Petr Šaloun

katedra informatiky FEI VŠB-TU Ostrava

13. října 2014

Funkce (1) – základy

Funkce
– základní stavební kámen procedurálního programu,
– nemá být rozsáhlá,
– řeší ucelený problém,
– je-li problém příliš složitý, volá na pomoc další funkce, – může vracet návratovou hodnotu,
– může mít argumenty.

Každý C++ program:

```
int main() {
ldots
return 0;
}
```

Funkce (2) – definice, deklarace

```
typ jmeno(formalni argumenty) {
 telo  funkce
}
```

definice funkce

– identifikátor,
– typ návratové hodnoty,
– typ a názvy argumentů, uveden mezi (a)
– tělo funkce, uvedeno mezi { a })
– vytváří (generuje) kód, který bude při každém volání funkce proveden

deklarace funkce

– informace pro překladač, nevytváří výkonný kód,
– kromě těla funkce jako definice funkce.

Funkce (3) – argumenty

argumenty funkce
–formální argumenty
–skutečné argumenty.

definice funkce:

```
void zamenit_hodnotou_nejde(int fa, int fb) {
  int pomocna = fa;
  fa = fb;
  fb = pomocna;
} // void zamenit_hodnotou_nejde(int fa, int fb)
```

volání funkce:

```
  int sprvni= 123, sdruhy = -456;

  zamenit_hodnotou_nejde(sprvni, sdruhy);
```

Funkce (4) – volání funkce

jmeno(skutecne argumenty)

skutečný argument – identifikátor proměnné, konstanty, výraz, přímo uvedená hodnota konstanty.

Počet *skutečných argumentů* funkce je dán potřebou programátora:
jmeno() – ? skutečných argumentů,
jmeno(void) – nula skutečných argumentů,
jmeno(...) – proměnným počet argumentů,
jmeno – bez závorek – adresa funkce (vstupní bod funkce).

Funkce (5) – návratová hodnota funkce

return vyraz_vhodneho_typu;

příklad:

```
int isqr(int i) {
  return i * i;
}
...
  int vysledek;
  vysledek = isqr(4);
```

Funkce (6) – Argumenty funkcí a způsob jejich předávání

– *hodnotou:* kopie skutečných arg. na zásobník, změna formálních argumentů jen ve funkci,
– *adresou:* ukazatel realizuje vazbu mezi formálním a skutečným argumentem,
– *odkazem:* vazbu mezi skutečnými a. a formálním a. realizuje překladač.

poznámka: pozor
& – adresový operátor
$*$ – dereference

Funkce (7) – příklad: fn-argumenty.cpp (a)

```
void zamenit_hodnotou_nejde(int fa, int fb) {
 int pomocna = fa;
 fa = fb;
 fb = pomocna;
} // void zamenit_hodnotou_nejde(int fa, int fb)

void zamen_ukazatelem(int *fa, int *fb) {
 int pomocna = *fa;
 *fa = *fb;
 *fb = pomocna;
} // void zamen_ukazatelem(int *fa, int *fb)

void zamen_odkazem(int& fa, int& fb) {
 int pomocna = fa;
 fa = fb;
 fb = pomocna;
} // void zamen_odkazem(int& fa, int& fb)
```

Funkce (7) – příklad: fn-argumenty.cpp (b)

```
int sprvni= 123, sdruhy = -456;
cout << "pocatecni stav" << endl;
cout << "sprvni= " << sprvni<< ", sdruhy = " << sdruh

zamenit_hodnotou_nejde(sprvni, sdruhy);
cout << "po volani fce 'zamenit_hodnotou_nejde()'" <<
cout << "sprvni= " << sprvni<< ", sdruhy = " << sdruh

zamen_ukazatelem(&sprvni, &sdruhy);
cout << "po volani fce 'zamen_ukazatelem()'" << endl;
cout << "sprvni= " << sprvni<< ", sdruhy = " << sdruh

zamen_odkazem(sprvni, sdruhy);
cout << "po volani fce 'zamen_odkazem()'" << endl;
cout << "sprvni= " << sprvni<< ", sdruhy = " << sdruh
```

Funkce (7) – příklad: fn-argumenty.cpp (c)

```
pocatecni stav
skutecny1 = 123, skutecny2 = -456

po volani fce 'zamenit_hodnotou_nejde()'
skutecny1 = 123, skutecny2 = -456

po volani fce 'zamen_ukazatelem()'
skutecny1 = -456, skutecny2 = 123

po volani fce 'zamen_odkazem()'
skutecny1 = 123, skutecny2 = -456
```

Paměťové třídy

auto – umístění na zásobník, neinicializované;
extern – nevytvářet, bude připojen z jiného modulu;
register – přání umístit do registru procesoru, neinicializované;
static – umístění do datového segmentu, inicializované nulou;
typedef – nemá paměťový význam, pojmenovává konstrukci definice;

modifikátor – význam
const – vyjadřuje neměnitelnost;
volatile – vyjadřuje neustálou proměnnost (nekešovat)

Pravidla pro implicitní určení paměťové třídy.

objekt – paměťová třída, výklad, umístění
glob. prom. – **static** a **extern**, inicializovány nulou, DS;
lokální prom. – **auto**, neinicializovány, zásobník;
formální arg. – **auto**, neinicializovány, zásobník;
def. fce – **extern**, definice dle kódu, CS

Rekurze

Formální argumenty funkce i její lokální proměnné se umisťují na **zásobník** – LIFO *Last In First Out – poslední dovnitř, první ven*.

příklad: faktoriál
$n! = n \times (n-1)!$
$0! = 1$.

Rekurze

Formální argumenty funkce i její lokální proměnné se umisťují na **zásobník** – LIFO *Last In First Out – poslední dovnitř, první ven*.
$n! = n \times (n-1)!$
$0! = 1$.

Příklad: faktoriál (zdroják)

```
double fact(long n) {
  if (n == 0L)
    return 1.0L;
  else
    return n * fact(n-1);
} // double fact(long n)

  static long n;
  cout << "Pro vypocet faktorialu zadej prirozene n:";
  cin >> n;
  cout << n << "! = " << fact(n) << endl;
```

Příklad: faktoriál (výstup)

```
Pro vypocet faktorialu zadej prirozene cislo n:5
5! = 120

Pro vypocet faktorialu zadej prirozene cislo n:69
69! = 1.71122e+098
```