Mgr. Jana Machová, RNDr. Mária Spišáková

PROCEDÚRY A FUNKCIE V PASCALE

Zbierka úloh

OBSAH

PREDHOVOR

Kniha, ktorú čítate, je zbierkou riešených príkladov z programovania v jazyku Turbo Pascal. Chce byť ukážkou spôsobu pri postupovaní v štúdiu tohto programovacieho jazyka. Vznikla z dôvodov „vákua“ podobne spracovaných tém na slovenskom knižnom trhu. V literatúre s podobnou problematikou, ktorá sa nám dostala do rúk, sme sa často stretali s riešenými úlohami, ktoré odpudzovali svojou veľkosťou. Boli príliš dlhé a tým aj málo čitateľné. Často aj v malých programoch bolo zbytočne veľa identifikátorov, ktoré tiež prispievajú k neprehľadnosti riešenia.

Programátorský štýl je veľmi dôležitý hlavne v pedagogickej praxi:

- Spôsob, akým je program napísaný ovplyvňuje nielen jeho čitateľnosť, pochopenie a jeho použiteľnosť, ale ovplyvňuje najmä programátorský štýl študentov. Ak učiteľ nepíše svoje vzorové riešenia príkladov štrukturovane, nemôže očakávať od študentov dodržiavanie tohto štýlu.
- V našej pedagogickej praxi sa často stretávame pri výuke programovania so zaradením štruktúr, t.j. procedúr a funkcií, až do záverečných fáz tématických plánov. Chceme poukázať na to, že programovanie pomocou štruktúr v začiatočných fázach štúdia uľahčuje tvorbu algoritmov a rozširuje spektrum nástrojov programovania v ďalšom štúdiu.
- Taktiež chceme poukázať na vlastnosť modulárneho a štrukturovaného spracovávania čo aj jednoduchých problémov a úloh a ich využiteľnosť pri preberaní ďalších tém. K tomu prispievajú aj vhodne pomenované identifikátory. Študent si postupne buduje vlastnú knižnicu procedúr a funkcií, ktoré môže, vďaka ich univerzálnemu zápisu, kedykoľvek použiť v programe resp. inom module.

Riešenia úloh sme vo väčšine príkladov obohatili o stručný komentár. Myslíme si však, že v niektorých úlohách nie je vôbec potrebný. Obtiažnosť úloh v jednotlivých kapitolách je vyznačená nasledovne:

- jednoduchá úloha
- ●● úloha predpokladá základné vedomosti z danej oblasti
- ●●● zložitejšia úloha vyžadujúca podrobnejšie vedomosti

Za každým príkladom sú uvedené otázky, ktoré navodzujú témy na zamyslenie a hlbšie rozoberajú daný príklad. Riešenia príkladov sú v tvare samostatných procedúr, niektoré majú tvar úplného programu. Jednotlivé tématické celky sú zakončené zadaním ďalších úloh, ktoré rozširujú danú tému a môžu slúžiť ako zásobník príkladov. Algoritmy

uvedené v knihe sú vložené do užívateľských knižníc podľa témy. Sú to knižnice s názvami CISLA, BOOLOVSKE, PREVODY, POLIA, MATICE A RETAZCE, ktoré sú obsahom príloh.

Zbierka je vhodná pre učiteľov programovania na stredných školách a tiež pre študentov a iných záujemcov o programovanie. Môžu ju študovať aj bez pomoci učiteľa. (Na nich sme mysleli pri niektorých rozsiahlejších komentároch.) Obtiažnosť preberanej problematiky zodpovedá úrovni základného kurzu programovania na stredných školách. Pri výuke na škole by zbierka mala slúžiť ako doplnok a protipól nejakej vhodnej učebnice programovania.

V prílohách okrem odladených užívateľských knižníc uvádzame návrh niekoľkých didaktických testov, ktoré uvítajú učitelia, ale aj študenti.

Čitateľov prosíme pri štúdiu o zhovievavosť pri posudzovaní prípadných nedokonalostí resp. omylov, na ktoré narazia pri používaní tejto publikácie.

Ďakujeme všetkým, ktorí priamo i nepriamo ovplyvnili vznik tejto knihy .

Prajeme čitateľom veľa úspechov pri hľadaní ešte jednoduchších riešení.

Autorky.

ÚVOD

Sú dve cesty ako pristupovať k písaniu počítačových programov:
prvá - vypracovať ich tak jednoducho, že v nich zrejme nie sú žiadne nedostatky;
druhá - vypracovať ich také zložité, že v nich nie sú žiadne nedostatky zrejmé.
T.Hoare

Dôvodom pre vybratie témy procedúry a funkcie vo vyučovaní Pascalu na SŠ bol fakt, že táto téma sa nám javí trochu podcenená, ale pri tom veľmi dôležitá pre vytvorenie kvalitného programátorského štýlu. Aj jednoduché programy sa dajú zostavovať štrukturovane už v úvode kurzu programovania.

Prečo zamerať pozornosť na procedúry a funkcie? Sú to prvky jazyka Pascal, ktoré umožňujú programovať štrukturovane a modulárne. Hovorí sa, že ak dáme naprogramovať nejaký problém dvadsiatim programátorom, tak dostaneme 20 rôznych programov. Nejde nám o to, aby všetci programovali rovnako, ale to, aby programy napísané žiakmi a najmä učiteľmi boli čitateľné, prehľadné a nekomplikované a tým aj správne.

Prečo je užitočné tvoriť vlastné programy štrukturovane? Pri ladení programu sa obyčajne vyskytnú očakávané aj neočakávané chyby. Vtedy je potrebné prekontrolovať zdrojový kód programu. Zvyčajne nie je ľahké sa v ňom vyznať a kontrolovať ho. Existujú programy zapísané tak, že zorientovať sa v nich je hazardovaním s duševným zdravím. Naopak existujú algoritmy, ktoré aj keď sú komplikované a rozsiahle, sú zapísané zrozumiteľne a prehľadne.

O ŠTÝLOCH

Je veľmi jednoduché niečo skomplikovať, ale býva veľmi komplikované niečo zjednodušiť.

Mayerov zákon

Každý programátor má svojrázny štýl vytvárania programov, ktorý mu najviac vyhovuje. Tak napríklad programátor amatér väčšinou tvorí tak, že algoritmus objaví niekde vo svojom podvedomí a snaží sa toto „tušenie algoritmu“ zapísať čo najrýchlejšie. Problémy nastávajú, ak takto štýlovo napísaný program zoberie do rúk iný programátor, alebo sám autor po dlhšom čase. Potom už nikto nevie „ o čom to bolo“. (Česť výnimkám - programátorom so sklonmi prehľadne zapisovať algoritmy.) Zápis programu je ťažko čitateľný, je príčinou vzniku chýb a sťažuje ich odstránenie.

Štruktúrované programovanie:

Najznámejšou metodikou podporujúcou dobrý programovací štýl je štruktúrované programovanie. Je to metodika, ktorá pomáha tvoriť prehľadné a zrozumiteľné programy. Môžeme ju charakterizovať nasledujúcimi princípmi:

- Program je treba vytvárať systematicky „zhora nadol“, teda začať návrhom jeho celkovej logiky a postupne sa jeho hrubé črty budú spresňovať v ďalších krokoch.
- Program by mal byť členený na jednotlivé časti (moduly), riešiace ucelené dielčie činnosti, moduly by nemali byť príliš rozsiahle. (Pri tvorbe žiackych programov modul tvorí maximálne 10-15 riadkov zdrojového kódu.)
- V programe používať prostriedky zvyšujúce jeho zrozumiteľnosť(mená identifikátorov, komentáre).

Modulárne programovanie:

Ide o odlišný spôsob programovania ako pri štrukturovanom programovaní („zdola na hor“), hoci vyššie sme použili pojem delenia programu na moduly. Modulárne programovanie využíva už existujúce, alebo dopredu napísané moduly. Telo hlavného programu sa tvorí nakoniec s využitím týchto modulov.

Ideálnou metódou sa zdá byť *prepojenie oboch metód:*

1. možnosť: univerzálne moduly napísať dopredu (napr. načítanie resp. výpis vstupných hodnôt ...) a potom programovať zhora nadol podľa princípov štrukturovaného programovania.

2.možnosť: začať písať hlavný program v tvare postupnosti modulov a tie rozpracovať dodatočne.

V našej práci sme uplatnili obe metódy vytvárania programov resp. samotných modulov. V kapitole *Vektory a matice* sme najprv zostavili podprogramy uložené v knižnici POLIA, viď pr.1., a potom sme ich používali pri tvorbe ostatných podprogramov a programov.

1. POJEM ALGORITMU, ČASOVÁ A PAMÄŤOVÁ ZLOŽITOSŤ

Zložitosť algoritmu rastie, až prekročí schopnosti programátora, ktorý ho musí udržiavať.

Programátorský folklór

Algoritmus chápeme ako konečný súbor pravidiel, ktoré dávajú návod k vyriešeniu určitej triedy úloh. Nedá sa jednoznačne definovať. Popisujeme ho jeho vlastnosťami:

- *rezultatívnosť* - algoritmus speje k správnemu výsledku
- *konečnosť* - algoritmus po konečnom počte krokov skončí
- *hromadnosť* - algoritmus rieši skupinu podobných problémov
- *determinovanosť* – jednoznačnosť výberu nasledujúceho kroku algoritmu
- *elementárnosť* – algoritmus je zostavený z jednoduchých úkonov

Vo vyučovaní programovania na SŠ sa zaoberáme problémami, ktoré sú matematicky alebo logicky formulovateľné a zahrňujú často nekonečný počet konkrétnych prípadov, teda sú hromadné. Pre tieto problémy sa dá po kratšej či dlhšej dobe nájsť algoritmus na ich vyriešenie. Hovoríme, že sú *algoritmicky riešiteľné.*

Existujú však aj problémy, pre ktoré je dokázaná ich algoritmická neriešiteľnosť. Napr. nedá sa vytvoriť algoritmus pre hľadanie chýb v algoritme. Takýmto úlohám hovoríme *algoritmicky neriešiteľné.*

Pri písaní algoritmu je dôležité si uvedomiť jeho časovú a pamäťovú zložitosť, tvoriť algoritmy tak, aby boli ekonomické (šetrili pamäť počítača) a efektívne (šetrili pracovný čas). Nemusí však platiť, že program napísaný zrozumiteľne je efektívny a ekonomický.

Pamäťovú zložitosť algoritmu definujeme ako počet pamäťových miest závislých od rozmeru úlohy, ktoré algoritmus spotrebuje v priebehu vykonávania.

Pod *časovou zložitosťou* rozumieme čas spotrebovaný algoritmom v závislosti od rozmeru úlohy pri riešení problému.

Na ohodnotenie napísaného algoritmu používame kritériá správnosti a časovej zložitosti. Napríklad pre riešenie sústavy lineárnych rovníc sa môžeme rozhodnúť pre dva z viacerých algoritmov:

* Cramerovo pravidlo
* Gaussova eliminačná metóda.

Pre n neznámych Cramerovo pravidlo potrebuje zhruba (n+2)! operácií a Gaussova eliminačná metóda len $n^3/3$ operácií, čo pre 10 neznámych pri Gaussovej metóde počítač vykoná približne 333 operácií a pri Cramerovom pravidle 479 001 600 operácií!

Polynomiálna a exponenciálna časová zložitosť

Prvý algoritmus z predchádzajúceho príkladu má exponenciálnu a druhý polynomiálnu časovú zložitosť, lebo ich odhad zložitosti sa dá vyjadriť ako odpovedajúca funkcia rozmeru úlohy. V algoritmoch, ktorých odhad časovej zložitosti sa vyjadrí ako exponenciálna funkcia, už pre malé hodnoty rozmeru úlohy rastie počet operácií mimoriadne rýchlo.

Napríklad:
Jednoduchší program pre šachovú hru, ktorý v každej situácii preberie všetky možné ťahy, všetky možné súperové reakcie na ne, potom všetky naše odpovede na súperove reakcie, atď, sa dá vyjadriť v tvare „stromu“ možných ťahov. Problém je v tom, že takýto algoritmus vyžaduje mimoriadne veľký počet operácií. Ak by sme v každej situácii mali možnosť sledovať napr. iba 10 nasledujúcich možných ťahov, tak by sme museli preveriť 100 súperových odpovedí, 1000 našich odpovedí atď. Teda pre hĺbku „n“ ťahov potrebujeme spočítať 10^n možností, čo je exponenciálna zložitosť. Takto robiť rozbor ťahu sa dá len do veľmi malej hĺbky (4, 5 ťahov) a to pre kvalitný šachový program nestačí.

