

Programování v C++

Funkce a ukazatele 5

Pavel Fibich

pavel.fibich@prf.jcu.cz

27-10-2010

Funkce - volání hodnotou

navratovy_typ nazev_funkce(typ_parametru1 parametr1, ..) { .. }

```
#include <iostream>
using namespace std;
int soucet (int a, int b) {
    int r;
    r=a+b;
    return (r);
}
void vypis (void) {
    cout << "Vypisuju!" << endl;
}
int main () {
    int z;
    z = soucet(5,3);
    cout << "Vysledek1 " << z << endl;
    cout << "Vysledek2 " << soucet(z,4) << endl;
    vypis();
}
```

Funkce - volání odkazem

Volání odkazem (propagace změn argumentů).

```
#include <iostream>
using namespace std;
void zdvojeni (int& a, int& b, int& c){
    a*=2;
    b*=2;
    c*=2;
}
int main () {
    int x=1, y=3, z=7;
    zdvojeni(x, y, z);
    cout << "x=" << x << ", y=" << y << ", z=" << z;
    return 0;
}
```

Umažte některý z &.

Funkce - def. argumenty a přetížení

Defaultní hodnota argumentů

```
int vydel (int a, int b=2) {  
    return (a/b);  
}  
...  
vydel(10,5);  
vydel(6);
```

Přetížení funkce (ne jen návratová hodnota)

```
int spocitej (int a, int b) {  
    return (a*b);  
}  
float spocitej (float a, float b) {  
    return (a/b);  
}
```

Zkuste si přetížení funkce `spocitej`

Rekurzivita funkce

.. je vlastnost funkce, která volá sebe sama.

$$n! = n * (n - 1) * (n - 2) * (n - 3) \dots * 1$$

```
#include <iostream>
using namespace std;
long factorial (long a){
    if (a > 1)
        return (a * factorial (a-1));
    else
        return (1);
}
int main () {
    long cislo;
    cout << "Zadejte cislo ";
    cin >> cislo;
    cout << cislo << "! = " << factorial (cislo);
    return 0;
}
```

Deklarace funkce

- Funkce musí být deklarována před jejím prvním použitím (viz. předchozí příklady).
 - Můžeme použít jen prototyp, tj. hlavička funkce bez těla (tj. bez { .. }).
 - Cílem je přehlednost, mít hlavičky na jednom místě.
-

```
#include <iostream>
using namespace std;
int prototyp (int, int);
int main () {
    ...
}
int prototyp (int a, int b) {
    ...
}
```

Pole jako parametr

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

void vypisPole (int arg[], int delka) {
    for (int n=0; n<delka; n++)
        cout << arg[n] << " ";
    cout << "\n";
}

int main (){
    int prvniPole[] = {5, 10, 15};
    int druhePole[] = {2, 4, 6, 8, 10};
    vypisPole (prvniPole,3);
    vypisPole (druhePole,5);
    return 0;
}
```

Ukazatele

& reference (adresa, address of)

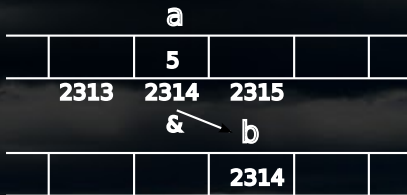
* dereference (hodnota na kterou ukazuje, value pointed by)

`int a=5, *b; // *b je int, tedy b je ukazatel na int`

`b = &a; // b je ukazatel na a`

`*b = 8;`

`cout << "b = " << b << " a = " << a << endl;`



Ukazatele

& reference (adresa, address of)

* dereference (hodnota na kterou ukazuje, value pointed by)

