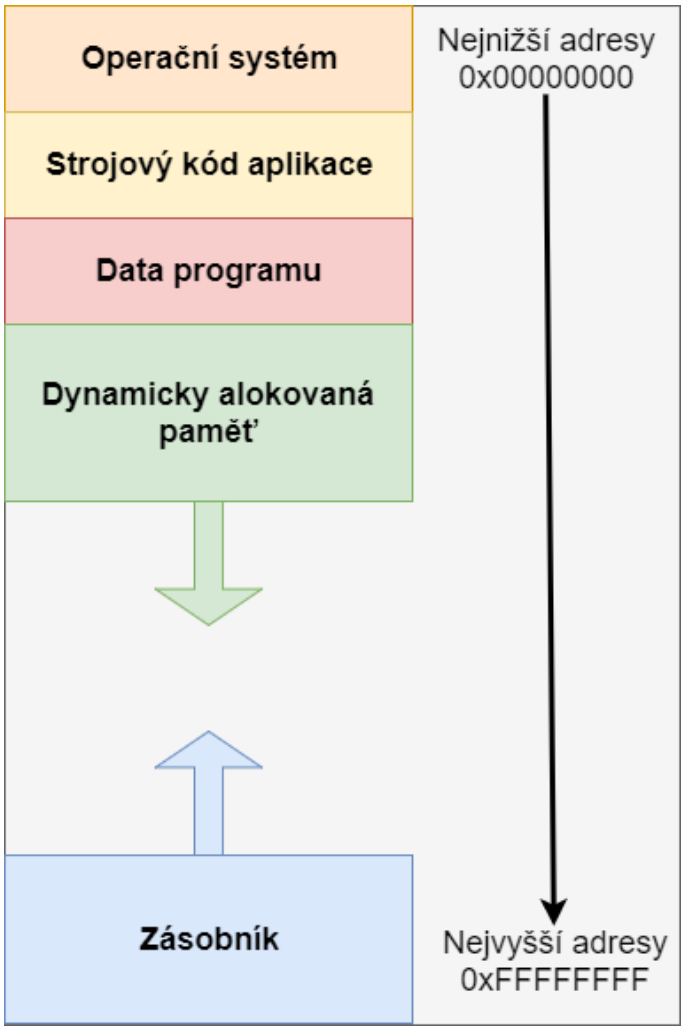
```
const int horni_mez = 10;
const double pi = 3.1415926;
```

Řízení běhu programu - podprogramy

1. Napište a zavolejte funkci **double mocnina(double x, int n)**, která spočte a vrátí x^n (řešení pomocí for-cyklu nebo while-cyklu).
2. Napište a zavolejte funkci **double euler(int n)**, která spočítá odhad hodnoty eulerovy konstanty $e = lim_{n \to \infty}(1 + \frac{1}{n})^n$ spočtením $(1 + \frac{1}{n})^n$ pro zadané $n \in N$ (využije funkci **mocnina**).
3. Napište a zavolejte funkci **int najdi(double x)**, která najde nejmenší $n \in N$ takové, že $x <= 2^n$ (využije funkci **mocnina**).
4. Napište a zavolejte funkci **bool jePrvocislo(int x)**, která otestuje, zda je zadané číslo prvočíslem (vrátí true, pokud je x prvočíslo, jinak vrátí false).

Žádná z funkcí **nebude nic vypisovat do konzole, ani z ní číst**, o povídací část programu (načtení zadání, výpis výsledků) se postará funkce main.

Organizace dat v paměti za běhu programu



zdroj: http://www.jazyk-cpp.cz/organizace-dat-v-pameti-ram/

Podprogramy (funkce) - volání funkce

- parametry se předávají hodnotou

1. volající funkce spočte hodnoty skutečných parametrů a s návratovou adresou je uloží na zásobník
2. řízení je předáno volané funkci
3. volaná funkce vyhradí na zásobníku místo pro lokální proměnné
4. vykoná se tělo volané funkce
5. volaná funkce odstraní ze zásobníku lokální proměnné
6. řízení se vrátí volající funkci
7. volající funkce odstraní ze zásobníku parametry volané funkce

Rekurze

Rekurzivní funkce

- funkce, která "volá sama sebe"ještě před svým dokončením

Příklad: faktoriál

- $f(n) = n! = n(n-1)(n-2)...2 \cdot 1, \quad n \in N$

Faktoriál - rekurzivní definice:

