Základy programování v C++ - 4. cvičení

Zuzana Petříčková

12. října 2018

Přehled

Už jsme probírali

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

Číselné datové typy – Přetypovávání

znaky	char
	unsigned char
celá čísla	short
	unsigned short
	int
	unsigned int
	long
	unsigned long
	long long
	unsigned long long
racionální čísla	float
	double
	long double

- $\Downarrow$ přetypování je implicitní (netřeba uvádět)
- $\Uparrow$ přetypování je explicitní (třeba uvádět, dochází k ořezání)

Příklad:

```
int cele = 8;
double realne = 3.14;
cele = (int) realne; // 3
realne = cele; // 3.0
```

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;
if (a == b)
    cout << "Cisla se rovnaji.";
else
    cout << "Cisla se rovnaji."
```

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 )
```

Příklad

```
if (a == 0 || b == 0)
    cout << "Jedno z cisel je 0.";
```

Číselné, relační a logické operátory – priorita a asociativita

Priorita	Operátor	Asociativita
2	postfixové ++ −−	zleva
3	prefixové ++ −−	zprava
	unární + −	zprava
	!	zprava
5	* / %	zleva
6	+ −	zleva
8	< <= > >=	zleva
9	'==' '! ='	zleva
13	&&	zleva
14	||	zleva
15	=	zprava
	'+=' '-=' '*=' '/=' '%='	zprava

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 a zavolejte 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: možné 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 1: alternativní řešení

```
bool je_sude1(int i)
{
    if (i % 2 == 0)
    {
        cout << "cislo " << i << " je sude" << endl;
        return true;
    }
    else
    {
        cout << "cislo " << i << " je liche" << endl;
        return false;
    }
}
```

Řízení běhu programu - Podmíněné bloky

Příklad 1: alternativní řešení

```
bool je_sude2(int i)
{
    if (i % 2 == 0)
    {
        cout << "cislo " << i << " je sude" << endl;
        return true;
    }
    cout << "cislo " << i << " je liche" << endl;
    return false;
}
```

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 3: řešení pomocí složených podmínek

```
int nejvetsi(int a, int b, int c)
{
    int nej;
    if (a >= b && a >= c)
        nej = a;
    if (b >= a && b >= c)
        nej = b;
    if (c >= a && c >= b)
        nej = c;
    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;
    }
}
```

Řízení běhu programu - Cykly

Cykly:

- část kódu, která se provádí vícekrát
- o dalším opakování rozhoduje splnění podmínky
- počet opakování může a nemusí být znám předem

Řízení běhu programu - Cyklus WHILE

Jeden příkaz:

```
while (testovaci_podminka)
        prikaz1;
```

Více příkazů:

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

testovaci_podminka = výraz, který lze převést na logickou hodnotu

Řízení běhu programu - Cyklus WHILE

Příklad:

```
...
int i = 0;
while (i <= 10)
{
    cout << i << " " << i*i << endl;
    i++;
}
...
```

Řízení běhu programu - Cyklus DO-WHILE

```
do
{
        prikaz1;
        prikaz2;
        ...
}
while (testovaci_podminka)
```

Příklad: funkce, která načte číslo v daném rozsahu
int nacti_cislo (int minimum, int maximum)

Řízení běhu programu - Cyklus DO-WHILE

Příklad: funkce, která načte číslo v daném rozsahu

```
int nacti_cislo (int minimum, int maximum)
{
    int n;
    cout << "Zadej cislo:" << endl;
    do
    {
        if (!(cin >> n))
            ...;
        if (n < minimum || n > maximum)
            cout << "Neplatne zadani. Zkus to znovu:" << endl;
    }
    while (n < minimum || n > maximum);
    return n;
}
```

Řízení běhu programu - cyklus WHILE

Příklady na procvičení WHILE:

1. Napište a zavolejte funkci **long long faktorial(int n)**, která pro zadané celé číslo n spočte a vrátí $n!$.
2. Napište a zavolejte funkci **double mocnina(double x, int n)**, která spočte a vrátí x^n.
3. 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.
4. (pro dobrovolníky) Napište a zavolejte funkci **int najdi(double x)**, která najde nejmenší n takové, že $x <= 2^n$.

Řízení běhu programu

Příklad 1: řešení

```
long long faktorial (int n)
{
    long long y = 1;
    while (n > 0)
    {
        y *= n;
        n--;
    }
    return y;
}

int main()
{
    int n = nacti_cislo(0,20);
    cout << "Faktorial je " << faktorial(n) << endl;
    return 0;
}
```

Řízení běhu programu

Příklad 2: řešení

```
double mocnina (double x, int n)
{
    double y = 1;
    while (n > 0)
    {
        y *= x;
        n--;
    }
    return y;
}
int main()
{
    int n = nacti_cislo(0,100);
    double x;
    cin >> x;
    cout << "Mocnina je " << mocnina(x,n) << endl;
    return 0;
}
```

Řízení běhu programu

Příklad 3, 4: řešení

```
double euler(int n)
{
    double x = (1 + 1.0 / n);
    return mocnina(x, n);
}


int najdi(double x)
{
    int n = 0;
    while (x > mocnina(2,n))
      n++;
    return n;
}
```

Domácí úkol

Minule:

- Vytvořte "povídací" program, který umí řešit v oboru reálných čísel 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.

Rozšíření:

- Program ošetří speciální případy (např. situaci, kdy rovnice nemá řešení v R, kdy má jeden dvojnásobný kořen, případ $a = 0$).
- Program bude načítat zadání a vypisovat výsledky v cyklu (ukončeném uživatelem).
- Program bude kontrolovat vstup od uživatele (že zadal číslo).