Podmínky a cykly

13. listopadu 2011

Mezi první algoritmické pomůcky patří podmínky a cykly. Podmínky slouží k definici příkazů, které se mají provést pouze tehdy, pokud je splněna daná podmínka. Cykly se používají k opakování určité skupiny příkazů opět pokud je určitá podmínka platná.

Podmínky

Nejjednodušším typem podmínky je podmínka `if`, česky „když“:

```
if ( výraz ) {
   příkaz1;
   příkaz2;
}
```

V tomto případě se příkazy uzavřené ve složených závorkách provedou pouze tehdy, pokud je *výraz* vyhodnocen jako pravdivý. Výraz představuje nejčastěji nějaké porovnání. Máme-li definované proměnné `a`, `b` a `c`, můžeme se ptát zda:

slovní vyjádření	zápic v C	matematický zápis
a rovná se b	`a==b`	$a = b$
a nerovná se b	`a!=b`	$a \neq b$
a je větší než b	`a>b`	$a > b$
a je menší než b	`a<b`	$a < b$
a je větší nebo rovno b	`a>=b`	$a \geq b$
a je menší nebo rovno b	`a<=b`	$a \leq b$
a je větší než b a zároveň menší než c	`(a>b)&&(a<c)`	$a > b \wedge a < c$
a je větší než b nebo menší než c	`(a>b)||(a<c)`	$a > b \vee a < c$

V tabulce jsou uvedené logické výrazy, o kterých je vždy možné rozhodnout, zda jsou pravdivé či nikoliv. V programovacím jazyce C je možné rozhodnout i o samotné hodnotě celočíselných či reálných proměnných, zda je pravdivá či nikoliv. **Nulová hodnota se považuje za nepravdivou, všechny ostatní za pravdivé.** Z toho vyplývá, že podmínka `if (a)` má stejný význam jako `if (a!=0)` a naopak podmínka `if (!a)` má stejný význam jako `if (a==0)`.

Pokud se má při splnění podmínky provést jen jeden příkaz, není nutné ho uzavírat do složených závorek, ale bývá zvykem psát ho hned vedle podmínky:

```
if ( výraz ) příkaz;
```

Pokud se mají nějaké příkazy provést pouze pokud podmínka splněna není, uvedeme hned za pravou složenou závorkou ukončující tělo podmínky `if` klíčové slovo `else`, za kterým opět uvedeme buď přímo na řádku jeden příkaz nebo více příkazů, které uzavřeme do složených závorek:

```
if ( výraz ) {
   příkaz1;
```

```
   příkaz2;
}
else {
   příkaz3;
   příkaz4;
}
```

V tomto případě se tedy *příkazy 1 a 2* provedou pouze pokud je *výraz* pravdivý a *příkazy 3 a 4* pouze pokud je *výraz* nepravdivý.

Program, kterému zadáme tři čísla a on nám určí, které číslo je největší, bude vypadat takto:

```
#include <stdio.h>

main (void)
{
  double x, y, z, max ;

  printf ("Zadej tri cisla:\n");
  scanf ("%lf %lf %lf", &x, &y, &z);

  if (x > y) {
    if (x > z) max = x;
    else max = z;
  }
  else {
    if(y > z) max = y;
    else max = z;
  }

  printf ("Nejvetsi cislo je %g\n", max);
  return 0;
}
```

V některých případech nemá smysl, aby program při splnění podmínky dále pokračoval. Elegantní způsob, jak program ukončit, je zavolání příkazu `return 0;`. Tímto příkazem program vždy skončí.

```
if (a<0) {
  printf{ ‘‘Pro zaporna cisla neni vypocet definovan.\n’’ };
  return 0;
}
```

V tomto případě se tedy následující příkazy při splnění podmínky neprovedou a není nutné ani používat klíčové slovo `else`.