Lineárna a logaritmická časová zložitosť

Algoritmy s takouto časovou zložitosťou sú efektívne a ekonomické, ale ich objavenie a zápis nie je často triviálny. Žiaci ľahšie nachádzajú ekvivalentné algoritmy s polynomiálnou časovou zložitosťou, ale očakáva sa od učiteľa prezentovanie už vytvorených algoritmov s lineárnou resp. logaritmickou zložitosťou.

Napríklad :
Študent vie sám vytvoriť algoritmus triedenia postupnosti čísel metódou výberu maximálneho resp. minimálneho prvku, ale na princíp stromového triedenia postupnosti pravdepodobne nepríde bez pomoci učiteľa.

2. PRVÉ KROKY PRI ŠTRUKTUROVANOM PROGRAMOVANÍ

Axiomatická teória programovania:
Definícia 1.: Program je konečná postupnosť príkazov.
Axióma 1.: V každom programe je aspoň jedna chyba.
Axióma 2.: Každý program, ktorý obsahuje viac než jeden príkaz, je možné napísať o jeden príkaz úspornejšie.
Veta 1.: Každý program je možné zkrátiť na jediný príkaz, a ten je chybný.

Programátorský folklór.

Prvé zoznamovanie sa so štrukturovaným programovaním môžeme zaradiť hneď po prvých príkazoch Turbo Pascalu. Potrebné vedomosti sú :

- štruktúra programu v TP
- príkazy vstupu a výstupu
- zápis aritmetických, logických a relačných výrazov
- priorita operátorov a funkcií
- štandardné typy dát

• Príklad 1.: Vytvorte funkciu SUCET pre sčítanie dvoch celých čísel a využite ju v programe.

Poznámka: V deklarácii funkcie sa nebudú vyskytovať parametre a lokálne premenné, funkcia spočítava čísla deklarované v hlavnom programe.

Riešenie:

Najskôr napíšme program postupom „ zhora nadol“.

1. krok:
Ak by počítač rozumel slovensky napísaným „príkazom“, tak v tomto prípade to je „príkaz“ SUCET.

```
program SCITANIE;
var A, B : integer;
begin
    readln (A);
    readln (B);
    writeln(‘ Sucet A, B je :’, SUCET );
```

```
end.
```

2.krok: Sme si vedomí toho, že niekde je potrebné urobiť preklad „príkazu“ SUCET .

```
function SUCET : integer ;
  begin
   SUCET := A+B
  end;
```

3. krok: Zlúčime tieto dva kroky do jedného programu. Je potrebné si uvedomiť štruktúru programu v Pascale a to, že deklarácie funkcií a procedúr sa nachádzajú v časti deklarácií a definícií, teda hneď za hlavičkou programu.

```
program SCITANIE;
  var A, B : integer;
  function SUCET : integer ;
    begin
          SUCET := A+B
    end;
 begin    {hlavný program}
    write (‘ zadaj A’); readln (A);
    write (‘ zadaj B’); readln (B);
    writeln(‘ Sucet je :’, SUCET );
end.
```

Otázky:

- Môžeme v hlavnom programe použiť iné identifikátory premenných ako vo funkcii?
- Ak chceme v hlavnom programe použiť funkciu SUCET viac krát, ale s inými premennými, čo musíme zmeniť? Odpoveď nájdete v kroku 4.

4. krok: Použitie funkcie s parametrami:

```
function SUCET  ( x, y : integer ): integer ;
begin
   SUCET := x + y
end;
```

Parametrom x, y hovoríme, že sú formálne parametre, nahrádzajú lokálne parametre v tele funkcie. V hlavnom programe na výpočet súčtu premenných a, b použijeme *volanie funkcie hodnotou* : SUCET (A, B). Do funkcie sa premenným x, y priradia hodnoty premenných A, B. Premenné A, B aj x, y musia byť rovnakého typu. Premenné x, y už v procedúre, ani nikde inde nedeklarujeme.

5. krok: Hotový program vyzerá:

```
program SCITANIE;
  var A, B : integer;
   function SUCET ( x, y : integer) : integer;
```

```
    begin
      SUCET := x + y
    end;
  begin { hlavný program}
    write (‘ zadaj A’); readln (A);
    write (‘ zadaj B’); readln (B);
    writeln(‘ Sucet je :’, SUCET ( A, B) );
  end.
```

• **Príklad 2.:** Zostrojte program, ktorý z čísel a, b, c, d, načítaných na vstupe vypočíta súčin a*b, c*d.

Riešenie:
Budeme postupovať „zhora nadol“. Najprv hlavný program, a detaily dorobíme neskôr.

1. krok:

```
begin
    zadaj (A);
    zadaj (B);
    zadaj (C);
    zadaj (D);
   writeln ( A,’ * ‘ ,B , ‘ = ‘ , SUCIN (A,B);
   writeln ( C,’ * ‘ ,D , ‘ = ‘ , SUCIN (C,D);
end.
```

2. krok: deklarácia premenných:

```
var A, B, C, D : integer;
```

3.krok: deklarácia procedúry ZADAJ (x).

Otázky:

- Prečo je to procedúra a nie funkcia?
- Musí to byť procedúra s parametrom?

```
Procedure ZADAJ ( x: integer);
begin
   write ( ‘ zadaj  hodnotu’ );
   readln (x)
end;
```

4. krok: deklarácia funkcie SUCIN (x, y: integer);

Otázky:

- Ako vieme, že tento modul má byť funkciou a nie procedúrou?

```
function SUCIN ( x, y : integer) : integer;
begin
   SUCIN := x * y
end;
```

5.krok : spojenie jednotlivých častí do programu:

```
program NASOBENIE;
      var A, B, C, D : integer;
   procedure ZADAJ ( x: integer);
        begin
           write ( ' zadaj  premennú' ); readln (x)
        end;
    function SUCIN ( x, y : integer) : integer;
        begin
           SUCIN := x * y
        end;
begin            {hlavny program}
    zadaj (A);
    zadaj (B);
    zadaj (C);
    zadaj (D);
    writeln ( A,' * ' ,B , ' = ' , SUCIN (A,B);
    writeln ( C,' * ' ,D , ' = ' , SUCIN (C,D);
end.
```

• **Príklad 3.:**Na základe zadanej výšky vkladu v Sk a ročnej úrokovej miery v percentách spočítajte, aký bude zostatok na účte, na konci roka, ak ďalšie vklady neboli?

Riešenie: Programujeme „zhora nadol".

1. krok

```
begin { hlavný program}
    write (' Vyska vkladu? '); readln ( vklad);
    write  (' Urokova miera ? '); readln ( urok);
    writeln ( ' Zostatok na konci roka : ' , ZOSTATOK);
end.
```

2. krok: **var** ..... (deklarácia premenných)

3. krok: deklarácia podprogramu ZOSTATOK. (Je to funkcia alebo procedúra?)

```
function ZOSTATOK ( v: real, u: real ): real;
begin
```

```
    ZOSTATOK : = v * ( 1 + u )
end;
```

4. krok: spojenie do programu

```
program BANKA;
   var .....;
   function ZOSTATOK ( v: real, u: real ): real;
    begin
      ZOSTATOK : = v * ( 1 + u )
   end;
begin { hlavný program}
  write (‘ Vyska vkladu? ‘); readln ( vklad);
  write  (‘ Uroková miera ? ‘); readln ( urok);
  writeln ( ‘ Zostatok na konci roka : ‘ , ZOSTATOK( vklad, urok);
end.
```

JEDNODUCHÉ ÚLOHY:

Snažte sa vytvárať štrukturované programy!

a) Za koľko rokov sa klient banky pri jednorázovom vklade o výške v a úrokovej sadzbe u stane milionárom? Aký musí byť jednorázový vklad, aby sa klient banky stal milionárom pri rovnakej úrokovej sadzbe za r rokov?

b) Údaj v m/s premeňte na km/h.

c) Údaj v sekundách premeňte na hodiny, minúty a sekundy.

d) Do súťaže sa prihlásilo N hráčov. Určte, koľko zápasov je treba zohrať, aby každý hráč hral s každým iba raz.

e) Nádrž má štvorcové dno so stranou veľkosti A. Priteká do nej voda rýchlosťou N litrov za sekundu. Spočítajte, ako dlho musí pritekať voda, aby hladina stúpla o dva metre.

3.TROCHU TEÓRIE

Procedúry a funkcie, nazývané tiež podprogramy, dovoľujú rozčleniť program do prehľadných častí a zavádzať nové užívateľom definované príkazy jazyka. Každá deklarácia podprogramov obsahuje hlavičku funkcie alebo procedúry, deklaráciu lokálnych objektov a telo podprogramu.

Deklarácia procedúry:

```
procedure MENO_PROCEDURY( zoznam formálnych parametrov);      hlavička

  var
  const
  procedure      } deklarácia lokálnych objektov
  function
 type
begin
  .              } telo procedúry
  .
end;
```

Deklarácia funkcie:

```
function MENO_FUNKCIE( zoznam formálnych parametrov) : typ_funkcie;hl
   var
  const
  procedure      } deklarácia lokálnych objektov
  function
  type
begin
  .
  .              } telo funkcie
  .
end;
```

Slová **procedure**, **function** sú vyhradené slová. Rozdiel medzi procedúrou a funkciou je ten, že funkcia vracia ako výsledok jednu hodnotu. Telo funkcie musí obsahovať aspoň jeden priraďovací príkaz s menom funkcie na ľavej strane. Hodnota, ktorá bola pridelená

identifikátoru funkcie ako posledná, je výslednou hodnotou funkcie. Volanie funkcie môže nastať len na takom mieste programu, kde sa očakáva nejaká hodnota.

Volanie podprogramu v programe má tvar:

```
begin
  .
   Meno ( p1,p2,....,pn) ;   . . . . . p 1,p2,...,pn  sú skutočné parametre
   .
end.
```

V hlavičke podprogramu môže zoznam formálnych parametrov chýbať. Potom hovoríme o podprogramoch *bez parametrov.* Napríklad :

```
function AHOJ : integer;
   var  i: integer;
begin
        for i:= 1 to 10 do writeln( ´AHOJ´ )
end.
```

Parametre procedúr a funkcií sú formálne, pretože počas výpočtu sú podľa dohodnutých pravidiel nahrádzané skutočnými parametrami. Môžu byť:

- parametre volané hodnotou
- parametre volané odkazom
- konštantné parametre
- parametre procedurálneho typu
- parametre volané menom
- schéma konformného poľa

a) ***Parametre volané hodnotou.***

Špecifikácia parametrov má tvar :

$p_1, p_2, p_3, \ldots . p_n$: identifikátor typu

Príklad hlavičky:
```
function SUCIN ( x, y : integer) : integer;
procedure VYMENA ( a,b: integer);
```

Tieto parametre sú používané v prípade, že chceme podprogramu odovzdať nejakú vstupnú hodnotu. Zmena tohto parametra nemá vplyv na skutočný parameter. Viď príklad

8. Zo 4. Kapitoly- funkcia na výpočet kombinačného čísla. V skutočnosti sa v procedúre namiesto formálneho parametra počíta s pracovnou premennou, do ktorej sa na začiatku procedúry prekopíruje hodnota skutočného parametra. Na konci procedúry ku spätnému kopírovaniu nedochádza. Skutočné parametre môžu byť premenné alebo konštanty rovnakého typu ako formálne parametre.

b) ***Parametre volané odkazom.***

Špecifikácia parametrov má tvar :

var $p_1, p_2, p_3, \ldots . p_n$: identifikátor typu.

Príklady hlavičiek:

procedure VYMENA_2(**var** a,b: **integer**);
procedure VELKY_FAKTORIAL (n:**word**; **var** p: postupnost) ;

Parametre volané odkazom sú pri volaní nahradzované hodnotou skutočných parametrov a majú charakter výstupu z procedúry. Celý výpočet podprogramu prebieha so skutočnými parametrami, takže každá zmena parametra v podprograme sa ihneď premietne do skutočného parametra. Preto miesto týchto parametrov sa nedajú použiť konštanty.

c) ***Konštantné parametre.***

Špecifikácia parametrov má tvar :

const $p_1, p_2, p_3, \ldots . p_n$.

Príklad hlavičky: **procedure** SINUS (**const** PI).

