# Úvod do programování

# **Lekce 5**

## *Funkce*

***definice funkce***
zahrnuje specifikaci počtu a typů parametrů, typu návratové hodnoty a vlastního těla funkce

příklad: funkce najde maximum ze dvou celých čísel

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

int maximum(int a, int b){                    // hlavicka funkce
  return(a>b ? a : b);                        // telo funkce
}

int main(void){
  int c1,c2;
  cout << "zadej dve cela cisla: ";
  cin >> c1 >> c2;
  cout << "vetsi je: " << maximum(c1,c2) << endl; // volani funkce
}
```

poznámky:

- příkaz `return` předává řízení nadřazené funkci a zároveň vrací výsledek
- pokud funkce nemá parametry, popř. nevrací žádnou hodnotu, použijeme typ `void`

příklad: program využívá funkce `tab()` a `podtrhni()` pro formátování výpisu

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

void tab(void){                   // zadne parametry, nevraci
hodnotu
  cout << "          ";           // telo funkce
}                                 // neni nutno pouzit return()

void podtrhni(void){
  int i;
  putchar('\n');
  for(i = 1;i < 60;i++)           // podtrhne
    cout << '-';
  cout << endl;                   // ukonci radek
}

int main(void){
  cout << "prvni sloupec";
  tab();                          // zavorky () nesmi byt
vynechany!
  cout << "druhy sloupec";
  tab();
```

```
  cout << "treti sloupec";
  podtrhni();
}
```

***rekurzivní funkce***

- funkce může volat sama sebe
- využití rekurze může vést k nepřehlednému a neefektivnímu kódu – nevyužívat, pokud není nutné

příklad: výpočet faktoriálu celého čísla s využitím rekurze

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

int fakt(int n){
  return((n<=0)? 1 : n*fakt(n-1));   // rekurzivni volani funkce
}

int main(void){
  int k;
  cout << "zadej k: ";
  cin >> k;
  cout << k << "! = " << fakt(k) << endl;
}
```

***oblast platnosti proměnných***

- lokální proměnné – definice uvnitř funkce, platnost do konce funkce
- globální proměnné – definice vně funkce, platí do konce souboru

příklad: lokální proměnná z funkce `main()` je nedostupná ve funkci `tisk_b()`

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

int a;                 // definice globalni promenne

void tisk_a(void){
   cout << a << endl;
}

void tisk_b(void){
  int b;              // definice lokalni promenne ve fci tisk_b()
  b = 20;
  cout << b << endl;  // tiskne hodnotu "20"
}

int main(void){
  int b;              // definice lokalni promenne ve fci main()
  a = b = 10;
```

```
  tisk_a();          // tiskne hodnotu "10"
  tisk_b();
  cout << b << endl;  // tiskne hodnotu "10"
}
```

***ukazatel (pointer)***

objekt, který ukazuje na hodnotu uloženou v paměti

```
int *p_a;     // definuje ukazatel na typ int
int a;        // definuje proměnnou int
p_a = &a;     // "p_a" nyní ukazuje na "a"
*p_a = 6;     // uloží hodnotu do "a" (totéž co a = 6)
p_a++         // posun o velikost daného typu
sizeof(int)   // vrací velikost typu int [totéž co sizeof(a)]
sizeof(int *) // vrací velikost ukazatele na int
```

poznámky:

- referenční operátor * vrací hodnotu uloženou na adrese
- deferenční operátor & vrací adresu objektu

***volání parametrů***

- volání hodnotou – do funkce se předávají kopie skutečných parametrů, nelze tedy skutečné parametry uvnitř funkce měnit
- volání odkazem – do funkce jsou předány odkazy (adresy) skutečných parametrů, všechny manipulace ve funkci probíhají se skutečnými parametry

příklad: funkce vymění obsah proměnných

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

void vymen(int &x, int &y){
  int pom;
  pom = x;
  x = y;    // ulozeno do skutecneho parametru, tj. "a"
  y = pom;  // ulozeno do "b"
}

int main(void){
  int a = 1, b = -1;
  cout << "puvodne: a=" << a << ", b=" << b << endl;
  vymen(a,b);                  // do funkce posilame adresy!
  cout << "po vymene: a=" << a << ", b=" << b << endl;
}
```

poznámka:

- volání odkazem lze využít pro vrácení dvou a více hodnot z funkce, viz. následující příklad

příklad: funkce vypočítá součet a rozdíl dvou celých čísel a výsledek vrátí skrz parametry

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

void spatne(int a, int b, int soucet, int rozdil){
   soucet = a + b;                 // tyto hodnoty budou po
opusteni
   rozdil = a - b;                 // funkce zapomenuty
}

void spravne(int a, int b, int &soucet, int &rozdil){
   soucet = a + b;              // "soucet" -> "suma"
   rozdil = a - b;              // "rozdil" -> "roz"
}

int main(void){
  int a = 10, b = 22, suma = 0, roz = 0;
  spatne(a,b,suma,roz);     // predani "suma","roz" hodnotou
  cout << "spatne():  suma= " << suma << ", roz= " << roz << endl;
  spravne(a,b,suma,roz);  // predani "suma","roz" odkazem
  cout << "spravne(): suma= " << suma << ", roz= " << roz << endl;
}
```

