štruktúrované príkazy – Vetvenie

Pri programovaní budeme často riešiť situácie, keď sa niektoré príkazy budú vykonávať len pri splnení istých podmienok. **Podmienkou** budeme rozumieť **logický výraz**, ktorého výsledkom je buď **pravda – TRUE** alebo **nepravda – FALSE**. Podmienka (logický výraz) je splnená, ak nadobúda hodnotu pravda, inak je podmienka nesplnená. Takéto situácie budeme riešiť pomocou vetvenia – podmienenými príkazmi **`if`** a **`case`**.

Najskôr sa budeme venovať podmienkam, teda logickým výrazom. Existujú dva spôsoby vytvorenia logického výrazu:

1. **Jednoduché logické** výrazy vzniknú **porovnávaním** aritmetických výrazov pomocou relačných operátorov. **Relačné operátory** sú: =, <>, <, <=, >, >=.
   Pozor! Ako vidíme, niektoré relačné operátori zapisujeme inak, ako v matematike.

   Takto vyzerajú zápisy niekoľkých jednoduchých logických výrazov:

   - `X > 100`
   - `I mod 5 = 0`
   - `I div 10 >= 1`
   - `A > B`
   - `1 + 1 = 2`

   Výsledok podmienky väčšinou závisí od nejakých premenných (napr. **X** > 100), ale niektoré podmienky sú vždy nepravdivé (napr. 3 mod **X** < 0), iné sú zas pravdivé vždy (napr. **X** = **X**).

2. **Zložené logické výrazy** sa skladajú z dvoch, prípadne viacerých jednoduchých logických výrazov. Tie spájame pomocou **logických operátorov** - **`and`**, **`or`**, a zátvorkami **()**.
   Existuje aj takzvaný **unárny logický operátor `not`**, ktorý vkladáme pred výraz. Má za úlohu negovať ho. Ak bol pôvodný výraz pravdivý, výsledný výraz bude nepravdivý a naopak. Napríklad: `not (12 > 5)`.

   Nech A a B sú logické výrazy. Nasledovná tabuľka zobrazuje výsledky použitia logických operátorov `and, or, not`.

A	**B**	A and B	A or B	not A
True	**True**	True	True	False
True	**False**	False	True	False
False	**True**	False	True	True
False	**False**	False	False	True

   Pri vyhodnotení výrazu platí nasledovné poradie:

   1. funkcie
   2. ( ),
   3. not(),
   4. * / div mod and
   5. + - or
   6. < > = <> ...

   Takto vyzerajú zápisy niekoľkých zložených logických výrazov:

   - `(X=8)and(Y>0)`
   - `(X+Y=0)and(Y<8)`
   - `(x>0)and(x<10) v matematike je to x∈(0,10)`

Pozor na nesprávne zápisy!

- `1 < 5 and 6 > 3`

Poradie pri vyhodnocovaní

1. `1 < 5 and 6 > 3`
2. `1 < 5 and 6 > 3`
3. `1 < 5 and 6 > 3`

Vysvetlenie: Ako vidíme v tabuľke priorít, `and` má väčšiu prioritu, ako < a >. Preto počítač najprv vyhodnotí `and`, až potom < a >. To je ako keby ste chceli aby počítač vyhodnotil pravdivosť výroku 5 `and` 6 Prioritu jednotlivých výrazov v zloženom logickom