Ku konštantným parametrom sa prekladač chová ako ku skutočným konštantám, nemôžu stáť na ľavej strane priraďovacieho príkazu.

d) ***Parametre procedurálneho typu.***

Jazyk Pascal umožňuje používať procedúty a funkcie ako parametre iného podprogramu. Nasledujúci program ilustruje použitie funkcií ako parametre procedúry TLAC.

```
program TABULKY;
type funkcia = function ( x , y : integer) : integer ;
function SUCET ( i, j : integer) : integer ;
    . . . .
function SUCIN ( i, j : integer) : integer ;
```

```
        . . . .
    procedure TLAC ( n1, n2 : integer; operacia : funkcia);
        . . . .
begin  { hlavny program }
       TLAC ( 10, 10, SUCET);
       TLAC ( 10, 10, SUCIN );
end.
```

4. PROCEDÚRY A FUNKCIE PRE ČÍSELNÉ VÝPOČTY

S ďalším hlbším oboznamovaním s podprogramami pokračujeme po zvládnutí príkazov vetvenia, cyklov a jedno- a dvoj-rozmerných polí.

Vstupné vedomosti:

- štruktúra podprogramov
- parametre v podprogramoch
- volania podprogramov

S číselnými algoritmami sa stretávame najčastejšie na hodinách matematiky. Naprogramovanie takýchto úloh je preto samozrejmosťou. Sú vhodný prostriedkom, ako vysvetliť štruktúru procedúr a funkcií, ich volaní a parametrov.

• **Príklad 1. :** Vytvorte procedúru na výmenu hodnôt dvoch celočíselných premenných a, b.

Riešenie:

```
procedure VYMENA;
    var pom: integer;
begin
    pom : = a ;
    a : = b ;
    b : = pom ;
end;
```

Otázky :

* Kde môže byť deklarovaná pomocná premenná pom, a kde musia byť deklarované premenné a, b ?
* Môžeme procedúru VYMENA použiť na výmenu hodnôt iných premenných ako sú a,b?

• **Príklad 2.**: Vytvorte procedúru na výmenu hodnôt dvoch ľubovoľných celočíselných premenných .

Riešenie:

```
procedure VYMENA_2( var a,b: integer);
    var pom: integer;
begin
    pom : = a ;
    a : = b ;
```

```
        b : = pom ;
end;
```

Otázky :

- Môžeme slovo *var* v deklarácii parametrov procedúry VYMENA_2 vynechať?
- Dá sa vymeniť obsah dvoch celočíselných premenných bez pomocnej premennej? (Návod na riešenie: a:= a + b; b:= a –b; a:= a –b;)

• Príklad 3.: Vytvorte funkciu, ktorá z dvoch čísel vráti väčšie z nich.

Riešenie:

```
function MAX(a,b: real): real;
begin
   if a > b then   MAX : = a
            else   MAX : = b
 end;
```

Otázky:

- Hodnotu ktorej premennej dá funkcia na výstupe, ak a = b ?
- Môžeme použiť slovo *var* v deklarácii parametrov funkcie MAX ?
- Dá sa napísať funkcia MAX bez použitia príkazu *if* ?
  (Návod na riešenie: MAX = (a+ b + abs(a-b)) / 2)

• Príklad 4.: Vytvorte funkciu, ktorá vypočíta n - tú mocninu reálneho čísla X. ($n \in N$)

Riešenie:

```
function MOCNINA ( n: integer ;  x : real ) : real;
  var p : real ;
begin
    p:=1;
    for  i:=1  to  n  do  p:= p*x ;
     MOCNINA: = p
end;
```

Otázky:

- Dopustili by sme sa chyby pri použití príkazu :
  `for i:=1 to n do MOCNINA:= MOCNINA * x ?`
- Dal by sa výpočet mocniny urýchliť a za akých podmienok?
  (Návod na riešenie: Využite zápis exponentu n ako mocninu čísla 2.)

• Príklad 5.: Vytvorte funkciu na výpočet faktoriálu prirodzeného čísla.

Riešenie :

```
function FAKTORIAL ( cislo : integer) : longint ;
```

```
    var  i: integer ;
         f: longint ;
begin
   f : = 1 ;
   for  i:=cislo downto 2 do  f:= f*i ;
        FAKTORIAL : =  f ;
end;
```

Otázky:

* Dopustili by sme sa chyby pri použití príkazu :
  for i:=cislo downto 2 do FAKTORIAL:= FAKTORIAL * x ?
* Pre aké veľké celé číslo typu *integer* funkcia ešte vypočíta faktorál?

Poznámka: Pre veľké vstupné číslo daný problém rieši funkcia VELKY_FAKTORIAL. (viď ďalej)

• **Príklad 6.:** Vytvorte funkciu na výpočet ciferného súčtu daného celého čísla.

Riešenie: Zo vstupného čísla postupne oddeľujeme cifry pomocou funkcie *mod* a číslo skrátime o poslednú cifru pomocou funkcie *div.*

```
function CIF_SUCET( cislo : integer): integer ;
   var sucet: integer ;
begin
      sucet:=0 ;
      while cislo<>0 do
          begin
              sucet  := sucet + c mod 10 ;
              cislo:= cislo div 10
          end ;
      CIF_SUCET:= sucet
end ;
```

Otázky:

* V ktorých príkazoch by sa líšila funkcia, ktorá by oddeľovala cifry od najvyšších rádov?

• **Príklad 7.:** Vytvorte funkciu na výpočet najväčšieho spoločného deliteľa dvoch celých čísel metódou Euklidovho algoritmu.

Poznámka: Euklidov algoritmus využíva matematickú vetu: pre ľubovoľné dve prirodzené čísla a,b plati, ak a>b potom NSD(a,b) = NSD(a-b,b).

Riešenie: Využíva už definovanú procedúru VYMENA_2

```
function NSD( a,b : integer): integer ;
  begin
    while a<>b do
        begin   if  b>a then VYMENA_2(a,b) ;
                a := a - b
        end ;
     NSD := a
  end ;
```

Otázky:

* Kde musí byť deklarovaná procedúra VYMENA a ktorú verziu môžeme použiť v tele funkcie NSD ?

- **Príklad 8.:** Vytvorte funkciu na výpočet kombinačného čísla (n nad k, kde $n \geq k, n, k \in N$).

Riešenie I.: Využíva už definovanú funkciu FAKTORIAL a matematickú definíciu kombinačného čísla :

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

```
function KOMB_CISLO ( n, k : integer) : longint;
  begin
     KOMB_CISLO:= FAKTORIAL(n) div  FAKTORIAL(k) div
                                    FAKTORIAL(n-k)
  end;
```

Otázka:

* Pre aké najväčšie hodnoty n, k môžeme funkciu KOMB_CISLO ešte volať ?

Riešenie II.: Využíva upravený tvar matematickej definície.

```
function KOMB_CISLO_2 (n , k : integer) : longint;
   var p: longint; i : integer ;
begin
  p:= 1;
  for  i:=n  downto n-k+1 do p : = p * i ;
  KOMB_CISLO_2 : = p  div  FAKTORIAL( k )
end;
```

Otázky:

- Prečo je riešenie II. efektívnejšie ?
- Porovnajte časovú zložitosť oboch riešení.

• Príklad 9.: Vytvorte procedúru pre výpis Pascalovho trojuholníka až do rádu n.

Riešenie I. : Pascalov trojuholník pozostáva z kombinačných čísel $\binom{n}{k}$ kde n >=0, 0 <= k <= n.

```
procedure PASCAL (n : integer) ;
    var i,j: integer;
begin
   writeln ( ' 1 ' );
   writeln ( ' 1   1' ) ;
   for i:= 2  to  n  do
       begin
          write ( ' 1      ' );
           for j : = 1 to  i - 1 do write ( KOMB_CISLO(i , j ) :8 ) ;
           writeln ( ' 1 ' );
       end;
end;
```

Otázky:

- V akom formáte bude táto procedúra vypisovať Pascalov trojuholník ?
- Ako upraviť procedúru, aby bol výpis v tvare rovnoramenného trojuholníka ?
- Ako sa dajú určiť prvky Pascalovho trojuholníka bez použitia výpočtu kombinačných čísel?

Riešenie II. :

Bez použitia výpočtu kombinačných čísel, algoritmus je založený na tvorbe pascalovho trojuholníka t.j. prvky na ramenách trojuholníka sa rovnajú jednej a sčítaním dvoch susedných prvkov v riadku dostaneme prvok medzi nimi o riadok nižšie.

```
procedure QUICKPASCAL( n: integer; var  a:matica);
      var i,j: integer;
 begin
     a[0,0]:=1;  a[1,0]:=1;  a[1,0]:=1;
     for i:= 2 to n do
          begin
```

```
            a[i,0]:= 1;
            for j:=1 to i-1 do a[i,j]:=a[i-1,j-1] + a[i-1,j];
            a[i,i]: = 1
        end;
    PIS_MATICU(a,n,n)
end;
```

Poznámka: Typ matica a procedúra PIS_MATICU(a,n,n) je deklarovaná v 7. kapitole.

Otázky:

- Aké sú výhody riešenia II ?
- Dala by sa zostaviť procedúra bez použitia matice a ?
  (Návod: stačí si pamätať len predposledný riadok.)

Ukážka jednoduchých rekurzívnych podprogramov

Rekurzívne funkcie a procedúry vo svojom tele volajú samých seba s inými hodnotami. Nutnou podmienkou ukončenia volaní je stanovenie hraničných podmienok.

••• Príklad 10.: Vytvorte funkciu na výpočet faktoriálu prirodzeného čísla rekurzívnou metódou.

Riešenie: Algoritmus je založený na vzťahu n! = n.(n-1)!. Úloha sa rozloží na menšie úlohy a tie sa vyriešia rovnakým spôsobom. Rozklad zastaví obmedzujúca hraničná podmienka 0! = 1.

```
function FAKTORIAL_2 ( k: integer): longint;
begin
     if  k=0 then FAKTORIAL_2 := 1
             else FAKTORIAL_2 :=k* FAKTORIAL_2 ( k-1)
end;
```

••• Príklad 11.: Vytvorte funkciu na výpočet NSD dvoch prirodzených čísel rekurzívnou metódou.

Riešenie: Aj tu sa využíva pre výpočet NSD Euklidov algoritmus.

```
procedure NSD_2( a,b: integer, var nsd: integer);
begin
   if  a=b then nsd:= a
   if   a>b then NSD_2(a-b,b,nsd)
   if   a< b then NSD_2(b-a, a,nsd )
end;
```

••• Príklad 12.: Vytvorte rekurzívnu funkciu na výpočet kombinačného čísla.

Riešenie: (pozri riešenie príkladu 9.)

```
function KOMB_CISLO3 (p,r : integer): integer;
begin
      if (p = r) or(r = 0)  then KOMB_CISLO3 : = 1
                            else
KOMB_CISLO3 : = KOMB_CISLO3 (p – 1, r – 1) + KOMB_CISLO3 (p – 1, r)
end;
```

ÚLOHY:

a) Zostavte funkciu , ktorá určí najväčšieho deliteľa celého čísla okrem seba samého.

b) Zostavte procedúru na výpočet uhlov trojuholníka podľa kosínusovej vety, ak sú dané veľkosti jeho strán.

c) Zostavte funkciu , ktorá určí najväčšieho spoločného deliteľa troch čísel.

d) Zostavte funkciu na výpočet najmenšieho spoločného násobku dvoch čísel.

e) Zostavte procedúru na krátenie zlomkov, v riešení využite funkciu NSD.

f) Zostavte funkciu na výpočet súčtu cifier na párnych pozíciách od najnižších rádov.

g) Zostavte funkciu na výpočet početnosti danej cifry v danom celom čísle.

h) Vypočítajte hodnotu výrazu zloženého z *n* druhých odmocnín:

$$\sqrt{2+\sqrt{2+\ldots\ldots\ldots+\sqrt{2}}}$$

5. BOOLOVSKÉ FUNKCIE

Jednoduché funkcie typu *boolean* sú výborným prostriedkom pre zápis kódu, v ktorom treba rozhodnúť, či daná premenná spĺňa istú požiadavku, alebo ju nespĺňa. V prípade dlhých, ťažko čitateľných boolovských výrazov pri príkazoch *IF, CASE, WHILE a REPEAT* je ich použitie žiadané. Nasledujúce príklady riešia obidva prípady:

• **Priklad 1.:** Rozhodnite, či trojuholník so stranami a, b, c existuje ?
(Či platí trojuholníková nerovnosť.)