- $f(0) = 1$
- $f(n) = n! = n \cdot (n-1)! = n \cdot f(n-1); \quad n > 1$

Rekurze

Faktoriál ... řešení

```
long long faktorialRek(int n)
{
    if (n <= 0)
        return 1;
    return n * faktorialRek(n-1);
}
```

Rekurzivní funkce

Příklady na procvičení REKURZE:

1. Napište a zavolejte rekurzivní funkci **double mocninaRek(double x, int n)**, která spočte a vrátí x^n.
2. Napište a zavolejte rekurzivní funkci **int fibonacciRek(int n)**, která spočte a vrátí n-tý člen Fibonacciho posloupnosti.
3. **(kdo stihne)** Napište a zavolejte funkci **int fibonacci(int n)**, která spočte a vrátí n-tý člen Fibonacciho posloupnosti bez využití rekurze.

Organizace programu

Rozdělení programu do několika souborů

- zpřehlednění kódu
- znovupoužitelnost kódu
- rychlejší překlad u větších projektů (překládají se pouze změněné soubory)

Dva typy souborů

- zdrojové soubory ... přípona .cpp
- hlavičkové soubory ... přípona .h
  - deklarace funkcí, externích proměnných, datových typů, tříd ap.

Organizace programu

knihovna.h

```
#pragma once // zabrani opakovanemu vlozeni hlavickoveho soub
extern const double pi; //informativni deklarace promenne
double obsahKruhu(double r); // informatvni deklarace funkce
```

knihovna.cpp

```
#include <cmath>       // systemova knihovna
#include "knihovna.h" // nase knihovna

const double pi = 3.1415926;
double obsahKruhu(double r)
{
    return pi * pow(r,2);
}
```

main.cpp

Organizace programu

main.cpp

```
#include <iostream>
#include "knihovna.h"
using namespace std;

int main()
{
    double polomer;
    ... // nacteni polomeru
    cout << "Obsah kruhu je " << obsahKruhu(polomer) << endl;
    return 0;
}
```

Průběh překladu programu v C++

1. zpracování každého zdrojového souboru **preprocesorem**
   - odstraní komentáře
   - zpracuje direktivy (začínají #, např. #include)
   - ...
2. zpracování každého zdrojového souboru **překladačem (kompilátorem)**
   - výsledkem jsou **relativní soubory** (.obj, .o), které obsahují (binární) strojový kód
3. **sestavovací program (linker)** sestaví z relativních souborů spustitelný program
   - kontroluje, zda našel definice všech použitých funkcí a proměnných

Příklady na procvičení REKURZE:

1. Napište a zavolejte rekurzivní funkci **double mocninaRek(double x, int n)**, která spočte a vrátí x^n.
2. Napište a zavolejte rekurzivní funkci **int fibonacciRek(int n)**, která spočte a vrátí n-tý člen Fibonacciho posloupnosti.
3. **(kdo stihne)** Napište a zavolejte funkci **int fibonacci(int n)**, která spočte a vrátí n-tý člen Fibonacciho posloupnosti bez využití rekurze.

Příklad

- Rozložte program s funkcemi z 5-6. cvičení (**mocnina, euler, najdi, jePrvocislo, mocninaRek, fibonacciRek**) do více souborů:
  - **funkce.cpp** ... definiční deklarace funkcí
  - **funkce.h** ... informativní deklarace funkcí
  - **main.cpp** ... funkce main() (povídací část programu, odkud se budou jednotlivé funkce volat)

Další příklady na procvičení aktuální látky

1. Napište a zavolejte funkci **bool jeFibonacci(int x)**, která otestuje, zda je zadané číslo prvkem Fibonnaciho posloupnosti (vrátí true nebo false).
2. Napište a zavolejte funkci **int nsd(int x, int y)**, která spočte a vrátí největšího společného dělitele dvou čísel.
3. Napište a zavolejte funkci **int nsn(int x, int y)**, která spočte a vrátí nejmenší společný násobek dvou čísel.

Žádná z funkcí **nebude nic vypisovat do konzole, ani z ní číst**, o povídací část programu (načtení zadání, výpis výsledků) se postará funkce main. Případně zkuste program rozdělit do tří souborů obdobně jako v minulém příkladě.

Generování pseudonáhodných čísel

```
#include <ctime> // time
#include <cstdlib> // srand, rand
...
int main()
{
    srand(time(NULL)); // pocatecni nastaveni generatoru
                       // (volame pouze jednou)
    ...
    int cislo = rand(); // nahodne cele cislo z intervalu {0
    cislo = rand() % 10;  // nahodne cele cislo z intervalu
}
```

Příklad: myslím si číslo