Cykly

Cykly slouží k definici příkazů, které se mají opakovat, dokud platí daná podmínka. Nejběžnějším typem cyklu je tzv. cyklus for. Tento cyklus se používá tehdy, když předem známe, kolikrát má cyklus proběhnout. K odpočítávání jednotlivých cyklů používáme pomocnou celočíselnou proměnnou. Schéma cyklu for vypadá následovně:

```
for ( inicializace ; podmínka platnosti ; krok ) {
   příkaz1;
```

```
   příkaz2;
}
```

inicializace je příkaz, ve kterém nastavíme počáteční hodnotu proměnné, pomocí které budeme cykly počítat. *podmínka platnosti* je podmínka, která definuje, do kdy se má cyklus opakovat. Pozor, nejedná se o ukončovací podmínku, ale o podmínku, kdy ještě se má cyklus opakovat. *krok* je opět příkaz, kterým definujeme, jak se má po uplynutí cyklu změnit hodnota pomocné proměnné.

Program, který vypíše hodnotu čísla od 1 do 10 bude s využitím cyklu for vypadat následovně:

```
#include<stdio.h>

main (void)
{
  int i;

  for ( i=1; i<11; i++ ) {
    printf( "%d ", i );
  }
  printf( "\n" );

  return 0;
}
```

V případě, že chceme vytvořit cyklus, u kterého neznáme počet opakování předem, ale vyplyne až z nějakého složitějšího výpočtu, je vhodné použít cyklus while. Schéma takového cyklu vypadá následovně:

```
while ( podmínka platnosti ) {
   příkaz1;
   příkaz2;
}
```

V tomto případě se příkazy provedou pouze tehdy, když je podmínka platnosti splněna. Pokud tomu tak již od počátku není, neprovedou se vůbec. Pokud ovšem potřebujeme, aby se příkazy nejprve provedly a pak teprve aby se vyhodnotila podmínka platnosti, můžeme použít třetí typ cyklu do-while, jehož schéma je:

```
do {
   příkaz1;
   příkaz2;
}
while ( podmínka platnosti );
```

V některých případech se může stát, že z nějakého důvodu můžeme potřebovat cyklus ukončit předčasně. V takovém případě můžeme použít příkaz `break;`, který cyklus v daném okamžiku ukončí. V případě dvou do sebe vnořených cyklů ukončí jen ten vnitřní.

Switch

V případě, že bychom potřebovali mnohonásobnou podmínku, která by rozlišovala více různých situací a v každé by spouštěla příslušné příkazy, potřebovali bychom použít `if` vícekrát za sebou, což by nebylo právě přehledné. Pro takové situace používáme podmínku switch. Její použití si ukážeme na situaci, kdy se mají různé příkazy provádět pro různé hodnoty proměnné `j`:

```
int j;
switch(j){
  case 0:
    prikaz1;    //tento prikaz se provede pouze pokud j==0
    break;
  case 1:
    prikaz2;    //provedou se všechny následující příkazy,
    prikaz3;    //dokud se nenarazí na prikaz break
    break;
  case 2:
    prikaz4;
  case 3:       //protoze za case 2 neni uveden prikaz break
    prikaz5;    //vykona se pro hodnoty 2 a 3 totez
    break;
  default:      //tuto vetev obvykle piseme i kdyz je prazdna
    break;
}
```

Tato podmínka přehledně specifikuje, že příkaz 1 se má provést pouze pokud $j = 0$, příkazy 2 a 3 se provedou pouze pokud $j = 1$, příkaz 4 se provede pokud $j = 2$ a příkaz 5 se provede tehdy, pokud $j = 2$ nebo $j = 3$. Pokud se j nerovná žádné z vyjmenovaných hodnot, neprovede se nic, protože ve větvi default žádné příkazy nejsou.

Pro rozhodování pomocí switch se většinou používají proměnné typu `int` a nebo `char`, který je použit v ukázkovém příkladu `kalkulacka.c`.