Riešenie:
```
function EXISTUJE : boolean;
begin
  if (a< b+c) and (b<a+c) and (c<a+b) then  EXISTUJE:= true
                                      else  EXISTUJE:= false
end;
```

Otázky:

* Ako by ste podmienku zapísali pomocou operátorov *or*?
* Dá sa zapísať telo funkcie bez použitia príkazu *if* ?

• **Priklad 2.:** Vytvorte funkciu, ktorá rozhodne o celom čísle , či je párne ,alebo nepárne.

Riešenie:
```
function PARNE (c : integer) : boolean;
 begin
      if  c mod 2= 0  then  PARNE:= true
                      else   PARNE:= false
 end;
```

Otázky:

* Ako inak by ste mohli zapísať podmienku?
* Dá sa zapísať telo funkcie bez použitia príkazu *if* ?

• **Priklad 3.:** Rozhodnite, či úsečka AB o súradniciach A[x1,y1], B[x2,y2] leží celá vo vnútri kružnice k[S,r], ak stred kružnice má súradnice S[x,y].

Riešenie:

```
function  USECKA_IN : boolean;
  function PODMIENKA: boolean;
  begin
      PODMIENKA:=(sqr(x1-x)+sqr(y1-y)<sqr(r)) and
                 (sqr(x2-x)+sqr(y2-y)<sqr(r))
  end;
begin
   if  PODMIENKA  then  USECKA_IN := true
                else  USECKA_IN := false;
end ;
```

Otázky:

* Ako by ste podmienku zapísali pomocou operátora ***or?***

•• **Priklad 4.:** Vytvorte funkciu, ktorá rozhodne o celom čísle , či je dokonalé.
(Číslo je dokonalé , ak sa rovná súčtu svojich deliteľov okrem seba samého.)

Riešenie:

```
function DOKONALE (cislo : integer) : boolean;
  function  SUCET;
     var s, delitel :integer;
   begin  s:=0;
     for delitel := 1 to  cislo div 2 do
       if  cislo mod delitel =0 then s:=s+delitel ;
     SUCET:=s
   end;
begin
  if  SUCET = cislo   then  DOKONALE:= true
                      else   DOKONALE:= false
end;
```

Otázky:

* Môžeme v tele funkcie SUCET použiť príkaz: SUCET : = SUCET + deliteľ ?

••Priklad 5.: Vytvorte funkciu, ktorá rozhodne o celom čísle , či je prvočíslo. (Prvočíslo má iba triviálnych deliteľov t.j. jednotku a seba samého. Pri riešení stačí testovať delitele nanajvýš rovné odmocnine z daného čísla.)

Riešenie: Pri riešení stačí testovať delitele menšie nanajvýš rovné odmocnine z daného čísla podľa platnej matematickej vety: Ak p nie je prvočíslo potom existuje deliteľ čísla p, menší nanajvýš rovný √p.

```
function PRVOCISLO (cislo : integer) : boolean;
    var  delitel :integer;
begin
    PRVOCISLO:= false ;
    for delitel := 2 to  trunc(sqrt(cislo)) do
          if  cislo mod delitel = 0 then  exit ;
    PRVOCISLO:= true
end;
```

Otázky:

* Ako by ste zapísali telo funkcie bez použitia volania *exit ?*

•• Priklad 6.: Zostavte funkciu , ktorá zistí o danom čísle, či je Amstrongovo. (Pre Amstrongove čísla platí : súčet tretích mocnín cifier daného čísla sa rovná sebe samému.)

Riešenie: Lokálna funkcia SUCET_MOCNIN oddeľuje cifry vstupného čísla a vytvorí súčet ich tretích mocnín.

```
function AMSTRONG (cislo : integer) : boolean;
      function SUCET_MOCNIN(c:integer): integer;
          var sucet, cifra :integer;
      begin
           sucet:=0 ;
           while c<>0 do
                begin
                     cifra:= c mod 10;
                      sucet  := sucet + cifra*cifra*cifra ;
                     c:= c div 10
                 end  ;
           SUCET_MOCNIN:= sucet
      end;
 begin
          if  SUCET_MOCNIN(cislo) = cislo  then AMSTRONG := true
```

```
                                    else AMSTRONG := false
        end;
```

Otázky:

* V ktorých riadkoch by sa líšilo riešenie, ak by SUCET_MOCNIN bola procedúrou ?
* Ako by ste zapísali telo funkcie bez použitia príkazu *if* ?

Príklad využitia boolovských funkcií v programe

•• **Priklad 7.:** Vypíšte všetky prvočísla (dokonalé čísla, amstrongove čísla,...) menšie ako 1000.

Riešenie :

```
program UKAZKA;
    var i :integer;
begin { hlavny program }
    for i:= 2 to 1000 do
        if { PRVOCISLO(i)
             DOKONALE(i)
             AMSTRONG(i) } then write ( i: 6)

end.
```

ÚLOHY:

a) Zostavte funkciu , ktorá zistí, či trojuholník so stranami a,b,c je pravouhlý.

b) Zostavte funkciu , ktorá zistí, či cifra K sa nachádza v zápise čísla N.

c) Zostavte funkciu , ktorá zistí, či dané číslo sa rovná druhej mocnine svojho ciferného súčtu.

d) Zostavte funkciu , ktorá zistí o dvoch daných číslach či sú spriatelené.
(Pre spriatelené čísla platí : súčet všetkých kladných deliteľov jedného z nich, okrem seba samého, sa rovná druhému číslu a naopak.)

e) Zostavte program pre výpis všetkých prvočísel, dokonalých čísel, resp. Amstrongových čísel zo zadaného intervalu.

6. *PREVODY POZIČNÝCH SÚSTAV*

Dôležitými pojmami v predmete informatika sú pojmy kód, spôsoby kódovania, binárna, oktálova, hexadecimálna sústava... Algoritmy pre prevody čísel medzi pozičnými sústavami sú ľahko naprogramovateľné a sú súčasťou iných algoritmov . Nasledujúce príklady demonštrujú prevody medzi dekadickou a binárnou sústavou, a ich využitie v iných úlohách.

●● **Príklad 1.:** Vytvorte funkciu, ktorá prevedie prirodzené číslo z desiatkovej do dvojkovej pozičnej sústavy.

Riešenie : Algoritmus postupne oddeľuje cifry z desiatkového čísla a buduje dvojkové číslo.

```
function DVOJKOVE_CISLO( desiatkove:word ): string;
  var  d,z: string;
begin
        d :='';
     while desiatkove>0 do
        begin
          Str( desiatkove mod 2, z);
          d:= d + z;
          desiatkove := desiatkove  div 2
        end;
     DVOJKOVE_CISLO := d
end;
```

Otázky:

- Aký typ funkcie je vhodné použiť?
- Aké najväčšie číslo by sme mohli previesť do dvojkovej sústavy, ak by funkcia bola typu integer, resp. longint?
- Môžeme v tele funkcie použiť príkaz :
  DVOJKOVE_CISLO :=DVOJKOVE_CISLO + z ?
- Aký je rozdiel v zápise príkazov:
  d:= d + z a d:= z + d ?
- V ktorom príkaze sa líši prevod z desiatkovej sústavy do oktálovej (trojkovej, hexadecimálnej,...) ?

●● **Príklad 2.:** Vytvorte funkciu, ktorá prevedie prirodzené číslo z dvojkovej pozičnej sústavy do desiatkovej .

Riešenie I. : Riešenie s využíva pozičný zápis čisla s daným základom.
(Premennú *chyba* vyžaduje syntax procedúry *val*.)

```
function DESIATKOVE_CISLO( dvojkove: string): longint;
   var i, d, nasobok, pom, chyba : integer;
begin
   nasobok:=1; d:=0;
   for i:=length(dvojkove) downto 1 do
      begin
            val( dvojkove[i], pom, chyba);
         d:= d + pom * nasobok;
         nasobok:= nasobok *2
      end
   DESIATKOVE_CISLO:= d
end;
```

Riešenie II. : Riešenie využíva zápis polynómu pomocou Hornerovej schémy:
$a_n x^n + a_{n-1}x^{n-1} + \ ... + a_2x^2 + a_1x + a_0 = a_0 + x (a_1 + x (a_2 + x (....x (a_{n-1}+x (a_n)))))$

```
function DESIATKOVE_CISLO2( dvojkove: string): longint;
   var i, d, pom, chyba : integer;
begin
   d:=0;
   for i:=1  to length(dvojkove)  do
        begin
           Val( dvojkove[i], pom, chyba);
            d:= d * 2 + pom;
        end;
   DESIATKOVE_CISLO2:= d
end;
```

Otázky:

* Môžeme v tele funkcie použiť príkaz :
  DESIATKOVE_CISLO :=DESIATKOVE_CISLO*2 + pom ?
* Koľko miestne môže byť dvojkové číslo, zapísané v tvare reťazca, ktoré dokáže táto funkcia ešte previesť do desiatkovej sústavy ?
* V ktorom príkaze sa bude líšiť prevod z oktálovej, (trojkovej, hexadecimálnej,...) do desiatkovej sústavy ?
* Porovnajte časovú zložitosť oboch riešení !

Príklady využitia funkcií na prevody v programe

•• Príklad 3.: Vytvorte funkciu, ktorá zistí, či dvojkový zápis daného celého čísla je symetrický.

Riešenie : Prevedie desiatkové číslo na dvojkové a porovnáva zhodnosť cifier na odpovedajúcich symetrických miestach.

```
function SYMETRICKY( cislo:integer):boolean;
  var  bin: string;
       dlzka_bin: byte;
begin
   SYMETRICKY:=false;
   bin:=DVOJKOVE_CISLO(cislo);
   dlzka_bin:= length(bin);
   for i:=1 to dlzka_bin div 2 do
      if bin[i] <> bin[dlzka_bin-i+1] then exit;
   SYMETRICKY:=true;
end;
```

Otázky:

* Ako by ste zapísali telo funkcie bez použitia volania *exit* ?

•• Príklad 4.: Vytvorte funkciu, ktorá zistí, či dané dvojkové číslo je prvočíslo.

Riešenie : Využíva funkciu DESIATKOVE_CISLO na prevod z dvojkovej sústavy a boolovskú funkciu PRVOCISLO, ktorá je volaná hodnotou desiatkového čísla .

```
function DVOJKOVE_PRVOCISLO( cislo:string):boolean;
 begin
   if  PRVOCISLO( DESIATKOVE_CISLO( cislo ) )
         then  DVOJKOVE_PRVOCISLO:=true
         else  DVOJKOVE_PRVOCISLO:=false
end;
```

Otázky:

* Ako by ste zapísali telo funkcie bez použitia príkazu *if* ?

 **ÚLOHY:**

a) Zostavte funkciu , ktorá zistí pre dané dvojkové číslo najväčšieho deliteľa okrem seba samého.

b) Zostavte funkciu , ktorá zistí, či dvojkový zápis daného čísla obsahuje párny počet jednotiek.

c) Zostavte funkciu , ktorá zistí, či dané dvojkové číslo je deliteľné číslom K.

d) Zostavte funkciu , ktorá zistí súčet, rozdiel alebo súčin dvoch dvojkových čísel podľa zadanej operácie.

e) Zostavte funkciu , ktorá zistí NSD dvoch dvojkových čísel v dvojkovom tvare.

f) Vytvorte funkcie na prevod čísel z desiatkovej sústavy do oktálovej resp. hexadecimálnej

g) Vytvorte funkcie na prevod čísel z oktálovej resp. hexadecimálnej sústavy do desiatkovej .

h) Vytvorte funkciu RIMAN na prevod arabského čísla na zápis pomocou rímskych znakov.

i) Vytvorte funkciu ARAB na prevod rímskeho čísla na arabské.

