Základy programování v C++ - 3. cvičení

Zuzana Petříčková

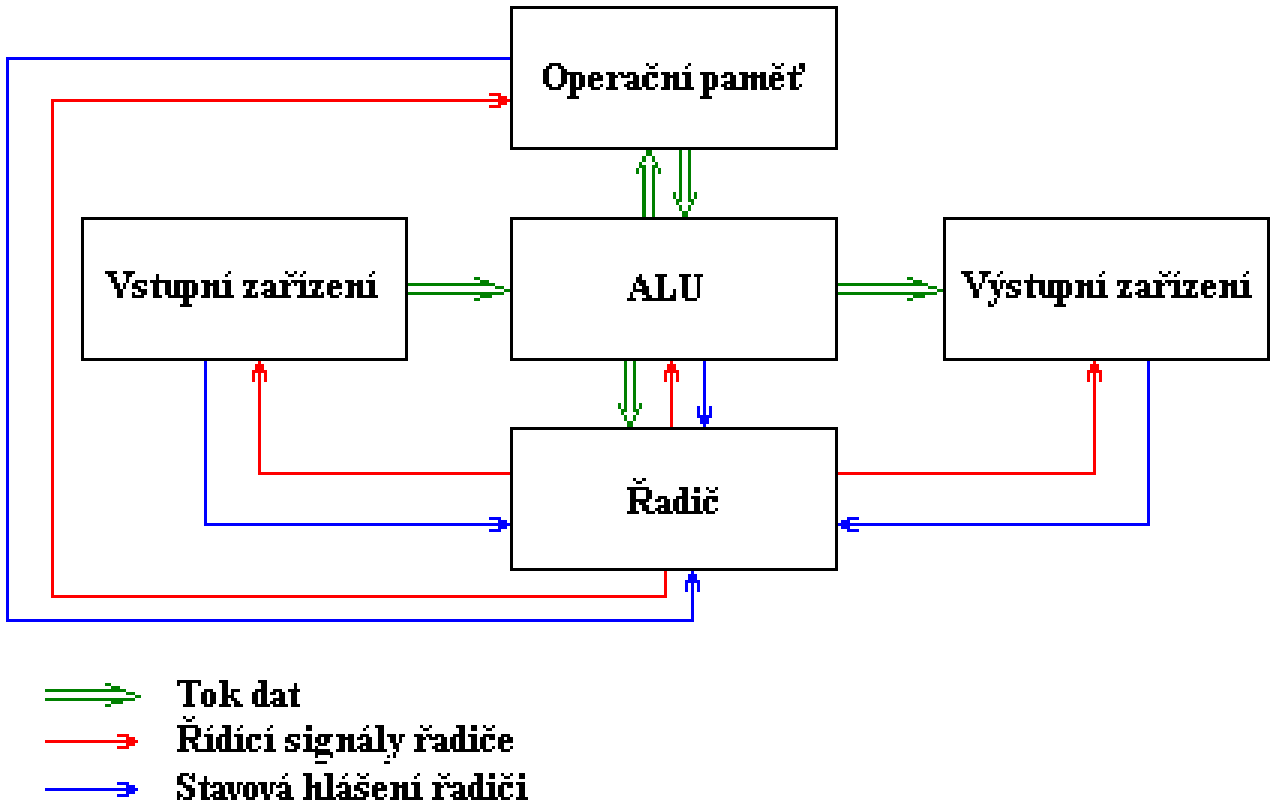
9. října 2018

Přehled

Bylo minule

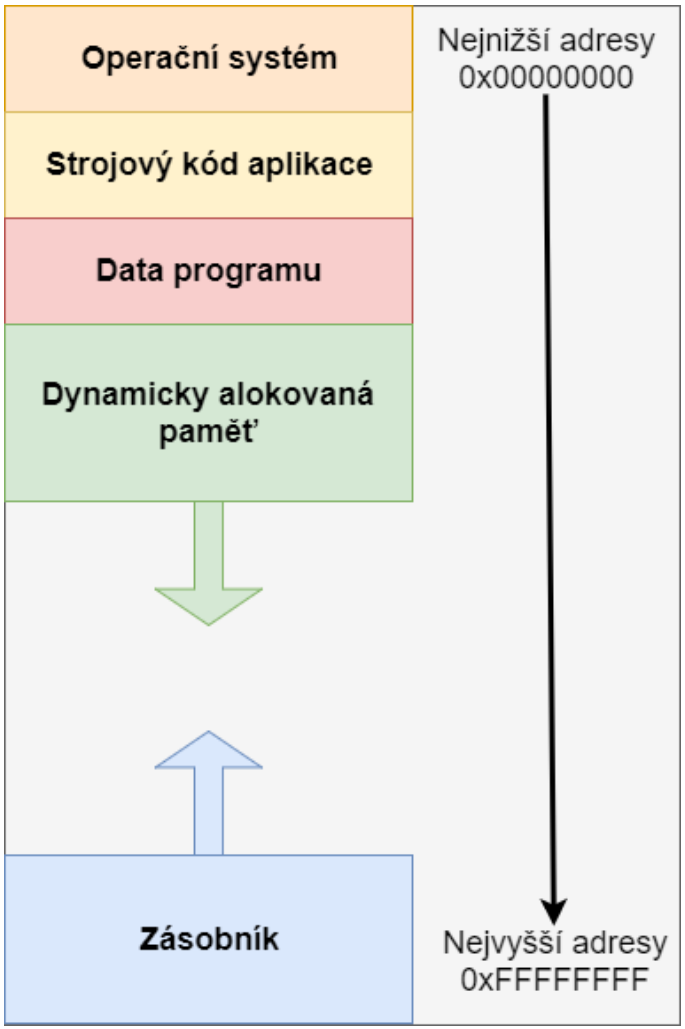
- proměnné a datové typy, definice konstanty
- číselná aritmetika, operátory, knihovna **cmath**
- konvence při psaní programů: komentáře, odsazení
- rozdělení programu na podprogramy (funkce)
  - význam
  - definiční deklarace, informativní deklarace
  - volání
- rozdělení programu do více souborů

Organizace dat v paměti za běhu programu



zdroj:
https://www.fi.muni.cz/usr/pelikan/ARCHIT/TEXTY/VNEUM.HTML

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 v 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

Bylo minule: Čtverec II

- Napište program, aby obsahoval následující funkce, které se budou volat z funkce main():

```
double obsah_ctverce(double strana);
double obsah_kruhu(double polomer);
double objem_valce(double polomer, double vyska);
double objem_koule(double polomer);
```

- Objem koule: $V = 4/3 \cdot \pi \cdot R^3$

Čtverec II ... řešení

```
const double pi = 3.1415926;
double obsah_ctverce(double strana)
{
    return strana * strana;