```
#include <iostream>
using namespace std;
int main () {
    int firstvalue = 5, secondvalue = 15;
    int * p1, * p2;

    p1 = &firstvalue; // p1 = adresa firstvalue
    p2 = &secondvalue; // p2 = adresa secondvalue
    *p1 = 10;          // value pointed by p1 = 10
    *p2 = *p1;         // value pointed by p2 = value pointed by p1
    p1 = p2;           // p1 = p2 (zkopirovani ukazatele)
    *p1 = 20;          // value pointed by p1 = 20

    cout << "firstvalue is " << firstvalue << endl;
    cout << "secondvalue is " << secondvalue << endl;
    return 0;
}
```

Pole a ukazatele

- Identifikátor pole je ekvivalentní adrese prvního prvku.
- Ukazatel + 1 je skok o jeden objekt (ne o 1 políčko v paměti).

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

int main () {
    int numbers[5];
    int * p;
    p = numbers;  *p = 10;
    p++;  *p = 20;
    p = &numbers[2];  *p = 30;
    p = numbers + 3;  *p = 40;
    p = numbers;  *(p+4) = 50;
    for (int n=0; n<5; n++)
        cout << numbers[n] << ", ";
    return 0;
}
```

Ukazatele

- ukazatel na ukazatel **

```
char a, * b, ** c;  
a = 'z';  
b = &a;  
c = &b;
```

- void *** - beztypový ukazatel, před dereferencí se musí přetypovat, na co přetypovat zjistíme pomocí `sizeof()`
- null** - neukazuje nikam

```
int * p;  
p = 0;
```

- ukazatel na funkci

```
int (*minus)(int,int)  
int funkce (int x, int y, int (*volaniF)(int,int)) {  
    ..  
    g = (*volaniF)(x,y);  
}
```

Ukol 2. - podmínky

- odvezdání přes <http://moodle.prf.jcu.cz/course/view.php?id=16>
- 0-10 bodů
- za opisování jen jedno varování a mínus body pro oba účastníky
- odevzdáte zabalený (.zip nebo .tar.gz) soubor: komentovaný zdrojový soubor (.cpp) a z něj přeložený soubor (.exe)
- na začátku .cpp souboru bude zakomentovaná a vyplněná následující hlavička

AUTOR: (jmeno prijmeni, osobni cislo zacinajici B)

EMAIL:

SKUPINA: (kam chodite a jste zapsani na cviceni)

VERZE G++: (ziskate pomoci prikazu g++ -v)

PRIKAZ PREKLADU: (jaky prikazem byl program prelozen)

OS: (operacni system, napr. Win 7 64b)

CISLO UKOLU:

Ukol 2. - zadání

Napište program, který bude akceptovat jeden argument (int v rozsahu 2-10). Argument určí kolik souřadnic bodů bude následně zadáno uživatelem. Ze souřadnic bodů program vypočítá délku lomené čáry, která je tvořena body a počet bodů v kvadrantech. Výstup programu bude mít stejný formát jako v příkladech. Součástí programu budou funkce:

- funkce pro výpočet vzdálenosti 2 bodů

```
float vzdalenost (float x1, float y1, float x2, float ←  
y2)
```

- funkce zjišťující počet bodů v kvadrantech

```
void kvadranty (float x[], float y[], int bodu, int& ←  
plusplus, int& plusminus, int& minusplus, int& ←  
minusminus)
```

Ukol 2. - příklady

- ukol2.exe 2 3

Program akceptuje jeden parametr 2-10!

- ukol2.exe 30

Program akceptuje jeden parametr 2-10!

- ukol2.exe 3

1. bod souradnice

x = 1

y = 2

2. bod souradnice

x = 8

y = 2

3. bod souradnice

x = -3

y = 2

Delka lomenne krivky dane body je 18

Pocet bodu v kvadrantech (++,+,-,-) 2 0 1 0

- ukol2.exe 2

1. bod souradnice

x = 3.4

y = 2.6

2. bod souradnice

x = -4.2

y = -1.6

Delka lomenne krivky dane body je 8.68332

Pocet bodu v kvadrantech (++,+,-,-) 1 0 0 1

Ukol 2. - příklady

- ukol2.exe 3

1. bod souradnice

x = -4

y = -2

2. bod souradnice

x = -6

y = -4

3. bod souradnice

x = -7

y = -7

Delka lomenne krivky dane body je 5.9907

Pocet bodu v kvadrantech (++,+,-,-) 0 0 0 3

- ukol2.exe 4

1. bod souradnice

x = -9

y = -2

2. bod souradnice

x = -7

y = 1.4

3. bod souradnice

x = -3

y = -3.7

4. bod souradnice

x = 1

y = 1

Delka lomenne krivky dane body je 16.5978

Pocet bodu v kvadrantech (++,+,-,-) 1 0 1 2