Úvod do programování

Lekce 2

Řízení běhu programu - řídící struktury

větvení programu v závislosti na podmínce

```
if (výraz) příkaz
```

- příkaz se provede pokud má výraz nenulovou hodnotu
- pokud chceme provést více než jeden příkaz, je nutno celý blok uzavřít do složených závorek a vytvořit tak složený příkaz
- splnění resp. nesplnění podmínky je reprezentováno hodnotami `1` resp. `0`
- součástí výrazu může být i přiřazení, např. výsledek výrazu `a=b` je hodnota uložená do proměnné "`a`"

příklad: program vypočítá absolutní hodnotu zadaného reálného čísla

```
#include<iostream>
using namespace std;
int main(void){
  double x;
  cin >> x;
  if (x<0)
    x = -x;  // obrati znamenko pokud je x<0
    cout  <<  "|x|=" << x << endl;
  return 0;
}
```

příkaz if - else

```
if (výraz) prikaz_a else příkaz_b
```

- když je podmínka za `if` splněna provede se `příkaz_a`, jinak se provede `příkaz_b`
- příkazy `if` je možné do sebe vkládat a vytvářet tak složitější větvení

příklad: program vypíše, zde je vložené celé číslo kladné, záporné nebo rovno nule

```
#include<iostream>
using namespace std;
int main(void){
  int i;
  cin >> i;
  if (i>0)
    cout << "Cislo je kladne.";
  else
    if (i<0)        // tento test probehne pouze pokud i<=0
      cout << "Cislo je zaporne.";
    else
      cout << "Cislo je rovno nule.";
  cout << endl;
}
```

totéž bez použití `else`

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

```
int main(void){
  int i;
  cin >> i;
  if (i>0) cout << "Cislo je kladne.";
  if (i<0) cout << "Cislo je zaporne.";
  if (i==0) cout << "Cislo je rovno nule.";
  cout << endl;
}
```

poznámky:

- první varianta je méně přehledná ale probíhá v ní méně testů - proto může být rychlejší
- místo `if (výraz==0) příkaz` lze použít `if (!výraz) příkaz`
- podobně místo `if (výraz!=0) příkaz` lze použít `if (výraz) příkaz`

podmíněný výraz

```
výraz_podm ? výraz_a : výraz_b
```

použití:

- pokud je podmínka `výraz_podm` splněna (nenulová) vrací `výraz_a`, pokud není splněna vrací `výraz_b`
- srovnejte nalezení většího ze dvou čísel využitím příkazu `if`

  ```
  if(a>b) max=a; else max=b;
  ```

  a podmíněného výrazu provádějící totéž

  ```
  max=(a>b)?a:b;
  ```

cyklus while

```
while (výraz) příkaz
```

- příkaz se provádí dokud je splněna podmínka za `while`
- tělo cyklu může obsahovat jen jeden příkaz (i složený)
- typicky používáme pokud dopředu nevíme, kolikrát cyklus proběhne viz. dále příkaz `for`

příklad: program opíše zadaný text, všechna malá písmena nahradí velkými

```
#include<iostream>
using namespace std;
int main(void){
  char znak;
  while((znak = cin.get()) != '\n'){
    //cyklus konci ctenim znaku konce radku
    if(znak <= 'z' && znak >= 'a')  // test na male pismeno
      znak += 'Z'-'z';    // prevod maleho pismene na velke
    cout << znak;
  }
  cout << endl;
}
```

příklad: program počítá velká a malá písmena ve vstupním textu

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

```
  int main(void){
  int znak,velkych,malych;
  malych=velkych=0;        // inicializace
  while((znak = cin.get()) != '\n'){
    if(znak>='a' && znak<='z') malych++;   //pocita mala pismena
    if(znak>='A' && znak<='Z') velkych++;  //pocita velka ...
  }
  cout << "V textu bylo " << malych <<  " malych a " << velkych <<
" velkych pismen.\n";
}
```

cyklus s podmínkou na konci

`do příkaz while (výraz)`

- při nesplnění podmínky program opouští cyklus
- cyklus proběhne vždy alespoň jednou!
- méně přehledný než cyklus `while`

cyklus for

`for (výraz_ini;výraz_podm;výraz_iter) příkaz`

- typicky pro cykly kde známe dopředu počet průběhů cyklem
- příkaz `for` postupně provádí následující operace:
  1. před vstupem do vlastního cyklu je vyhodnocen `výraz_ini` - obvykle zde nastavíme řídící proměnnou cyklu na počáteční hodnotu
  2. je vyhodnocena podmínka `výraz_podm` - není-li splněna (má nulovou hodnotu), cyklus končí
  3. je proveden `příkaz` tvořící tělo cyklu
  4. je vyhodnocen `výraz_iter` - obvykle použijeme ke změně (inkrementaci) řídící proměnné cyklu
  5. nová iterace cyklu začíná krokem 2
- kterýkoli z výrazů `výraz_ini,výraz_podm,výraz_iter` může být vynechán, je ale nutné ponechat příslušný středník

příklad: program vypíše ASCII tabulku v rozsahu 32 - 127

```
#include<iostream>
using namespace std;
int main(void){
  int i;
  char j;
  for(i = 32; i < 128; i++){ // "i" se meni od 32 do 127 po jedne
    j = i;
    cout << i  << '\t' << j << endl; // interpretuje "i" jako int
nebo char
  }
}
```

operátor čárky

`výraz_a , výraz_b`

- `výraz_a` je vyhodnocen, následně je vyhodnocen `výraz_b` a předán jako výsledek výrazu s operátorem čárky
- používá se zejména v řídící části cyklů, kde je možno provést více inicializací popř. inkrementací najednou

příklad: program výpočítá faktoriál celého čísla, hlásí chybu pro záporná čísla

```
#include<iostream>
using namespace std;
int main(void){
  int cislo,i;
  long faktorial;  // faktorial rychle roste s argumentem
  cout << "zadej cele cislo: ";
  cin >> cislo;
  if(cislo < 0)
    cout << "Chyba: zadane cislo je zaporne!\n";
  else{
    for(faktorial = 1, i=2; i<=cislo; i++) // byl pouzit operator
carky
      faktorial = faktorial*i;        // take mozno faktorial*=i;
    cout << cislo << "! = " << faktorial << endl;
  }
}
```

změna chování cyklů

- `break` – ukončuje příslušnou smyčku, program opouští cyklus
- `continue` – přeskočí všechny následující příkazy v těle cyklu, cyklus pokračuje další iterací

příklad: program vynásobí deset zadaných celých čísel, pokud je některé nula je cyklus ukončen

```
#include<iostream>
using namespace std;
int main(void){
  int i,cislo;
  long soucin;
  cout << "zadej deset cisel: \n";
  soucin = 1;        // inicializace na "neutralni" hodnotu
  for(i = 0; i < 10; i++){  // deset cisel = deset smycek
    cin >> cislo;
    soucin = soucin * cislo;
    if(cislo == 0) break;  // ukoncuje cyklus s nulovym vysledkem
  }
  cout << "zpracovano " << i << " nenulovych cisel\n";
  cout << "soucin = " << soucin << endl;
}
```