Řídicí struktury

Pavel Čeleda

celeda@liberouter.org

Kurz jazyka C - přednáška č. 2

Řídicí struktury

- každé vnoření odsadit minimálně 2mi mezerami;
- levá { začínající tělo fce je vždy samostatně (platí i pro })
- { je na stejné řádce jako if, else, for, while, do, switch
- } těchto příkazů na samostatné řádce:

```
if (x == 5) {
  z = 7;
  y = 5;
}
else { /* nemusí být pro jednořádkové operace */
  z = 4;
}
```

Řídicí struktury

- Booleovské výrazy

rovnost	==
nerovnost	!=
log. součin	&&
log. součet	||
negace	!
relační operátory	<, <=, >=, >.

- POZOR porovnání není = ale ==
- priorita vyhodnocení výrazů na str. 270

Řídicí struktury

- Podmíněný výraz ? ternární operátor
  - *podmínka ? výraz_1 : výraz_2;*
  - význam **if** (podmínka) **then** výraz_1 **else** výraz_2;

- Operátor čárky
  - zaručí vyhodnocení levého operandu před pravým ||, &&, ?:

```
int i=2, j = 3; /* není operátor čárky */
j = (c + d, e + c);
```

  - používat pouze u **for** nebo **while**

- Příkaz **if** a příkaz **if-else**

Iterační příkazy - cykly

- **while**, **for**, **do - while**
- **break** - ukončuje, nejvnitřnější neuzavřenou smyčku - opouští okamžitě cyklus.
- **continue** - skáče na konec nejvnitřnější neuzavřené smyčky a tím vynutí další iterační cykly - cyklus neopouští.

Smyčka while

- *while (výraz) {*
  příkaz;
  }

- ```
  while (x < 10)
    x++;
  ```

- ```
  void main(void)
  {
    int c;

    while (1) {
      if ((c = getchar()) < ' ')
         continue;
      if (c == 'z')
         break;
      putchar (c);
    }
  }
  ```

Smyčka do - while

- *do {*
  příkazy;
  } while (výraz);

- pracuje dokud je pravda (!= 0)

```
do {
  i--;
} while (i);
```

-

```
void main(void)
{
  int c;

  do {
    if ((c = getchar()) > ' ')
      putchar(c);
  } while (c != 'z');
}
```

Smyčka for

- *for (výraz_start; výraz_stop; výraz_iter)*
  příkaz;
  }

- ```
  for  ( i= 0; i < 100; i++) {
    c += getchar();
  }
  ```

- for (;;) /* nekonečná for smyčka */
- **break** použít maximálně jednou
- continue nahradit **if** - **else**
  ```

Rozhodování switch

- mnohonásobné větvení programu;
- nelze udělat výčet několika hodnot pro jeden příkaz;
- rozhoduje se podle proměnné int;
- každá větev musí být ukončena příkazem break;
- doporučuje se definovat všechny možnosti přes default, provádí se jako poslední možnost ať je v kódu kdekoliv.

- ```
  switch ( výraz) {
       case hodnota_1: příkaz_1; break;
       case hodnota_2: příkaz_2; break;
       default: příkaz_def; break;
  }
  ```
  ```

Skok goto

- v dobře napsaném programu by se neměl vyskytovat;
- používá se jako odskok z vnořených cyklů;
- pro jednoduchou smyčku používat break;
- nemusí se předem definovat.

- ```
  for (i = 0; i < 10; i++) {
          for (j = 0; j < 10; j++) {
                  for (k = 0; k < 10; k++) {
                       if (k[x] == 0)
                          goto Error;
                     a[i] = a[i] + b[j] / c[i];
                  }
          }
  }
  Error:
  ```
  ```

Terminálový vstup a výstup – printf

- Základní zápis
  - %[příznaky][šířka][.přesnost][modifikátor] **konverze**
- Konverze

c	znak;
d, i	číslo **int** se znaménkem;
u	číslo **int** bez znaménka;
x	číslo **int** šestnáctkově malá písmena;
X	číslo **int** šestnáctkově velká písmena;
o	číslo **int** osmičkově;
f	číslo **float** nebo **double;** [-]dddd.dddd
e	číslo **float** nebo **double;** [-]d.dddd e[+/-]ddd
s	řetězec (očekává ukončující znak \0

- před d, u, x, X je možné doplnit l, pak je to LONG např. ld
- % tiskne vlastní znak „%“

Terminálový vstup a výstup – printf

- Základní zápis
  - %[příznaky][šířka][.přesnost][**modifikátor**] konverze
- Modifikátor
  - volitelný znak, musí přecházet znaku konverze;
  - mění implicitní velikost konverze;

    h d,i na typ **signed short int**;
    a u,o,x,X na typ **unsigned short int**;

    l d,i na typ **signed long int**;
    a u,o,x,X na typ **unsigned long int**;

    L f,e, E, g, G na **typ long double**.

Terminálový vstup a výstup – printf

- Základní zápis
  - %[příznaky][šířka][**.přesnost**][modifikátor] konverze
- Přesnost
  - dekadické číslo;
  - pro d, i, u, o, x, X nastavuje šířku;
  - pro f, e, E nastavuje počet cifer ze desetinnou tečkou;
  - pro g, G nastavuje maximální počet významových cifer;
  - pro s nastavuje maximální počet tištěných znaků;

!	desetinná tečka je také jeden znak;
.5	tiskne 5 znaků (ořeže nebo zaokrouhlí);
.0	pro d,i,u,o,x,X nastavuje implicitní hodnotu;
.	stejné jako .0
*	počet znaků je dán předchozím parametrem.

Terminálový vstup a výstup – printf

- Základní zápis
  - %[příznaky][**šířka**][.přesnost][modifikátor] konverze
- Šířka
  - dekadické číslo, nastavuje minimální počet vypisovaných znaků;
  - je-li číslo větší, pak se na šířku nebere ohled;

    5 tiskne se 5 znaků, má-li hodnota méně, je doplněna mezerami zprava;

    05 tiskne se 5 znaků, má-li hodnota méně, je doplněna nulami zleva;

    * počet znaků je udán předchozí hodnotou.

Terminálový vstup a výstup – printf

- Základní zápis
  - %[**příznaky**][šířka][.přesnost][modifikátor] konverze
- Příznaky
  - musí bezprostředně následovat za %;

-	výsledek je zarovnán doleva a zprava se doplní mezery, pokud není uveden, pak se zarovnává doprava a zleva se doplní mezery nebo nuly;
+	číslo bude vždy vytištěno se znaménkem +.
#	před osmičkové číslo doplní 0, před x,X doplní 0x / 0X
f, e, E	výsledek vždy obsahuje desetinnou tečku.

Terminálový vstup a výstup – printf

- Příklady
  - n = 555 nebo n = 5.5

příznak	6d	6o	8x	10.2e	10.2f
%-+#0	+555	01053	0x22b	+5.50e+00	+5.50
%-+#	+555	01053	0x22b	+5.50e+00	+5.50
%-+0	+555	1053	22b	+5.50e+00	+5.50
%-+	+555	1053	22b	+5.50e+00	+5.50
%-#0	555	01053	0x22b	5.50e+00	+5.50
%0	000555	001053	0000022b	005.50e+00	0000005.50
%	555	1053	22b	5.50e+00	5.50