výraze treba upraviť zátvorkami: `(1 < 5) and (6 > 3)`

- `1 + X < 5 or 6 - X > 0`

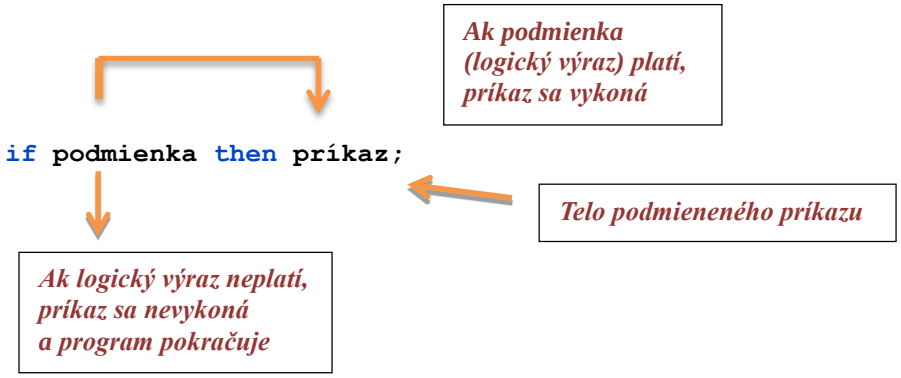
Vysvetlenie: `or`, `+` a `-` majú rovnakú prioritu, ale vyššiu, ako < a >. Teda výrazy sa vyhodnotia v poradí `1+X, 5 or 6, 6-X`. Až potom sa porovnajú, čo je nemožné. Preto musíme tento zložený logický výraz upraviť pomocou zátvoriek: `(1 + X < 5) or (6 - X > 0)`

Úloha

- V matematike používame takýto zápis x<=y<=z. Prečo je takýto zápis v pascale nesprávny?

Podmienený príkaz `if`

Teraz sa naučíme nový príkaz, **príkaz vetvenia** (hovoríme mu aj **neúplný podmienený príkaz**). Jeho zápis je nasledovný:



Inak povedané:

ak je splnená *podmienka*, **tak** vykonaj *príkaz*;

Príklad:

- Príklad zo života:

```
if prsi then zober_si_dazdnik;
chod_na_prechadzku;
```

- Zostavme program, ktorý načíta číslo A a zistí, či je párne.

```
var a:integer;
begin
     write ('Zadaj číslo:  ');
     readln (a);
     if (a mod 2 = 0) then writeln (a,' je párne');
end.
```

Program by sme mohli upraviť tak, aby sa v opačnom prípade vypísalo, že číslo je nepárne. Potrebujeme povedať, čo sa má vykonať, ***ak podmenka neplatí***, t.j. potrebujeme použiť takzvanú **`else` vetvu** (takéto vetvenie nazývame **úplný podmienený príkaz**).
Zápis bude vyzerať nasledovne:

Inak povedané:

ak je splnená *podmienka*, **tak** vykonaj *príkaz1*, **inak** vykonaj *príkaz2*;

Vždy sa vykoná len jeden z týchto dvoch príkazov a to buď iba ***príkaz1***, ak bola podmienka splnená alebo iba ***príkaz2***, ak bola podmienka nesplnená.
Treba si zapamätať pravidlo, že pred `else` sa bodkočiarka nepíše. Vysvetlite, prečo!

Príklad:

- Príklad zo života:

```
if prsi then zober_si_dazdnik
        else obuj_si_sandale;
chod_na_prechadzku;
```

- A teraz sľúbený program na rozpoznanie párneho a nepárneho čísla:

```
var a:integer;
begin
     write ('Zadaj číslo:  ');
     readln (a);
     if (a mod 2 = 0) then writeln (a,' je párne')
                      else writeln (a,' je nepárne');
end.
```

Zapamätajme si! Ak použijeme `else` vetvu, hovoríme o **úplnom** podmienenom príkaze, v opačnom prípade o **neúplnom**.

Môže sa stať, že potrebujeme zapísať viac príkazov v prvej, alebo v druhej vetve. Použijeme zložený príkaz**. Zložený príkaz** je postupnosť príkazov ohraničených kľúčovými slovami **`begin`** a **`end`** – treba sa na `begin` a `end` pozerať ako na príkazové zátvorky. Taktúto postupnosť príkazov vnímame ako jeden príkaz.