příklad: program demonstruje chybu výpočtu numerické derivace

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

double der(double x, double dx){
  double d;
  d = (sin(x+dx) - sin(x)) / dx;   // numericka derivace
  return(d);
}

int main(void){
  int i;
  double dx;
  for(i = -1; i > -16; i--){
    dx = pow(10,i);       // nastavi dx
    cout << setw(3) << i << fixed << setw(25) << setprecision(20);
    cout << cos(M_PI/4) - der(M_PI/4,dx) << endl;
  }
}
```

příklad: program ověří přesnost vlastní implementace funkce sin(x) založené na rozvoji sin(x) v mocninnou řadu:

$$\sin(x) = \sum_{i=1}^{\infty} \frac{(-1)^{i+1} x^{2i-1}}{(2i-1)!}$$

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

double faktorial(int n){
  double f = 1;
  int i;
  for(i = 2; i <= n; i++)
    f *= i;
  return(f);
}

double mocnina(double x, int a){
  double moc = 1;
  int i;
  for(i = 1; i <= a; i++)
    moc *= x;
  return(moc);
}


double mujsin(double x, int clenu){
  double suma;
  int i, n, znamenko;
  suma = 0; znamenko = 1;
  for(i = 1;i <= clenu; i++){
    n = 2*i - 1;
    suma += znamenko*mocnina(x,n)/faktorial(n);
    znamenko *= -1;          // znamenko v rade alternuje
  }
  return(suma);
}

int main(void){
  double x, dx, s1, s2, ds;
  int clenu;
  x=M_PI/4.;
  // nejdrive zjistime jak presnost vypoctu zavisi na poctu clenu
  // toto provedeme pro uhel pi/4
  cout << "clenu    mujsin         sin         rozdil\n";
  cout << "-----------------------------------------\n";
  for(clenu = 1; clenu <= 5; clenu++){
    s1 = mujsin(x,clenu);
```

```
    s2 = sin(x);
    ds = s1 - s2;
    cout << setw(3) << clenu << fixed << setprecision(8) <<
setw(14) << s1;
    cout << setw(13) << s2 << setw(14) << ds << endl;
  }
  // nyni overime zda je presnost stejna pro vsechny uhly
  clenu = 3;
  dx = M_PI/2./10.;          // krok v uhlu
  cout << endl << endl;      // odradkuje
  cout << "   x       mujsin         sin        rozdil\n";
  cout << "-----------------------------------------\n";
  for(x = 0;x <= M_PI/2.;x += dx){
    s1 = mujsin(x,clenu);
    s2 = sin(x);
    ds = s1 - s2;
    cout << setprecision(4) << x << setprecision(8) << setw(13) <<
s1;
    cout << setw(13) << s2 << setw(13) << ds << endl;
  }
}
```

***funkční prototyp***

pokud v programu volání nějaké funkce předchází její definici, uvedeme na začátku programu úplný funkční prototyp volané funkce; často jsou tyto deklarace umístěny v hlavičkovém souboru

příklad: funkce `mujsin()` a `mujcos()` se volají navzájem

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

double mujsin(double x);  // uplny funkcni prototyp

double mujcos(double x){
  if(x < M_PI/4.)
    return(1 - x*x/2.); // pro male uhly, jinak pouziji mujsin()
  else
    return(mujsin(M_PI/2. - x));
}

double mujsin(double x){
 if(x < M_PI/4.)
   return(x);      // pro male uhly, jinak pouziji mujcos()
 else
   return(mujcos(M_PI/2. - x));
}

int main(void){
  printf("sin(0.01)=%f\n",mujsin(0.01));
```

```
  printf("sin(pi/2-0.01)=%f\n",mujsin(M_PI/2.-0.01));
}
```

***ukazatel na funkci***

```
double (*p_fce)(double);  // ukazatel na funkci vracející double
p_fce = sin;          // p_fce ukazuje na funkci sin()
y = p_fce(x);         // nyní totéž jako: y = sin(x)
p_fce = cos;
y = p_fce(x);         // nyní totéž jako: y = cos(x)
```

příklad: stejný výraz vrátí buď sin(x) nebo cos(x)

```
#include<iostream>
#include<cmath>
using namespace std;
int main(void){
  double x = M_PI/6.;  // 30deg
  double (*p_fce)(double);

  p_fce = sin;
  cout << p_fce(x) << endl;

  p_fce = cos;
  cout << p_fce(x) << endl;
}
```