Cykly

Základy programování 1

Martin Kauer

Z minula

- Chary můžete používat jako znaky ale i jako čísla 0 - 255. Jakou formu vybrat záleží na konkrétní aplikaci. Když pracujete se znaky, používejte konstanty znaků (’a’,’0’, ...).
- Rozmyslete se jaká konstrukce pro větvení je pro danou aplikaci výhodná.
- Větvení si dobře promyslete a nebojte se ho popřípadě celé předělat.
- Programy si vždy pořádně otestujte.
  - Program si vyzkoušejte s nejrůznějšími vstupy.
  - Program nesmí havarovat = robustnost.
  - Program se musí chovat dle zadání (dávat správné výsledky) = korektnost.
- Zejména se soustřeďte na hraniční případy vstupů.

Motivace pro cykly

- Vypište prvních 5 přirozených čísel

```
printf("1\n");
printf("2\n");
printf("3\n");
printf("4\n");
printf("5\n");
```

- Vypište prvních n přirozených čísel

```
int n;
scanf("%d", &n);

if (n >= 1) printf("1\n");
if (n >= 2) printf("2\n");
if (n >= 3) printf("3\n");
if (n >= 4) printf("4\n");
if (n >= 5) printf("5\n");
...
```

Cyklus while

- Opakování bloku příkazů
- Dokud je splněna uvedená podmínka
- Podmínku píšeme na začátek cyklu
- Testuje se na začátku každého průchodu cyklem
- Tělo cyklu nemusí být provedeno ani jednou
- Vhodné, když nevíme dopředu počet iterací
- Typický zápis:

```
while (podmínka)
{
    příkazy
    ...
}
```

Příklady while cyklu

```c
int n, i;

scanf("%d", &n);

i = 1;
while (i <= n){
    printf("%d\n", i);
    i++;
}
```

```c
int vetsi, mensi, zbytek;
...
while (mensi != 0){
    zbytek = vetsi % mensi;
    vetsi = mensi;
    mensi = zbytek;
}

printf("Vysledek je %d. \n", vetsi);
```

Cyklus for

- Složitější konstrukce cyklu
- Podporuje nejběžnější způsob použití cyklu
  - Inicializace řídících proměnných před cyklem
  - Test podmínky na začátku cyklu
  - Změny řídících proměnných mezi průchody
- Typický zápis:

```
for (inicializace; podmínka; iterace)
{
    příkazy
    ...
}
```

Příklady for cyklu

```c
int od, po, i;

scanf("%d", &od);
scanf("%d", &po);

for (i = od; i <= po; i++)
{
    printf("%d\n", i);
}
```

```c
int od, krok, i;

scanf("%d", &od);
scanf("%d", &krok);

for (i = od; i > 0; i-= krok)
{
    printf("%d\n", i);
}
```

For vs. while cyklus

- Cyklus for lze přepsat pomocí while
- For cyklus:

```
for (inicializace; podmínka; iterace)
{
    příkazy
    ...
}
```

- Odpovídá:

```
inicializace;
while (podmínka)
{
    příkazy
    ...
    iterace;
}
```

Největší společný dělitel (pomocí for a nepěkně)

```
int prvni, druhe;
int vetsi, mensi, zbytek;
...
for (vetsi = max(prvni,druhe), mensi = min(prvni,druhe) ; mensi != 0; )
{
    zbytek = vetsi % mensi;
    vetsi = mensi;
    mensi = zbytek;
}

printf("Vysledek je %d. \n", vetsi);
```

Cyklus do-while

- Na rozdíl od předchozích typů cyklu testuje podmínku až na konci prvního průchodu
- Tělo cyklu se vždy alespoň jednou provede
- Typický zápis:

```
do
{
    příkazy
    ...
} while (podmínka);
```

Příklady cyklu do-while

```c
int n, i;

scanf("%d", &n);

i = 1;
do
{
    printf("%d\n", i);
    i++;
} while (i <= n);
```

```c
int vetsi, mensi, zbytek;
...
do
{
    zbytek = vetsi % mensi;
    vetsi = mensi;
    mensi = zbytek;
} while (mensi != 0);

printf("Vysledek je %d. \n", vetsi);
```

Přerušení cyklu

- Cykly bývají v reálných programech i složitější
- Někdy potřebujeme přerušit cyklus i jinde než na začátku/konci
- Často v závislosti na různých podmínkách
- Příkazy pro přerušení cyklu:
  - okamžité vyskočení - `break;`
  - přerušení aktuálního průchodu a přechod k dalšímu - `continue;`
- Vztahují se vždy k "nejbližšímu" cyklu

Příklad přerušení break

```
int cislo;
int mocnina;
int min = 100;

for (cislo = 1; ; cislo++)
{
    mocnina = cislo * cislo;
    if (mocnina > min)
    {
        printf("%d\n", mocnina);
        break;
    }
}
```

Příklad přerušení continue

```
int pocet = 5;
int cislo = 1;
int mocnina = cislo * cislo;
int max = 500;

while (mocnina < max)
{
    printf("%d, ", mocnina);
    cislo++;
    mocnina = cislo * cislo;
    if ((cislo-1) % pocet != 0) continue;
    printf("\n");
}
```

Složitější zacyklení

- Jeden cyklus může (samozřejmě) být uvnitř druhého cyklu:

```
int pocet = 10;

int delkaRadku = 3;

int i, j;

for (i = 0; i < pocet; i++)
{
    for (j = 0; j < pocet; j++)
    {
        printf("%d * %d = %d, \t", i, j, i * j);
        if ((i * pocet + j + 1) % delkaRadku != 0) continue;
        printf("\n");
    }
}
```

- Příkazy pro přerušení cyklu se týkají vždy toho nejvnitřnějšího cyklu, ve kterém jsou uvedeny.

Poznámky

- Co patří do předpisu cyklu?

```
for (delitel = 2; delitel <= cislo / 2;)
{
      …
      delitel++;
}
```

- Dodržovat jednu konvenci při psaní složených závorek!

```
for (a = 0; a < velikost; a++) {
     if (hvezda > velikost)
     {
          break; }
     else {
          …
     }}
```

- Zbytečné podmínky / větvení

```
for (cislo = 1; cislo <= velikost; cislo++)
{
     if (cislo > velikost)
          break;
}
```

Cvičení

Napište program, který umí do konzole vykreslit pomocí zadaného znaku pravoúhlý trojúhelník a čtverec přesně tak, jak je v následující úkázce. Od uživatele načtěte volbu co se má vykreslit, pomocí kterého znaku a délku strany. Pokud uživatel zadá neplatnou volbu, program po uživateli bude požadovat zadání znova a to dokud volba nebude platná.

Příklady použití:

```
Co si prejete vykreslit:
     a) trojuhelnik
     b) ctverec
a
Zadejte znak: $
Zadejte velikost strany: 4
$
$$
$ $
$$$$
```

```
Co si prejete vykreslit:
     a) trojuhelnik
     b) ctverec
b
Zadejte znak: *
Zadejte velikost strany: 3
***
* *
***
```

```
Co si prejete vykreslit:
     a) trojuhelnik
     b) ctverec
1
Neplatna volba, zadejte volbu znovu.
a
Zadejte znak: +
Zadejte velikost strany: -1
Neplatna volba, zadejte volbu znovu.
2
+
++
```