}
double obsah_kruhu(double polomer)
{
    return  pi * polomer * polomer;
}
double objem_valce(double polomer, double vyska)
{
    return vyska * obsah_kruhu(polomer);
}
double objem_koule(double polomer)
{
    return 4.0 / 3 * polomer * obsah_kruhu(polomer);
}
```

Aritmetika číselných datových typů

Další funkce pro práci s racionálními čísli: knihovna cmath

- sin(x), cos(x), tan(x) ... goniometrické funkce
- log(x), log10(x) ... přirozený a desítkový logaritmus
- pow(x,y) ... umocnění x^y
- sqrt(x) ... druhá odmocnina
- abs(x) ... absolutní hodnota
- ceil(x), floor(x) ... zaokrouhlení na celé číslo nahorů / dolů

Úkol:

- Vytvořte funkci, která spočítá hodnotu: $\ln(\sqrt{x + y - 5})$

Aritmetika číselných datových typů

Další funkce pro práci s racionálními čísli: knihovna cmath

- Úkol: Vytvořte funkci, která spočítá a vypíše hodnotu: $\ln(\sqrt{|x+y-5|})$

```
#include<cmath>
...
double vypocet(double x, double y)
{
        double a = log(sqrt(abs(x + y - 5)));
        cout << "Vysledek je " << a << endl;
        return a;
}
```

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, proměnných konstant, typů ap.