7. VEKTORY A MATICE

Súčasťou všetkých programov, ktoré pracujú s jednorozmernými aj viacrozmernými poľami je ich načítanie a výpis. Práve opakovanie týchto procedúr dáva možnosť vytvorenia knižnice, kde sa implementujú procedúry najviac využívané v programoch. Ich univerzálnosť zabezpečíme pomocou parametrov, ale tie sú štrukturované. Preto v knižnici definujeme typy index, postupnosť a matica nasledujúcim spôsobom:

```
type index = 0..h; (h je číselná horná hranica pre veľkosť polí)
     postupnost = array [index] of integer;
     matica = array [index, index] of integer;
```

Nasledujúce najčastejšie volané procedúry a funkcie tvoria obsah knižnice POLIA, pre prácu s vektormi a maticami, ktorá je uvedená v prílohách:

•Priklad 1.: Vytvorte procedúry resp. funkcie, ktoré sa najčastejšie používajú pri práci s vektormi a maticami.

```
procedure CITAJ_POSTUPNOST (var a : postupnost; n : index);;
    var i : index;
begin
   writeln ('Zadaj postupnost celych cisel');
   for i:=1 to n do
       begin
           write ('prvok(',i,')= ');
           readln (a[i])
       end
end;

procedure CITAJ_MATICU(var a : matica; m,n : index);
    var i,j : index;
begin
   writeln ('Zadaj prvky celociselnej matice');
   for i:=1 to m do
      for j:=1 to n do
          begin
              write ('prvok[',i,',',j,']= ');
              readln (a[i,j])
          end
end;
```

```
procedure PIS_POSTUPNOST(a : postupnost; n : index);
    var i : index;
begin
   for i:=1 to n do write (a[i], ' ');
   writeln
end;

procedure PIS_MATICU(a : matica; m,n : index);
    var i,j : index;
begin
    for i:=1 to m do
       begin
           for j:=1 to n do write (a[i,j]:4);
          writeln
       end
end;

function SUCET_POSTUPNOSTI( a: postupnost, n: index): integer;
     var i: index; s: integer;
begin
     for i:=1 to n do s:= s + a[i];
     SUCET_POSTUPNOSTI :=s
end;

  procedure NULOVANIE_POSTUPNOSTI( var a: postupnost, n:index );
     var i: index;
 begin
     for i:=1 to n do a[i]:=0;
 end;
```

Ďalšie príklady demonštrujú využitie štrukturálneho zápisu kódu programu pri riešení úloh s vektormi a maticami:

Jednorozmerné polia

•Priklad 2.: Žiaci dostali za úlohu nájsť vzdialenosť medzi mestami A, B. Z mapy si vypísali dĺžky úsekov medzi mestečkami a dedinami, ktoré mestá A, B spájajú. Pomôžte im riešiť túto úlohu.

Iná formulácia: Zostavte funkciu pre výpočet súčtu prvkov jednorozmerného poľa.

Riešenie :
```
function VZDIALENOST_MIEST( a: postupnost, n: index):integer;
  begin
     CITAJ_POSTUPNOST( a,n);
     VZDIALENOST_MIEST:=SUCET_POSTUPNOSTI(a,n}
  end;
```

••Priklad 3.: Na streleckých pretekoch štartovali dve družstvá A, B. Zvíťazilo družstvo s najväčším počtom bodov. Zostavte program, ktorý vypíše body jednotlivých športovcov za každé družstvo, priemerný počet bodov a na záver označí víťaza.

Iná formulácia: Zostavte program na výpis prvkov dvoch jednorozmerných polí., výpis priemerných hodnôt ako aj meno poľa s najmenším priemerom.

Riešenie :
```
program PRETEKY;
   var a, b :postupnost;
       n, m: index;
       priemer_a, priemer_b: real;
  begin
     write(´Zadaj pocty clenov druzstiev A, B:´);
     readln(n,m);
     CITAJ_POSTUPNOST( a,n);
     CITAJ_POSTUPNOST( b,m);
     priemer_a:= SUCET_POSTUPNOSTI(a,n) / n;
     priemer_b:= SUCET_POSTUPNOSTI(b,m) / m;
     PIS_POSTUPNOST(a,n); writeln (priemer_a);
     PIS_POSTUPNOST(b,m); writeln (priemer_b);
     if  priemer_a > priemer_b then writeln('Vitaz je A druzstvo')
                              else writeln('Vitaz je B druzstvo')
  end.
```

Otázky:

* Ako by ste upravili program, aby výpisy boli prehľadné?
* Čo všetko by sme museli zmeniť, ak by program vyhodnocoval plavecké prteteky? (časy pretekárov sú desatinné čísla)

••Priklad 4.: Pri platení nákupu nám pokladník vráti istú sumu peňazí najmenším počtom bankoviek. Simulujte jeho myslenie prostredníctvom procedúry.

Iná formulácia: Zostavte procedúru, ktorá určí početnosti jednotlivých bankoviek uložených v poli BANKOVKA (od 5000 Sk do 1 Sk) pri vyplatení danej sumy.

Riešenie :
```
procedure VYPLAT_SUMU( suma:integer);
    var  i: index;
begin
    for i:= 1 to 11 do
       begin
         počet_ban[i]:= suma div bankovka[i];
         suma := suma mod bankovka[i]
       end
end;
```

••Priklad 5.: Mzdová účtovníčka vypláca každému z n zamestnancov firmy každý mesiac istú sumu peňazí. Snaží sa vybrať z banky minimálny počet bankoviek, ale výpočet jej zaberá veľa času. Zostavte program na urýchlenie práce mzdovej účtovníčky.

Iná formulácia: Zostavte program na výpis početností jednotlivých druhov bankoviek pre vyplatenie n súm s využitím procedúry VYPLAT_SUMU.

Riešenie : Predošlá procedúra je súčasťou tohto riešenia, ale až tu je deklarované pole *bankovka*.

```
program MZDY;
   const  bankovka: array[1..11] of integer = (5000, 1000, 500, 200, 100, 50, 20,
                                               10, 5, 2, 1);
    var pocetnost, suma: postupnost;
        n: index;
begin
   readln(n);
     NULOVANIE_POSTUPNOSTI (pocetnost, 11)
```

```
    CITAJ_POSTUPNOST(suma, n);
    for i := 1 to n do
        begin
            VYPLAT_SUMU(suma[i]);
            for j:=1 to 11 do
              pocetnost[j]:= pocetnost[j]+ počet_ban[j];
       end;
    PIS_POSTUNOST( pocetnost, 11);
end.
```

Otázky:

- Ako by ste upravili program, aby čítanie a výpisy boli prehľadné?
- Program sa dá zjednodušiť úpravou procedúry VYPLAT_SUMU, ktorá používa cyklus premennej „j“, pričom ten istý cyklus používa aj telo programu, pokúste sa o to.

••Priklad 6.: Do školského basketbalového tímu prijímajú nových hráčov. Zostavte funkciu na vyhľadanie najvyššieho žiaka 1.A triedy .

Iná formulácia: Zostavte funkciu , ktorá nájde maximum v zadanej postupnosti celých čísel.

Riešenie :

```
function MAX( p: postupnost, n:integer): integer;
    var i : integer;
begin
    MAX:= -maxint;
    for i:= 1 to n do
        if p[i] > MAX then  MAX:= p[i]
end;
```

••Priklad 7.: Najvyšší žiak 1.A triedy ochorel. Do basketbalového tímu treba nájsť náhradníka. Zostavte funkciu na vyhľadanie druhého najvyššieho žiaka triedy.

Iná formulácia: Zostavte funkciu , ktorá nájde druhé maximum v zadanej postupnosti celých čísel.

Riešenie :

```
function DRUHE_MAX( p: postupnost, n:integer): integer;
    var i, max1: integer;
```

```
begin
      max1:= -maxint;
      max2:= -maxint;
      for i:= 1 to n do
          if p[i] > max1 then
                              begin
                                 max2:= max1;
                                 max1:= p[i]
                              end
                           else  if p[i] > max2  then max2:= p[i]
     DRUHE_MAX := max2
end;
```

Otázky:

* Ako by ste upravili funkciu DRUHE_MAX, ak by ste chceli využiť predošlú funkciu MAX?

••• Priklad 8.: Vytvorte funkciu, ktorá rozhodne o celom čísle , či je palindrom.
(Číslo je palindrom, ak je jeho zápis symetrický.)

Riešenie: Procedúra VYTVOR_POLE najprv rozdelí vstupné číslo na cifry a uloží ich do poľa *p* a v poli sa skúma jeho symetria.

```
function PALINDROM (cislo : integer) : boolean;
   var i,n:integer; p:pole;
   procedure VYTVOR_POLE ;
   begin  n:=0;
        repet  inc(n);
               p[n]:= cislo mod 10;
               cislo:=cislo div 10
        until cislo=0
   end;
begin
    PALINDROM := false ;
    VYTVOR_POLE;
    for i:= 1 to  n div 2  do
        if  p[i]<>p[n-i+1] then exit ;
    PALINDROM:= true
end;
```

Otázky:

* Ako by ste zapísali telo funkcie bez použitia volania *exit*

* Dá sa úloha vyriešiť bez použitia poľa ?

••• Príklad 9.: Vytvorte funkciu na výpočet faktoriálu prirodzeného čísla , ktorý svojou veľkosťou presiahne typ *longint.*

Riešenie: (V riešení je faktoriál zapísaný po cifrách v poli *p*, cifry sa násobia z konca poľa a *prenos* sa pripočítava k nasledujúcej cifre, pričom si pamätáme posledný počet obsadených miest v premennej *obsadene,* teda počet cifier faktoriálu)

```
procedure VELKY_FAKTORIAL( n: integer; var p: postupnost;  var obsadene:integer
)
     var  k, prenos : integer;
    procedure SPRACUJ_K;
      var i, pom: integer;
          procedure SPRACUJ_POSLEDNY_PRENOS ;
              begin
                 while  prenos>0  do
                     begin
                         inc(obsadene) ;
                         p[obsadene]:= prenos mod 10 ;
                         prenos := prenos div 10
                     end
              end; {SPRACUJ_POSLEDNY_ PRENOS}
     begin
        i:=1;
        while i <= obsadene do
           begin
              pom:= p[i]*k+ prenos ;
              p[i]:= pom mod 10;
              prenos := pom div 10;
              inc(i)
           end;
       SPRACUJ_POSLEDNY_ PRENOS
     end;{SPRACUJ_K}
begin
       p[1]:=1 ; obsadene:=1;
       for k:=1  to  n  do
          begin
              prenos:=0;
              SPRACUJ_K
          end
```

```
end;{ VELKY_FAKTORIAL}
```

Dvojrozmerné polia

• **Priklad 10.:** Vytvorte procedúru, ktorá vymení K-ty s L-tym riadkom vstupnej matice.

Riešenie:

```
procedure VYMENA_RIADKOV_MATICE (var a: matica; m,n,k,l: integer);
   var  j :integer;
begin
  for j:= 1 to n do VYMENA_2( a[K,j], a[L,j] )
end;
```

Otázky:

* Mohli by sme použiť na výmenu prvkov matice procedúru VYMENA ?
* Upravte procedúru tak, aby vymieňala dva ľubovoľné riadky!
* Čím by sa líšila procedúra VYMENA _STLPCOV_MATICE ?

•• **Priklad 11.:**Vytvorte program, ktorý vypíše vektor súčtov prvkov na diagonálach rovnobežných s hlavnou diagonálou matice. (Matica typu *mxn* má *m+n-1* diagonál rovnobežných s hlavnou diagonálou.)

Riešenie :

```
program SUCET_DIAGONAL;
     var a: matica;
         p: postupnost;
         m, n, k: index;
    procedure VYTVOR_SUCTY_DIAGONAL;
        var  i, j, posun: index;
    begin
      posun:= abs(m-n) div 2 +1;
       for i:= 1  to m  do
          for  j:= 1 to n  do  p[i-j+posun]:= p[i-j+posun] + a[i,j]
    end;
 begin
   readln( m, n );
   CITAJ_MATICU(a, m, n);
   NULOVANIE_POLA( p, n+m-1);
   VYTVOR_SUCTY_DIAGONAL;
   PIS_POSTUPNOST( p, n+m-1)
 end.
```

Otázky:

* zjednodušte telo programu tak, aby súčty diagonál matice boli po výpočte hneď vypisované!

••• Priklad 12. Vytvorte procedúru, ktorá vykreslí nasledujúci obrazec, zložený z *k* útvarov o *m* riadkov a *n* hviezdičiek v riadku.