Zápis bude vyzerať takto:

```
if podmienka then begin
                    príkaz1;
                    príkaz2;
                    ...
                   end
              else begin
                    príkaz3;
                    príkaz4;
                    ...
                   end;
```

zložený príkaz

zložený príkaz

Úlohy:

1. Zostavte program, ktorý načíta dve čísla A, B a vypíše väčšie z nich.
2. Zostavte program, ktorý načíta reálne číslo A a vypíše jeho absolútnu hodnotu.
3. Zostavte program, ktorý načíta dve čísla A, B koeficienty lineárnej rovnice (ax+b=0) a vypíše (ak existuje) jej koreň.
4. Zostavte program, ktorý načíta tri čísla X, Y, Z a vypíše maximum z nich.
5. Zostavte program, ktorý zistí, či tri zadané čísla K, L, M – na vstupe, môžu byť veľkosťami strán trojuholníka.
6. Doplňte predchádzajúci program, ak K, L, M môžu byť stranami trojuholníka vypíšte, či to je trojuholník rovnostranný, rovnoramenný, pravouhlý alebo iný.
7. Zostavte program, ktorý vypočíta BMI index a vypíše, či máte alebo nemáte nadváhu. BMI = telesný hmotnostný index, vyráta sa ako podiel hmotnosti v kilogramoch a výšky v metroch na druhú, BMI < 18,5 podváha, 18,5<=BMI<25 normálna hmotnosť, 25<=BMI<30 nadváha, BMI>30 obezita.
8. Zostavte program, ktorý zistí, či je zadané číslo A deliteľné trojkou a či je párne.
9. Zostavte program, ktorý načíta tri čísla A, B, C a vypíše ich od najmenšieho po najväčšie.
10. Zostavte program, ktorý načíta tri čísla A, B, C a usporiada ich od najmenšieho po najväčšie, teda aby platilo A<=B<=C.
11. Zostavte program, ktorý načíta tri čísla A, B, C a vypíše, či sú všetky tri rovnaké, či len dve, alebo sú všetky tri rôzne.
12. V meste je kino a divadlo. Divadlo hrá okrem pondelka vždy od 20 hodiny. Kino hrá každý deň o 17. a o 20. hodine. Lístok do kina stojí 7€ a lístok do divadla stojí 18€. Zostavte program, ktorý vám odporučí po zadaní vstupných dát – peniaze, deň (1 – 7), čas, ako máte stráviť voľný čas.
13. Zostavte program, ktorým získate pomocou generátora náhodných čísel pravdepodobnosť toho, že v manželstve budú obaja ľaváci. (pravdepodobnosť toho, že muž je ľavák je asi 0,04 a u žien 0,05)
14. Zostavte program na výpočet, na ktorý deň padne Veľkonočná nedeľa. Existuje niekoľko algoritmov, naprogramujte aspoň jeden Na nájdenie popisu algoritmu využite vyhľadávač.
15. Zostavte program, ktorý načíta číslo x (0<x<100) a vypíše gramaticky správne vetu:
    „Na poli sedelo x vrán“, namiesto x bude číslovka slovne.

Zložený podmienený príkaz `case`

Pascal poskytuje ešte jeden iný príkaz vetvenia – príkaz **`case`**. Tento príkaz nám umožňuje zjednodušiť zápis pri väčšom množstve podmienok, ale len v takom prípade, ak všetky podmienky porovnávajú obsah **jedného** (v každej podmienke rovnakého) výrazu, s konkrétnymi hodnotami. Ak má výraz určitú hodnotu, vykoná sa určený príslušný príkaz. Hodnota však musí byť vždy ordinálna – pre každú jej hodnotu vieme určiť nasledovníka a predchodcu (napr. celé číslo, znak).
Príkaz `case` zapisujeme takto:

```
case výraz of
     hodnota1: príkaz1;
     hodnota2: príkaz2;
     hodnota3: príkaz3;
     ...
end;
```

Telo zložného podmieneného príkazu

Príkaz `case` môže mať `else` vetvu. Ak hodnota výrazu nie je medzi uvedenými, tak sa vykoná príkaz v `else` vetve. (Podobne ako v `if` s `else` vetvou.)

```
case výraz of
     hodnota1: príkaz1;
     hodnota2: príkaz2;
     hodnota3: príkaz3;
     ...
else
     príkazZ;
end;
```

Inak povedané:

v prípade, že *výraz* **má**
hodnotu1: vykonaj *príkaz1*;
hodnotu2: vykonaj *príkaz2*;
hodnotu3: vykonaj *príkaz3*;
...
inak
vykonaj *príkazZ*;