Organizace programu

knihovna.h

```
// zabrani opakovanemu vlozeni hlavickoveho souboru
#pragma once

// deklarace funkci a definice konstant
int vynasob(int a, int b);
extern const double pi;
```

knihovna.cpp

```
#include "knihovna.h"

const double pi = 3.1415926;
// definice
int vynasob(int a, int b)
{
        return a * b;
}
```

main.cpp

Organizace programu

knihovna.h

```
#pragma once // zabrani opakovanemu vlozeni

// deklarace funkci a konstanty
int vynasob(int a, int b);
const double pi = 3.1415926;
```

knihovna.cpp
main.cpp

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

int main()
{
    int vysledek = vynasob(3,5);
    cout << "Vysledek testu je " << vysledek << 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

Čtverec III

- Rozložte program s výpočty obsahů a objemů ("Čtverec") do více souborů:
  - **funkce.h** ... deklarace funkcí a konstant
  - **funkce.cpp** ... definice funkcí pro obsahy a objemy
  - **main.cpp** ... funkce main()

Domácí úkol

- Vytvořte "povídací" program, který umí řešit kvadratickou rovnici v základním tvaru $ax^2 + bx + c = 0$
  - Postupně načte koeficienty a, b, c ze standardního vstupu.
  - Spočte diskriminant a oba kořeny (pomocí pomocných funkcí).
  - Vypíše výsledky.
- Zkuste program rozdělit do více souborů.

Čtverec IV

- Rozšiřte program "Čtverec" o kontrolu vstupu od uživatele (číslo, nezáporné číslo)

Řízení běhu programu

- funkce
- podmíněné bloky (podmínky)
- cykly (smyčky)

Řízení běhu programu - Podmíněné bloky

Jeden příkaz:

```
if (testovaci_podminka)
    prikaz1;
```

Více příkazů:

```
if (testovaci_podminka)
{
    prikaz1;
    prikaz2;
    ...
}
```

testovaci_podminka = výraz, který lze převést na logickou hodnotu

Řízení běhu programu - Podmíněné bloky

IF - ELSE:

```
if (testovaci_podminka)
        prikaz1;
else
        prikaz2;
```

Více příkazů:

```
if (testovaci_podminka)
        prikaz1;
else
{
        prikaz2;
        prikaz3;
        ...
}
```

Řízení běhu programu - Podmíněné bloky

IF - ELSE IF - ELSE:

```
if (testovaci_podminka1)
      prikaz1;
else if (testovaci_podminka2)
      prikaz2;
else
      prikaz3;
```

- opět lze použít i bloky příkazů

Řízení běhu programu - Podmíněné bloky

Testovací podmínky

- boolovský výraz
- libobolný výraz, který lze převézt na typ **bool** (čísla, znaky, ukazatele)

Relační operátory (operátory pro porovnání)

- == ... rovnost (NE pro racionální čísla)
- != ... nerovnost (NE pro racionální čísla)
- < ... je menší
- <= ... je menší nebo rovno
- > ... je větší
- >= ... je větší nebo rovno

Řízení běhu programu - Podmíněné bloky

Relační operátory (operátory pro porovnání)

- == ... rovnost (NE pro racionální čísla)
- != ... nerovnost (NE pro racionální čísla)
- < ... je menší
- <= ... je menší nebo rovno
- > ... je větší
- >= ... je větší nebo rovno

Příklad

```
int a = 1000, b = 2000;
bool vysl = a > b;
vysl = a;
cout << ( a <= b ) << endl;
```

Řízení běhu programu - - Podmíněné bloky

Automatická konverze na logickou hodnotu

- nenulové číslo → 1 (true)
- nula → 0 (false)

```
...
int main()
{
    int cislo;
    cout << "Zadej cislo" << endl;
    cin >> cislo;
    bool b = cislo;
    ...
    if (cislo)
        ;
    else
        cout << "Zadal jsi nulu." << endl;
    return 0;
}
```

Řízení běhu programu - - Podmíněné bloky

Logické operátory (operátory pro porovnání)