Napr. pre *k*=2, *m*=5, *n*=6 obrazec vyzerá takto:

```
****** ******
*    * *    *
*    * *    *
*    * *    *
****** ******
```

Riešenie:

```
procedure OBRAZEC( k, m, n:integer) ;
  var i : integer;
 procedure KRESLI_RIADOK(x : integer);
    var l,h : integer;
  begin
        for h:= 1 to k do
          begin
            write('*');
            for l:= 2 to n-1 do
              begin
                if (x=1) or (x=m) then write('*');
                if (x>=2) and (x<=m-1) then write(' ')
              end;
            write('*');
            write('  ')
          end;
        writeln
    end;
  procedure VYNECHAJ(k: integer);
        var l: integer;
  begin
        for l:= 1 to k do  write(' ')
  end;
begin
  for i:= 1 to m  do
    begin
      VYNECHAJ( m-i );
```

```
                KRESLI_RIADOK( i )
              end;
        end ;
```

Otázky:

* zjednodušte telo procedúry tak, aby vykreslené rovnobežníky boli vyplnené!

••• Priklad 13.: Vytvorte procedúru, ktorá danú maticu nxn otočí o 90 stupňov.

Riešenie I : Riešenie s použitím matice *b*, do ktorej ukladáme otočené prvky.

```
procedure OTOC;
        var i,j :integer ;
    begin
        for i:= 1 to n do
            for j:= 1 to n do  b[ j, n-i+1] := a[ i, j]
    end;
```

Ukážka programu, ktorý využíva procedúru OTOC:

```
program OTOCENIE_MATICE;
    var a,b: matica;
        n: index;
    procedure OTOC;
        var i,j :integer ;
    begin
        for i:= 1 to n do
            for j:= 1 to n do  b[ j, n-i+1] := a[ i, j]
    end;
begin
    readln( n );
    CITAJ_MATICU(a, n, n);
    OTOC;
    PIS_MATICU(b, n, n);
end.
```

Riešenie II : Procedura OTOC_KRUZNICU nepoužíva pomocnú maticu *b*, prvky matice *a* otáčame po kružniciach a vymieňame odpovedajúce prvky.

```
program OTOCENIE_MATICE_2;
    var a: matica; n: index;
        i:integer;

    procedure OTOC_KRUZNICU;
```

```
        var j, pom :integer ;
    begin
        for j:= i to n-i  do
            begin
                pom:= a[i, j] ;
                a[ i, j]:= a[n-j+1, i] ;
                a[n-j+1, i]:= a[n-i+1,n-j+1] ;
                a[n-i+1,n-j+1]:= a[j,n-i+1] ;
                a[j,n-i+1]:= pom
            end
    end ;
begin
    readln( n );
    CITAJ_MATICU(a, n, n);
    for i:= 1  to  n div 2  do OTOC_KRUZNICU;
    PIS_MATICU(a, n, n);
end.
```

ULOHY:

a) Je daná postupnost celých čísel. Zistite, či sa niektoré z nich opakuje.

b) Na vstupe sú dve veľmi dlhé čísla. Vypíšte ich súčet, resp. rozdiel.

c) Je daná postupnost celých čísel. Nájdite najdlhšiu neklesajúcu postupnosť obsiahnutú v danej postupnosti.

d) Akým najmenším počtom mincí je možné vyplatiť sumu S korún, ak máme mince o hodnotách1, 2, 3, 5, 10, 15, 20 a 50 korún.

e) Zimnej olympiády sa zúčastnilo „n“ krajín. Každá získala určitý počet zlatých, strieborných a brondzových medailí . Vypíšte 10 najúspešnejších krajín s najväčším počtom bodov, ak za medaily sú pridelené postupne 3, 2 a 1 bod.

f) Zostavte boolovské funkcie na zistenie vlastností vstupnej matice t.j. či je symetrická, diagonálna, trojuholníková.

g) Vytvorte procedúru, ktorá vykreslí nasledujúci obrazec, zložený z *k* útvarov o *m* riadkov napr. pre *k*=3, *m*=5 obrazec vyzerá takto:

```
*       *       *
```

```
 * *        * *        * *
 * * *      * * *      * * *
 * * * *    * * * *    * * * *
 * * * * *  * * * * *  * * * * *
```

h) Vytvorte procedúru, ktorá pre dané číslo n vypíše špirálu tvorenú číslami od 1 do n^2 , napr. pre n=4 špirála vyzerá takto:

```
 1   2   3   4
12  13  14   5
11  16  15   6
10   9   8   7
```

8. REŤAZCE

Často vynechávanou témou v predmete programovanie sú úlohy pre prácu s textami, znakmi. Práve formou takýchto príkladov sa žiaci môžu oboznámiť s pojmami kódovanie, šifrovanie, kryptografia, spoznajú rôzne druhy kódov (ASCII kód....). Táto téma napomáha hĺbšie nazrieť do spôsobu práce počítača. Úlohy sú pre žiakov príťažlivé a pútavé.

•Priklad 1.: Vstupné heslo do hry pozostáva z číslic. Súčasťou obslužného programu je funkcia, ktorá zo zadávaných znakov akceptuje len číslice. Napíšte ju.

Iná formulácia: Vytvorte funkciu , ktorá zistí o danom znaku, či je číslica.

Riešenie :
```
function JE_CISLICA(znak:char):boolean;
  begin
    if (ord (znak) > 48)and (ord(znak)< 59 ) then JE_CISLICA:=true
                                             else JE_CISLICA:=false
  end;
```

••Príklad 2.: Kód ku odstaveniu zabezpečovacieho systému budov má byť reťazec bez čísel. Zistite, či zadaný kód vyhovuje podmienke.

Iná formulácia: Zostavte program, ktorý určí, či daný reťazec obsahuje iba písmená.

Poznámka: V riešení využijeme predchádzajúcu funkciu.

Riešenie: Riešenie využíva predošlú funkciu JE_CISLICA, zapísanú iným spôsobom.

```
program KOD;
  var heslo: string;
 function JE_CISLICA(znak:char):boolean;
    begin
      JE_CISLICA:= (ord (znak) > 48)and (ord(znak)< 59 )
    end;
function OBSAHUJE_PISMENA :boolean;
  var i: integer;
        begin
       OBSAHUJE_PISMENA := false;
       for i:= 1 to lenght(heslo) do
           if JE_CISLICA(heslo[i]) then exit;
      OBSAHUJE_PISMENA :=true;
```

```
        end;
    begin        {hlavny program}
        readln ( heslo);
        writeln(' heslo obsahuje iba pismena ',OBSAHUJE_PISMENA)
    end.
```

Otázky:

- Funkciu OBSAHUJE_PISMENA prepíšte bez použitia príkazu *exit!*

••Priklad 3.: V krajine Hviezdičkovo sa obyvatelia rozhodli, že slová nebudú oddeľovať medzerou, ale hviezdičkou. Pomôžte im. Zostavte procedúru, ktorá spomínanú náhradu vykoná v ľubovoľnej vete.

Iná formulácia: Vytvorte procedúru , ktorá v zadanej vete zamení každú medzeru za hviezdičku

Riešenie :

```
procedure ZAMENA(var veta: string);
    var i: integer;
begin
   for i:= 1  to  length(veta)  do
    if  veta[i]=' ' then
      begin
        delete( veta,i,1);
        insert('*',veta,i)
      end
end;
```

Otázky:

- Aká dlhá môže byť vstupná veta ?
- Ako inak by sa dala zapísať podmienka príkazu *if*?

••Priklad 4.: Vytvorte funkciu , ktorá číslo typu *longint* prevedie na reťazec cifier.

Riešenie:

```
function NUM_STRING(cislo:longint): string;
 var s:string;
     cifra:integer;
begin
        s:='';
      repeat
        cifra:= cislo mod 10;
        s:= char( cifra + ord ('0')) + s;
        cislo:= cislo div 10;
      until cislo=0;
```

```
            NUM_STRING:= s
        end;
```

Otázky:

- Sú zhodné nasledujúce výrazy:
  s:= char(cifra + ord (‘0’)) + s
  s:= s + char(cifra + ord (‘0’))
- V čom by sa líšila funkcia STRING_NUM od NUM_STRING?

••Priklad 5.: Tlačiarenský škriatok zase šarapatil, tentokrát ešte v počítačovom formáte novín. V nadpisoch, ktoré majú byť písané veľkými písmenami, zamenil niektoré písmená za malé. Vráťte nadpisy do pôvodného stavu.

Iná formulácia: Zostavte procedúru, ktorá v zadanej vete zamení všetky malé písmená za veľké.

Riešenie:

```
procedure MALE_VELKE(var veta: string);
    var i: integer;
  function PODMIENKA: boolean;
  begin
    PODMIENKA:=( ord(veta[i]) >ord(‘a’) ) and ( ord(veta[i])< ord(‘z’) )
  end;
begin
  for i:=1 to length(veta) do
    if  PODMIENKA then  veta[i]:=chr(ord( veta[i]) - 33 )
end;
```

Otázky:

- Je nutné kľúčové slovo *var* v parametroch procedúry?
- Akým iným spôsobom by ste zapísali telo funkcie PODMIENKA?
- V ktorých príkazoch by sa líšila procedúra VELKE_MALE?

•••Priklad 6. Betka dostala za úlohu pripraviť triednu nástenku, ale nevie, či jej stačia písmená. Uľahčite Betke prácu a spočítajte, koľko ktorých písmen na nástenku potrebuje. (Na nástenku sa používajú iba veľké písmená .)

Iná formulácia: Zostavte program, ktorý vypíše početnosti veľkých písmen v zadanej vete.

Riešenie:

```
program  BETKA;
    type   pole= array[‘A’..’Z’] of integer;
```

```
  var  znak: char;
       pocet: pole;
       veta: string;
procedure NULOVANIE_POLA( var p:pole);
    var i:char;
    begin
        for i:='A' to 'Z' do p[i]:=0
    end;
 procedure ZISTI_POCETNOST;
          var i:integer ;
   function VELKE_PISMENO: boolean;
      begin
       VELKE_PISMENO:= ( veta[i] >= 'A' ) and ( veta[i]) <= 'Z' )
      end ;
       begin
         for i:= 1 to length(veta) do
            if  VELKE_PISMENO  then   inc( pocet[veta[i]] )
      end;
begin {hlavny program}
       readln (veta);
       NULOVANIE_POLA(POCET);
       ZISTI_POCETNOST;
    for  znak:='A'  to  'Z'  do  write( znak,'.....', pocet[znak] )
end.
```

Otázky:

- Mohli by sme deklarovať procedúru ZISTI_POCETNOST ako funkciu?
- Upravte program tak, aby vypisoval iba nenulové početnosti
- Zjednodušte program tak, aby vypisoval pocetnosti všetkých znakov vo vstupnej vete

ÚLOHY:

a) Na vstupe je postupnosť čísel a operandov + a - . Zostavte funkciu , ktorá vyhodnotí vstupný výraz.

b) Na vstupe je dané číslo K a správa, ktorú treba zakódovať cyklickým posunom abecedy o K znakov . Napr.: Písmeno A sa zakóduje pre K=5 ako písmeno F, písmeno Z sa zakóduje ako písmeno E atď.

c) Simulujte tzv. „živé noviny“. Na vstupe je reklamný slogan, ktorý sa cyklicky posúva smerom do prava .

d) Zostavte funkciu, ktorá vráti počet slov vstupnej vety.

e) Pri volení užívateľského hesla pre prihlásenie na počítačovú sieť sa vstupné heslo druhý krát zadáva pre potvrdenie prvého. Napíšte program , ktorý skontroluje zadané heslo.

