Programování 6.L, 2.A

Úkol od 2. 4. do 8. 4.

Globální proměnné, lokální proměnné a parametry procedur

Jakmile používáme procedury, je třeba začít rozlišovat proměnné.

Globální proměnné – deklarují se v záhlaví programu (mimo procedury) za slovem `var` a platí pro celý program, tzn. i ve všech procedurách (pokud lokální proměnná nemá stejný název). Nedoporučuje se používat globální proměnné uvnitř procedur – procedura je pak hůř přenositelná do jiných programů, nebo je u velkých programů problém s hledáním, ve které proceduře se nějaká globální proměnná změnila. Globální proměnné použijme jen v těle hlavního programu, ne v proceduře.

Lokální proměnné – deklarují se v proceduře (pod hlavičkou procedury) opět za slovem `var`. Platí jen pro tu jednu proceduru. Jakmile se procedura dokončí, lokální proměnná „zmizí“, včetně hodnoty, názvu a místa v počítačové paměti. Lokální proměnné používáme jako pomocné proměnné v proceduře. V různých procedurách mohou mít lokální proměnné stejný název, navzájem se neruší. Má-li lokální proměnná stejný název jako globální proměnná, nepřepíše v programu hodnotu globální proměnné, ale pro danou proceduru „zakryje“ globální proměnnou a procedura „vidí“ jen tu svou lokální proměnnou.

Parametry procedur – jsou proměnné pro přenos hodnot mezi programem a procedurami. Když si promyslíte hlouběji, co jsem napsal o globálních a lokálních proměnných, zjistíte, že program nemá jak vložit hodnotu do procedury nebo hodnotu z procedury vytáhnout (nechceme-li se prohřešit proti pravidlu, že se v proceduře nemají používat globální proměnné). Existují dva druhy parametrů – *parametr volaný hodnotou* a *parametr volaný odkazem*.

Parametr volaný hodnotou

Deklarujeme do závorky za název procedury s označením typu proměnné.

např. `procedure Sinus(`*`uhel`*`: real);`

Proceduru pak v programu voláme jejím názvem a závorkou, do které napíšeme buď přímo nějakou hodnotu nebo proměnnou stejného typu jako má parametr.

např. `Sinus(45);` nebo `Sinus(`*`x`*`);` kde *x* je proměnná typu real a má nějakou hodnotu (třeba *x*=45)

Do procedury vstoupí hodnota z programu. Procedura si s touto hodnotou může dělat, co chce (použít k výpočtu, jakkoli změnit, …), a přitom se navenek v programu nestane nic, hlavní program to „nevidí“.

Parametr volaný odkazem

Deklarujeme opět v závorce za názvem procedury, ale před název parametru dáme slovo `var`.

např. `procedure Sifra(var slovo: string[20]);`

Proceduru v programu voláme opět názvem se závorkou, do závorky ale nemůžeme dát hodnotu, musí tam být název nějaké proměnné –

např. `Sifra(x);` kde *x* je proměnná typu string[20] (pozor, musí být stejný typ!)

Do procedury tímto způsobem vstoupí hodnota proměnné z programu, ale cokoliv procedura udělá s touto hodnotou, stane se i v hlavním programu s původní proměnnou.

Pokud jste oba parametry dobře pochopili, víte že:

- parametr volaný hodnotou slouží pouze pro vložení hodnoty z programu do procedury
- parametr volaný odkazem slouží pro vložení hodnoty do procedury a zároveň pro výstup hodnoty z procedury zpět do programu

Parametrů může mít procedura více, viz příklad.

Příklad:

```
program ScitaniDvouCisel;
var a,b,s:real;                          {globální proměnné a, b, s typu real}

procedure Secti(x,y:real;var z:real);    {deklarace procedury Secti
begin                                     s dvěma parametry x, y volanými hodnotou
  z:=x+y;                                 a jedním parametrem z volaným odkazem}
end;                                      {x a y se sečtou a uloží do z}

begin                                    {začátek hlavního programu}
  write('Zadej prvni cislo: ');
  readln(a);                             {načtení hodnoty a}
  write('Zadej druhe cislo: ');
  readln(b);                             {načtení hodnoty b}
  Secti(a,b,s);                          {volání procedury Secti}
  writeln('Soucet je ',s);          {výpis výsledku – s se vypočetlo v proceduře}
end.
```

Fungování: V programu uživatel zadá hodnoty *a* a *b*. Pak se volá procedura a do ní se hodnoty *a* a *b* vloží pod názvem *x* a *y*. Ty se sečtou a výsledek se vloží do proměnné *z*, která díky tomu, že je to parametr volaný odkazem slouží jako „zástupce“ proměnné *s* v hlavním programu. Jakmile se změní *z*, změní se také *s* a hlavní program se dozví, kolik je součet, který vypíše na obrazovku.

Úkoly

1) Vytvořit proceduru `Nadpis`, do níž jako *parametr volaný hodnotou* vstoupí text nadpisu. Procedura pak na obrazovku vypíše daný text vodorovně zarovnaný na střed obrazovky. V proceduře bude třeba použít funkci `length` pro zjištění délky textu nadpisu. Komu se nechce počítat políčka obrazovky, tomu prozradím, že v Pascalu (v DOSu) se na šířku vejde 80 znaků. Samotné zarovnání se provede primitivně napsáním správného počtu mezer před text nadpisu. Kromě toho by v proceduře `Nadpis` mohly být i různé oddělovací čáry nebo jiné okrášlení, ale nemusí – hlavně když tam bude ten text zarovnaný na střed. Proceduru pak můžete nakopírovat do různých programů, co jsme dělali, a na jejich začátek dát volání procedury s příslušným textem nadpisu, např. `Nadpis('SIFROVANI').`

2) Předělat/vytvořit program pro 4 základní početní operace sčítání, odčítání, násobení a dělení tak, aby každá operace se počítala ve své proceduře a hodnoty získala pomocí parametrů, podobně jako v uvedeném příkladu. U dělení nezapomeňte ošetřit případ dělení nulou.

3) Zdrojový kód obou programů (např. soubory *nadpis.pas*, *operace.pas*) mi pošlete na e-mail an@glp.cz nejpozději **do středy 8. 4.**

Zhodnocení minulého úkolu – tři programy předělané do procedur

Z poslaných řešení bych řekl, že nebyl problém. Jen opakuji, že kdo si něčím není jistý, může mi psát na e-mail i během týdne.

Další lekci a úkoly zveřejním na webu školy ~~ve čtvrtek 9. 4.~~ v úterý 14. 4.

Filip Andziol