- && ... logické AND
- || ... logické OR
- ! ... logická negace

```
...
int main()
{
    int cislo;
    cout << "Zadej cislo" << endl;
    cin >> cislo;
    bool b = cislo;
    ...
    if (! cislo)
        cout << "Zadal jsi nulu." << endl;
    return 0;
}
```

Řízení běhu programu - Podmíněné bloky

Logické operátory (operátory pro porovnání)

- && ... logické AND
- || ... logické OR
- ! ... logická negace

Cvičení (papír a tužka)

```
! (1 || 0)
! (1 || 1 && 0 )
! ( (1 || 0) && 1 )
```

Řízení běhu programu - Podmíněné bloky

Příklad: složitější podmínka a funkce typu bool

```
...
bool v_mezich(int x)
{
    int minimum = 0, maximum = 2000;
    return ( x <= maximum ) && (x >= minimum );
}
...
int main()
{
    int cis;
    cout << "Zadej cislo" << endl;
    cin >> cis;
    if (! v_mezich(cis))
    {
        cout << "Spatne zadani!" << endl;
        return 1;
     }
    ...
    return 0;
}
```

Řízení běhu programu - Podmíněné bloky

Příklad: čtverec

- Rozšiřte program o kontrolu vstupu od uživatele (číslo, nezáporné číslo)

```
...
int main()
{
    cout << "Prosim, zadej cislo jako delku strany ctverce:"
    double cislo;
    cin >> cislo;
    if (cislo <= 0)
    {
        cout << "Delka hrany musi byt kladne cislo." << endl;
        return 1;
    }
    cout << "Obsah ctverce o delce strany "
         << cislo << "je" << obsah_ctverce(cislo) << endl;
    return 0;
}
```

Řízení běhu programu - Podmíněné bloky

Příklady na procvičení IF-ELSE:

1. Napište a zavolejte funkci **bool je_sude(int i)**, která pro zadané celé číslo rozhodne, zda je sudé (a vypíše výsledek).
2. Napište a zavolejte funkci **void porovnej(int a, int b)**, která porovná dvě celá čísla a vypíše, které je větší.
3. Napište a zavolejte funkci **int nejvetsi(int a, int b, int c)**, která vypíše a vrátí největší ze tří čísel.
4. (pro dobrovolníky) Napište funkci **void usporadej(int a, int b, int c)**,která vypíše čísla v pořadí od největšího po nejmenší.

Řízení běhu programu - Podmíněné bloky

Příklad 1: řešení

```
bool je_sude(int i)
{
    bool r = (i % 2 == 0);
    if (r)
        cout << "Cislo " << i << " je sude." << endl;
    else
        cout << "Cislo " << i << " je liche." << endl;
    return r;

    //return (i % 2 == 0);
}
```

Řízení běhu programu - Podmíněné bloky

Příklad 2: řešení

```
void porovnej(int a, int b)
{
   if (a < b)
      cout << b << " je vetsi nez " << a << endl;
   else if (a > b)
      cout << a  << " je vetsi nez " << b << endl;
   else
      cout << "cisla jsou stejna" << endl;
}
```

Řízení běhu programu - Podmíněné bloky

Příklad 3: řešení

```
int nejvetsi(int a, int b, int c)
{
    int nej;
    if (b > a)
    {
        if (c > b)
            nej = c;
        else
            nej = b;
    }
    else
    {
        if (c > a)
            nej = c;
        else
            nej = a;
    }
    cout << "nejvetsi je " << nej << endl;  return nej;
}
```

Řízení běhu programu - Podmíněné bloky

Příklad 4: řešení

```
void usporadej(int a, int b, int c)
{
    cout << "Cisla dle velikosti (od nejvetsiho): " << endl;
    if (b > a)
    {
        if (c > b)
            cout << c << "  " << b << "  " << a << endl;
        else if (a > c)
            cout << b << "  " << a << "  " << c << endl;
        else
            cout << b << "  " << c << "  " << a << endl;
    }
    else
    {
        if (c > a)
            cout << c << "  " << a << "  " << b << endl;
        else if (c > b)
            cout << a << "  " << c << "  " << b << endl;
        else
            cout << a << "  " << b << "  " << c  << endl;
    }
}
```