LITERATÚRA

J. Studenovský: Prednášky z programovania, PF UPJŠ Košice, 1996

M. Kvoch: Programování v TURBO PASCALU 7.0, KOPP České Budejovice 1996

D. Tőpflerová, P. Tőpfler: Sbírka úloh z programování, GRADA Praha edícia EDUCA ´99 1992

M. Kvoch, J. Jančík: Sbírka úloh z jazyka PASCAL, KOPP České Budejovice 1995

I. Kopeček, J. Kučera: Programátorské poklesky, Mladá fronta Praha 1989

PRÍLOHY

```
Unit CISLA;

INTERFACE

 uses POLIA;
 var a,b,pom,i,j,k,n,m:integer;
 procedure VYMENA_1;
 procedure VYMENA_2( var a,b: integer);
 function MAX(a,b: real): real;
 function MOCNINA ( n: integer ;  x : real ) : real;
 function FAKTORIAL_1 ( c : integer) : longint ;
 function CIF_SUCET( c : integer): integer ;
 function NSD( a,b : integer): integer ;
 procedure NSD_2( a,b: integer; var nsd: integer);
 function KOMB_CISLO ( n, k : integer) : longint;
 function KOMB_CISLO_2 (n , k : integer) : longint;
 procedure PASCAL (n : integer) ;
 procedure QUICKPASCAL (n : integer);
 procedure VELKY_FAKTORIAL (n:word; var p:postupnost; var obsadene:index);
 function FAKTORIAL ( k: integer): real;
 function KOMB_CISLO3 (p,r : integer): integer;

IMPLEMENTATION

 procedure VYMENA_1;
        var pom: integer;
        begin
          pom := a ;
          a := b ;
          b := pom ;
        end;
 procedure VYMENA_2;
        var pom:integer;
        begin
          pom := a ;
          a := b ;
          b := pom ;
        end;
 function MAX;
        begin
         if  a > b  then MAX:= a
                    else MAX:= b
        end;
```

```
function MOCNINA;
        var p : real ;
        begin
          p:=1;
          for  i:=1  to  n  do  p:= p*x ;
          MOCNINA:= p
        end;
function FAKTORIAL_1 ;
        var  k :longint;
             i :integer ;
        begin
          k:= 1 ;
          for  i:=c downto 2 do  k:= k*i ;
          FAKTORIAL_1:=k ;
        end;
function CIF_SUCET ;
        var sucet:integer;
        begin
          sucet:=0 ;
          while c<>0 do
          begin
            sucet:=sucet+c mod 10 ;
            c:= c div 10
          end ;
          CIF_SUCET:= sucet
        end ;
function NSD ;
        begin
          while a<>b do
            begin   if  b>a then VYMENA_2(a,b) ;
              a := a - b
            end ;
          NSD := a
        end ;
procedure NSD_2;
        begin
          if a=b then nsd:= a;
          if a>b then NSD_2(a-b,b,nsd);
          if a<b then  NSD_2(b-a, a,nsd);
        end;
function KOMB_CISLO ;
```

```
        begin
       KOMB_CISLO:= FAKTORIAL_1(n) div  FAKTORIAL_1(k) div
FAKTORIAL_1(n-k)
        end;
 function KOMB_CISLO_2 ;
        var p: longint; i : integer ;
        begin
          p:= 1;
          for  i:=n  downto n-k+1 do p:=p * i ;
          KOMB_CISLO_2:=p div FAKTORIAL_1( k )
        end;
 procedure PASCAL ;
        var i,j: integer;
        begin
          writeln ('1 ' );
          writeln ( '1       1' ) ;
          for i:= 2  to  n  do
            begin
              write ( '1' );
              for j:=1 to  i - 1 do write ( KOMB_CISLO(i , j ) :8 ) ;
              writeln ( '       1' );
            end;
            writeln
        end;
 procedure QUICKPASCAL;
        var  a: matica;
        begin
          a[1,1]:=1;
          for i:=2 to n do a[1,i]:=0;
          a[2,1]:=1;  a[2,2]:=1;
          for i:=3 to n do a[2,i]:=0;
          for i:= 1 to n do
            begin
              a[i,1]:= 1;
              for j:=2 to i-1 do a[i,j]:=a[i-1,j-1] + a[i-1,j];
              a[i,i]:=1;
              for j:=i+1 to n do a[i,j]:=0;
            end;
          PIS_MATICU(a,n,n)
        end;
 procedure VELKY_FAKTORIAL ;
```

```
          var  i, prenos :integer;
          procedure SPRACUJ_K;
            begin   i:=1;
              while i <= obsadene do
                begin
                 if prenos<>0 then
                   begin pom:= p[i]*k+ prenos ;
                        prenos:=0;
                   end
                 else pom:=p[i]*k;
                 p[i]:= pom mod 10;
                 prenos:= pom div 10;
                 inc(i)
                end;
              while prenos>0  do
                        begin inc(obsadene);
                             p[obsadene]:=prenos mod 10;
                             prenos:=prenos div 10
                        end;
            end;
          begin
            p[1]:=1 ; obsadene:=1;
            for k:=1  to  n  do
              begin
                prenos:=0;
                SPRACUJ_K
              end;
           end;
    function FAKTORIAL ;
           begin
             if  k=0 then FAKTORIAL:=1
           else FAKTORIAL :=k* FAKTORIAL ( k-1)
           end;
function KOMB_CISLO3;
      begin
          if (p = r) or(r = 0)  then KOMB_CISLO3 : = 1
             else
          KOMB_CISLO3 : = KOMB_CISLO3 (p – 1, r – 1) + KOMB_CISLO3 (p –
  1, r)
        end;
END.
```

```
Unit BOOLOVSKE;
INTERFACE

 uses POLIA;
 var x1,y1,x2,y2,x,y,r:real;
 function EXISTUJE(a,b,c:real):boolean;
 function USECKA_IN:boolean;
 function PARNE (c:integer):boolean;
 function DOKONALE (cislo:integer):boolean;
 function PRVOCISLO (cislo:integer):boolean;
 function AMSTRONG (cislo : integer) : boolean;

IMPLEMENTATION

 function EXISTUJE ;
     begin
       if (a< b+c) and (b<a+c) and (c>a+b) then  EXISTUJE:= true
                                  else  EXISTUJE:= false
     end;

 function  USECKA_IN : boolean;
   function PODMIENKA: boolean;
   begin
       PODMIENKA:=(sqr(x1-x)+sqr(y1-y)<sqr(r)) and
              (sqr(x2-x)+sqr(y2-y)<sqr(r))
   end;
  begin
    if  PODMIENKA  then  USECKA_IN := true
              else  USECKA_IN := false;
  end ;

 function PARNE;
   begin
     if  c mod 2= 0  then  PARNE:= true
                 else  PARNE:= false
   end;

 function DOKONALE;
    var s,delitel :integer;
    begin
      s:=1;
      for delitel:= 2 to  cislo div 2 do
        if  cislo mod delitel=0 then s:=s+delitel ;
```

```
     if  s=cislo   then  DOKONALE:= true
                 else  DOKONALE:= false
   end;

function PRVOCISLO;
   var  delitel :integer;
   begin
     PRVOCISLO:= false ;
     for delitel:= 2 to  trunc(cislo/2) do
        if  cislo mod delitel=0 then  exit ;
     PRVOCISLO:= true
   end;

function AMSTRONG (cislo : integer) : boolean;
  function SUCET_MOCNIN(c:integer): integer;
    var sucet, cifra :integer;
  begin
    sucet:=0 ;
    while c<>0 do
      begin
        cifra:= c mod 10;
        sucet  := sucet + cifra*cifra*cifra ;
        c:= c div 10
      end  ;
    SUCET_MOCNIN:= sucet
  end;
 begin
   if  SUCET_MOCNIN( cislo ) = cislo   then  AMSTRONG:= true
                               else  AMSTRONG:= false
 end;

END.
```

```
Unit PREVODY;

INTERFACE

uses POLIA, BOOLOVSKE;
function DVOJKOVE_CISLO( desiatkove:word ): string;
function DESIATKOVE_CISLO( dvojkove: string): longint;
function DESIATKOVE_CISLO2( dvojkove: string): longint;
function SYMETRICKY( cislo:integer):boolean;
function DVOJKOVE_PRVOCISLO( cislo:string):boolean;

IMPLEMENTATION

var i:integer;
function DVOJKOVE_CISLO( desiatkove:word ): string;
                    var  d, z: string;
                    begin
                          d:='';
                          while desiatkove>0 do
                             begin
                                str(desiatkove mod 2, z);
                                d:= d + z;
                                desiatkove := desiatkove  div 2
                             end;
                           DVOJKOVE_CISLO := d
                    end;
function DESIATKOVE_CISLO( dvojkove: string): longint;
                     var i, d, nasobok,dvoj, chyba: integer;
                 begin
                     nasobok:=1; d:=0;
                     for i:=length(dvojkove) downto 1 do
                        begin
                            val(dvojkove[i],dvoj,chyba);
                            d:= d + dvoj * nasobok;
                             nasobok:= nasobok *2
                        end;
                     DESIATKOVE_CISLO:= d
                 end;
function DESIATKOVE_CISLO2( dvojkove: string): longint;
                     var i, d, dvoj, chyba: integer;
                  begin
                     d:=0;
                     for i:=1  to length(dvojkove)  do
```

```
                    begin
                      val (dvojkove[i], dvoj, chyba);
                      d:= d * 2 + dvoj ;
                    end;
                      DESIATKOVE_CISLO2:= d
                end;
function SYMETRICKY( cislo:integer):boolean;
                    var  bin: string;
                         dlzka_bin: byte;
                  begin
                     SYMETRICKY:=false;
                     bin:=DVOJKOVE_CISLO(cislo);
                     dlzka_bin:= length(bin);
                     for i:=1 to dlzka_bin div 2 do
                      if bin[i] <> bin[dlzka_bin-i+1] then exit;
                     SYMETRICKY:=true;
                  end;
function DVOJKOVE_PRVOCISLO( cislo:string):boolean;
               begin
                  if  PRVOCISLO( DESIATKOVE_CISLO( cislo ) )
                        then  DVOJKOVE_PRVOCISLO:=true
                        else  DVOJKOVE_PRVOCISLO:=false
               end;

END.
```

```
Unit POLIA;

INTERFACE

  const h = 50;
  type index = 0..h;
       postupnost = array [index] of integer;
       matica = array [index, index] of integer;
  procedure CITAJ_POSTUPNOST (var a : postupnost; n : index);
  procedure CITAJ_MATICU (var a : matica; m,n : index);
  procedure PIS_POSTUPNOST (a : postupnost; n : index);
  procedure PIS_MATICU (a : matica; m,n : index);
  function SUCET_POSTUPNOSTI( a: postupnost; n: index): integer;
  procedure NULOVANIE_POSTUPNOSTI( var a: postupnost;n:index );

IMPLEMENTATION

  procedure CITAJ_POSTUPNOST;
    {Precita z klavesnice n celych cisel a ulozi ich do postupnosti a[1], a[2],...,a[n]. }
    var i : index;
  begin
    writeln ('Zadaj postupnost celych cisel');
    for i:=1 to n do begin
      write ('prvok(',i,')= ');
      readln (a[i])
    end
  end;

 procedure CITAJ_MATICU;
    { Precita celociselnu maticu A rozmeru m*n  z klavesnice a ulozi ju do pamati.
  }
    var i,j : index;
  begin
    writeln ('Zadaj prvky celociselnej matice');
    for i:=1 to m do
      for j:=1 to n do begin
        write ('prvok[',i,',',j,']= ');
        readln (a[i,j])
      end
  end;

  procedure PIS_POSTUPNOST;
    { Postupnost celych cisel a[1], a[2],...,a[n]  vypise na obrazovku. }
```

```
    var i : index;
  begin
    for i:=1 to n do write (a[i], ' ');
    writeln
  end;

   procedure PIS_MATICU;
    {  Celociselnu maticu A rozmeru m*n  vypise na obrazovku }
    var i,j : index;
  begin
    for i:=1 to m do
      begin
        for j:=1 to n do write (a[i,j]:4);
        writeln
      end
  end;

 function SUCET_POSTUPNOSTI;
        var i: index; s: integer;
      begin
        s:=0;
        for i:=1 to n do s:= s+ a[i];
        SUCET_POSTUPNOSTI:=s
      end;

 procedure NULOVANIE_POSTUPNOSTI;
      var i: index;
    begin
      for i:=1 to n do a[i]:=0;
    end;

END.
```

```
Unit MATICE;

INTERFACE

uses POLIA,CISLA;
var i,j,k,l:integer;
   pocet_ban:postupnost;
const bankovka: array[1..11] of integer = (5000, 1000, 500, 200,
                                  100, 50, 20,10, 5, 2, 1);
function VZDIALENOST_MIEST( a: postupnost; n: index):integer;
procedure VYPLAT_SUMU( suma:integer);
function DRUHE_MAX( p: postupnost; n:integer):integer;
procedure VYMENA_RIADKOV_MATICE (var a: matica; m,n: integer);
function MAX( p: postupnost; n:integer): integer;
procedure VELKY_FAKTORIAL( n: integer; var p: postupnost;
                  var obsadene:integer);
procedure OBRAZEC( k, m, n:integer) ;
procedure OTOCENIE_MATICE;
procedure OTOCENIE_MATICE_2;
function PALINDROM(c:longint):boolean;

IMPLEMENTATION

function VZDIALENOST_MIEST;
   begin
     CITAJ_POSTUPNOST( a,n);
     VZDIALENOST_MIEST:=SUCET_POSTUPNOSTI(a,n)
   end;

 procedure VYPLAT_SUMU;
      var  i: index;
    begin
      for i:= 1 to 11 do
        begin
          pocet_ban[i]:= suma div bankovka[i];
          suma := suma mod bankovka[i]
        end
    end;

  function DRUHE_MAX;
       var  max1, max2: integer;
      begin
        max1:= -maxint;
```

```
      max2:= -maxint;
      for i:= 1 to n do
        if p[i] > max1 then
            begin
              max2:= max1;
              max1:= p[i]
            end
        else
          if p[i] > max2  then max2:= p[i];
      DRUHE_MAX:=max2
    end;
procedure VYMENA_RIADKOV_MATICE ;
   { var  j :integer;}
    begin
      for j:= 1 to n do VYMENA_2( a[K,j], a[L,j] )
    end;

function MAX;
    var M : integer;
     begin
       M:= -maxint;
       for i:= 1 to n do
         if p[i] > M then M:= p[i];
       MAX:=M
     end;

procedure VELKY_FAKTORIAL;
  var  k, prenos : integer;
  procedure SPRACUJ_K;
    var i, pom: integer;
     procedure SPRACUJ_POSLEDNY_PRENOS ;
       begin
         while  prenos>0  do
           begin
             inc(obsadene) ;
             p[obsadene]:= prenos mod 10 ;
             prenos := prenos div 10
           end
       end; {SPRACUJ_POSLEDNY_PRENOS}
     begin
       i:=1;
```

```
        while i <= obsadene do
          begin
            pom:= p[i]*k+ prenos ;
            p[i]:= pom mod 10;
            prenos := pom div 10;
            inc(i)
          end;
        SPRACUJ_POSLEDNY_PRENOS
      end;{SPRACUJ_K}
  begin
    p[1]:=1 ; obsadene:=1;
    for k:=1  to  n  do
      begin
        prenos:=0;
        SPRACUJ_K
      end
  end;{ VELKY_FAKTORIAL}

procedure OBRAZEC( k, m, n:integer) ;
  var i: integer;
  procedure KRESLI_RIADOK(x: integer);
    var h,l: integer;
      begin
        for h:= 1 to k do
          begin
            write('*');
            for l:= 2 to n-1 do
              begin
                if (x=1) or (x=m) then write('*');
                if (x>=2) and (x<=m-1) then write(' ')
              end;
            write('*');
            write('    ')
          end;
        writeln
      end;
  procedure VYNECHAJ(x: integer);
    var l: integer;
      begin
        for l:= 1 to x do  write(' ')
      end;
```

```
    begin
      for i:= 1 to m  do
        begin
          VYNECHAJ( m-i );
          KRESLI_RIADOK( i )
        end;
    end ;

procedure OTOCENIE_MATICE;
  var a,b: matica;
      n: index;
  procedure OTOC;
    var i,j :integer ;
  begin
    for i:= 1 to n do
      for j:= 1 to n do  b[ j, n-i+1] := a[ i, j]
  end;
 begin
  readln( n );
  CITAJ_MATICU(a, n, n);
  OTOC;
  PIS_MATICU(b, n, n);
 end;

procedure OTOCENIE_MATICE_2;
    var a: matica; n: index;
        i:integer;
    procedure OTOC_KRUZNICU;
        var j, pom :integer ;
      begin
        for j:= i to n-i  do
          begin
            pom:= a[i, j] ;
            a[ i, j]:= a[n-j+1, i] ;
            a[n-j+1, i]:= a[n-i+1,n-j+1] ;
            a[n-i+1,n-j+1]:= a[j,n-i+1] ;
            a[j,n-i+1]:= pom
          end
      end ;
    begin
      readln( n );
```

```
    CITAJ_MATICU(a, n, n);
    for i:= 1  to  n div 2  do OTOC_KRUZNICU;
    PIS_MATICU(a, n, n);
  end;

function PALINDROM;
   var i,n:integer; p:postupnost;
   procedure VYTVOR_POLE ;
     begin
       n:=0;
       repeat
          inc(n);
          p[n]:= c mod 10;
          c:=c div 10
       until c=0
     end;
 begin
   PALINDROM := false ;
   VYTVOR_POLE;
   for i:= 1 to  n div 2  do
    if  p[i]<>p[n-i+1] then exit ;
   PALINDROM:= true
 end;

END.
```

```
Unit RETAZCE;

INTERFACE

function JE_CISLICA(znak:char):boolean;
procedure ZAMENA(var veta: string);
function NUM_STRING(cislo:longint): string;
procedure MALE_VELKE(var veta: string);

IMPLEMENTATION

function JE_CISLICA(znak:char):boolean;
  begin
    if  (ord (znak) > 48)and (ord(znak)< 59 ) then
        JE_CISLICA:=true
     else JE_CISLICA:=false
  end;
procedure ZAMENA(var veta: string);
   var i: integer;
  begin
    for i:= 1  to  length(veta)  do
    if  veta[i]=' ' then
      begin
        delete( veta,i,1);
        insert('*',veta,i)
      end
  end;
function NUM_STRING(cislo:longint): string;
    var s:string;
        cifra:integer;
   begin
      s:='';
      repeat
        cifra:= cislo mod 10;
        s:= char( cifra + ord ('0')) + s;
        cislo:= cislo div 10;
        until cislo=0;
        NUM_STRING:= s
   end;
procedure MALE_VELKE(var veta: string);
    var i: integer;
    function PODMIENKA: boolean;
       begin
```

```
    PODMIENKA:=( ord(veta[i]) >ord('a') ) and
              ( ord(veta[i])< ord('z') )
                end;
  begin
    for i:=1 to length(veta) do
     if PODMIENKA then  veta[i]:=chr(ord( veta[i]) - 32 )
  end;

END.
```

