Programování 6.L, 2.A

Úkol od 16. 4. do 22. 4.

Funkce

A je tady další důležitá věc – funkce. V Pascalu jsou funkce téměř totéž co procedury, s jedním podstatným rozdílem – funkce dává (vrací) nějakou hodnotu (číslo, znak, řetězec, apod). Dá se říct, že hlavní úkol procedur je udělat nějakou činnost, hlavní úkol funkce je vrátit nějakou hodnotu. Přitom i z procedury můžeme dostat hodnotu a naopak funkce může dělat nějakou činnost. Ale není to jejich hlavní úkol.

Syntaxe zápisu funkce

```
function Sinus(uhel:real):real;
var rad:real;
begin
  rad:=uhel*pi/180;
  Sinus:=sin(rad);
end;
```

1) klíčové slovo `function`
2) název funkce `Sinus`
3) parametr volaný hodnotou `(uhel:real)`
4) typ hodnoty funkce `:real`
5) lokální proměnná `var rad:real`
6) výpočet lokální proměnné
   `rad:=uhel*pi/180`
   (převedení stupňů na radiány)
7) určení hodnoty naší funkce
   `Sinus:=sin(rad)`
   (tuto hodnotu funkce vrátí do programu)

Z příkladu je vidět, že zápis funkce je podobný zápisu procedury. Rozdíly jsou dva:

- v hlavičce funkce použijeme slovo `function` a za dvojtečku dáme typ funkce (typ vrácené hodnoty)
- v těle funkce **musí být použit název funkce** a do něj **vložena nějaká hodnota** (stejného typu jako v hlavičce)

Co se používá úplně stejně v proceduře a ve funkci, jsou parametry volané hodnotou a parametry volané odkazem, a stejný je i způsob práce s lokálními proměnnými.

Volání funkce v programu

```
begin
  writeln(Sinus(45));
end.
```

1) použijeme název funkce `Sinus`
2) do závorky dáme hodnotu úhlu ve stupních `(45)`
3) funkce vypočte hodnotu a vrátí ji programu
4) program hodnotu vytiskne příkazem `writeln`

nebo

```
begin
  x:=45;
  writeln('Sinus 45° je ',Sinus(x):0:2);
end.
```

Totéž, jen „hezčí“ tisk a použití proměnné.

Využívání funkcí hodně vylepšuje programy, existují dokonce programovací jazyky, které používají jenom funkce (tzv. funkcionální programování). Proto se funkce budeme snažit dobře naučit a procvičit na množství úkolů.

Úkoly

1) Opět se podíváme na základní početní operace. Uvedu jednoduchý příklad programu na sčítání, úkolem je vytvořit obdobné programy s funkcemi na odčítání, násobení a dělení.

```
program Scitani;
uses Crt;
var x,y: real;

function Soucet(a,b:real):real;
begin
  Soucet:=a+b;
end;

begin
  clrscr;
  write('Zadej 1. sčítanec: ');
  readln(x);
  write('Zadej 2. sčítanec: ');
  readln(y);
  writeln('Součet je ',Soucet(x,y):0:3);
  readln;
end.
```

2) Vytvořit program s funkcí `delka`, která bude dělat totéž jako pascalovská funkce `length`, tj. vrátí délku řetězce. (Anglické názvy pascalovských funkcí si můžeme předělat na české názvy.)

3) Vytvořit funkci `prvnipismeno`, která ze zadaného řetězce vrátí první písmenko (znak).

4) Zdrojový kód programů **pošlete** na známý e-mail nejpozději **ve středu 22. 4.** Kdybyste si nebyli jisti, pište během týdne dotazy.

Zhodnocení minulého úkolu – „záhadná“ procedura

Jednalo se o proceduru, která zarovnává nadpis doprostřed, stejně jako v předchozím úkolu procedura `Nadpis`. Jen to dělá jiným způsobem, a to pomocí **rekurze**, neboli voláním sama sebe, přitom při každém dalším volání zvětšuje text nadpisu o jednu mezeru vpředu a jednu mezeru vzadu, tak dlouho, až nadpis dosáhne více než 78 znaků (79 u lichého počtu znaků, 80 u sudého počtu znaků). Pak se nadpis vytiskne a bude opticky uprostřed řádku (za předpokladu, že obrazovka má na šířku 80 znaků).

Vidíme, že i v Pascalu můžeme používat rekurzi = procedura nebo funkce volá sama sebe. Na rozdíl od Karla můžeme při rekurzi využít parametry procedur, což může mít velkou sílu. S rekurzí se dají dělat zajímavá kouzla, jen je třeba být opatrný. Připomínám, že rekurze jsou tři druhy:

- *nekonečná* – volání bez podmínky, opakuje se do nekonečna, nepoužívat!
- *nepravá* – volání s podmínkou, dá se jednoduše nahradit cyklem (náš příklad)
- *pravá* – volání s podmínkou a navíc za voláním se použijí další příkazy, které se provádějí při „zpětném“ ukončování procedur, nedá se jednoduše nahradit cyklem

Když se podíváte na „záhadnou“ proceduru, která dělá totéž jako procedura Nadpis, kterou jste posílali předtím (s výpočtem počtu mezer), možná vás překvapí, jak je díky rekurzi jednoduchá a že nic nepočítá. Z toho plyne, že používání rekurze není jen jiný způsob téhož, ale že je to svou podstatou **jiný způsob programátorského myšlení**.

Další lekci a úkoly zveřejním na webu školy ve čtvrtek 23. 4.

Filip Andziol