- Ukážka:

```
var a:integer;
begin
  writeln('Co nesie mala lod? Zadaj číslo!')
  read(a);
  case a of
    1: writeln('jablko');
    2: writeln('slona');
    3: writeln('vevericku');
    4: writeln('orechy');
    5: writeln('zajacikov');
    6: writeln('mrkvu');
  else
    writeln('nic iné');
  end;
end.
```

Ak chceme uviesť pre jednu hodnotu viac ako jeden príkaz, použijeme zložený príkaz:

```
case výraz of
     hodnota1: begin príkaz1; príkaz2; ... end;
     hodnota2: begin príkaz3; príkaz4; ... end;
     hodnota3: begin príkaz5; príkaz6; ... end;
     ...
end;
```

zložený príkaz

Miesto jednej hodnoty môžeme určiť i viacero naraz, či dokonca celý číselný rozsah.

```
case výraz of
      hodnota1,hodnota2: príkaz1;
     hodnota3..hodnota4: príkaz2;
                hodnota5: príkaz3;
     ...
end;
```

Pre 2 rôzne hodnoty sa vykoná ten istý príkaz

Pre niekoľko za sebou idúcich hodnôt sa vykoná ten istý príkaz

- Príklad:

```
var a:integer;
begin
  writeln('zadaj číslo')
  read(a);
  case a of
    0..9: writeln('menej ako 10');
      10: writeln('číslo sa rovná 10');
  else
    writeln('viac ako 10');
  end;
end.
```

Úlohy

1. Zostavme program, ktorý určí, ktorý je práve deň. Program čaká na zadanie čísla od 1 do 7. Ak zadáme číslo 1, program vypíše pondelok. Ak zadáme 2, vypíše utorok, atď...
2. Načítajte zo vstupu číslo mesiaca (1 až 12). Ak je to celé číslo z intervalu <1..3>, vypíšte '1. stvrtrok', ak je číslo z intervalu <4..6>, vypíšte '2. stvrtrok', ak je číslo z intervalu <7..9>, vypíšte '3. stvrtrok', ak je to číslo z intervalu <10..12>, vypíšte '4. stvrtrok'.
3. Zostavte program, ktorý vypíše zadané číslo x (z intervalu 1..10) rímskymi číslicami.
4. Zostavte program, ktorý pre zadaný rok a mesiac vypíše koľko dní má daný mesiac.
5. Zostavte program, ktorý vypočíta na základe počtu detí v rodine výšku prídavkov na deti za socializmu.
   1 dieťa – 200 Sk
   2 deti – 650 Sk
   3 deti – 1210 Sk
   4 deti – 1750 Sk
   viac detí – 1720 + 350 * (počet detí - 4)
6. Zostavte program na výpočet povrchu a obvodu zvoleného rovinného útvaru. Ponuka napr.: 1 – štvorec, 2 – obdĺžnik, 3 – trojuholník, 4 – lichobežník, 5 - kruh a pod.
7. Zostavme program, ktorý bude fungovať ako kalkulačka. Načítame dve čísla, vyberieme číslo operácie a podľa toho sa zobrazí výsledok. (1 sčítanie, 2 odčítanie, 3 násobenie, 4 delenie). Ošetrite aj tú možnosť, ak sa zadá zlé číslo operácie.
8. Zostavme program, ktorý zo zadanej prejdenej vzdialenosti (môže byť desatinné číslo) vypočíta taxikárovi platbu podľa sadzobníka:
   - 0 - 19 km 0,59 €/km
   - 20 - 39 km 0,56 €/km
   - 40 – 59 km 0,52 €/km
   - 60 km a viac 0,49 €/km

   Sadzba sa mení až po prejdení daného počtu km (20, 40, 60 km), t. j. zákazník, ktorý precestoval 39,9 km má stanovenú sadzbu 0,56 €/km.
9. Zostavte program, ktorý načíta číslo x (0<x<100) a vypíše gramaticky správne vetu:
   „Na poli sedelo x vrán", namiesto x bude číslovka slovne.