TEST Č.1.: PROCEDÚRY A FUNKCIE PRE ČÍSELNÉ VÝPOČTY.

1. Prepíšte funkciu SUCET ako procedúru:

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

.....
.....
.....
.....

1 bod

2. Čo vypíše program CO_ROBI1 a CO_ROBI2 ? Napíšte zdôvodnenie.

```
program CO_ROBI1;
  var a,b: integer;
 procedure UROB( var x,y:integer);
  begin
    x:=1; y:=2
 end;
begin
 writeln('dve cisla ');readln(a,b);
 UROB(a,b);
 writeln('a=',a,'  b=',b)
end.
```

.....
.....
.....

```
program CO_ROBI2;
  var a,b: integer;
 procedure UROB(  x,y:integer);
 begin
   x:=1; y:=2
 end;
begin
  writeln('dve cisla ');readln(a,b);
 UROB(a,b);
  writeln('a=',a,'  b=',b)
end.
```

.....
.....
.....

3 body

3. a) Akú hodnotu nadobudne premenná r pre c = 1234567, a prečo?

```
procedure ZISTI(c: longint;var r:longint);
  begin
    r:=1;
    while c>r do r:=r*10;
    r:=r div 10;
  end;
```

.....
.....
.....
.....

2 body

b) Nájdite a opravte chyby vo funkcii SUCET!

```
function SUCET(c,r:longint):integer;
 begin
   SUCET:=c div r;
   while c>100 do
    begin
      r:=r div 10;
      c:=c mod r;
      r:=r div 10;
      SUCET:=SUCET+(c div r);
    end;
 end;
```

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

3 body

c) Napíšte program, ktorý určí súčet cifier celého čísla na nepárnych pozíciách. Je možné využiť predchádzajúce podprogramy.

4 body

TEST Č.1.: PROCEDÚRY A FUNKCIE PRE ČÍSELNÉ VÝPOČTY - ŠPECIFIKAČNÁ TABUĽKA

PRÍKLAD		BODY	SPOLU
1.	postihnutie rozdielu medzi funkciou a procedúrou	1	1
2.	objavenie rozdielu Zdôvodnenie	1 2	 3
3.a) **3.b)** **3.c)**	hodnota premennej r nájdenie chýb oprava chýb vytvorenie programu	2 1 2 4	 3 4
			11 bodov

TEST Č.2.: PROCEDÚRY A FUNKCIE PRE ČÍSELNÉ VÝPOČTY.

1. Nájdite chyby v zápise funkcie MOCNINA, ktorá vypoèíta n-tú mocninu èísla 2:

```
function MOCNINA (n:integer): integer ;          .....
     var i: integer ;                            .....
begin                                            .....
     MOCNINA:=1;                                 .....
     for i:=1 to n do MOCNINA:= MOCNINA*2
end ;
```

1 bod

2. Čo vypíše nasledujúci program, ak na vstupe zadáme hodnotu 15 ?

```
program POKUS;
     var x,y: integer;
     function Z (x:integer ):integer;            .....
     begin                                       .....
          x:= x div 2;                           .....
          Z:= x                                  .....
     end;
begin
     read( x ) ;
     while x>1 do
          begin
               y:= Z(x) + 2 ;
               x:= x div 2
          end;
     writeln( y )
end.
```

4. body

3. a) Čo vykreslí program, ktorý má deklarovanú procedúru OBRAZOK, ak telo tvorí len jeden príkaz OBRAZOK(7) ? Procedúra má tvar:

```
procedure OBRAZOK (n:integer) ;
     var j: integer ;
     procedure ELEMENT (k:integer) ;
          var i: integer ;
     begin
          for i:=1 to K do  case  (i mod 2) of
                                  0: write(' ') ;
                                  !: write('*') ;
                            end
     end ;
begin
     for j:=1 to n do
          begin  ELEMENT( j ) ;
                 writeln ;
          end
end ;
```

4 body

b) Koľko hviezdičiek sa nakreslí pri volaní OBRAZOK(20) ?

2 body

4. **Profesor matematiky je veľmi zaneprázdnený. Pomôžte mu vyhodnotiť písomku, napíšte procedúru, ktorá určí priemernú známku v triede, počet 1, 2, 3, 4, 5 a počet žiakov, koľko písomku nepísalo, ak v triede je „n“ žiakov a príslušné známky sú uložené v poli Z[1...n], pričom hodnota z[i]=0, ak žiak písomku nepísal.**

4 body

TEST Č.2.: PROCEDÚRY A FUNKCIE PRE ČÍSELNÉ VÝPOČTY - ŠPECIFIKAČNÁ TABUĽKA

PRÍKLAD		BODY	SUCET
1.	objavenie rekurzívneho volania funkcie	1	1
2.	pochopenie algoritmu správny výsledok	2 2	 4
3. a) **3. b)**	pochopenie algoritmu obrázok počet hviezdičiek	2 2 2	 4 2
4.	určenie počtov známok výpis výsledkov deklarácia premenných	2 1 1	 4
			15 bodov

TEST Č.3.: BOOLOVSKÉ FUNKCIE.

1. Kedy funkcia CIFRY vracia hodnotu *true*?

```
fuction CIFRY( x: integer ): boolean ;
  var a: postupnost;
      i: integer;
  function PODMIENKA: boolean;
  begin
      PODMIENKA:= ( a [ 1 ] = a [ 2 ] )  and  ( a [ 2 ] = a [ 3 ] )
  end;
begin
       i:=0 ;
        while x>0 do
            begin
                inc(i) ;
                a[i]:= x mod 10 ;
                x:= x div +1000
            end;
        CIFRY:= PODMIENKA
 end ;
```

1 bod

b) Čo vypíše nasledujúca procedúra CISLA?

```
Procedure CISLA;
      var i: integer;
begin
      for i:= 100 to 999 do
          if  CIFRY(i)  then write( i : 4)
end;
```

2 body

2. Napíšte funkciu PODMIENKA, aby nasledujúca procedúra vykreslila štvorec z hviezdičiek o strane A a o hrúbke strany H.

napr. pre A=6 a H=2 procedúra vykreslí :

```
******
******
**    **
**    **
******
******
```

```
procedure STVOREC(A,H:integer) ;
     var i,j: integer ;
 begin
     for i:= 1 to A  do
        begin
          for j:=1 to A  do
              if  PODMIENKA then write('*')
                            else  write(' ') ;
          writeln ;
        end
 end ;
```

3 body

3. Zistite, či sa cifra K nachádza v zápise čísla C aspoň dva krát. Úlohu riešte tvorbou boolovskej funkcie.

4 body

TEST Č.3.: BOOLOVSKÉ FUNKCIE - ŠPECIFIKAČNÁ TABUĽKA.

PRÍKLAD		BODY	SPOLU
1.b)	porozumenie zápisu funkcie správny výsledok a zdôvodnenie	1 2	3
2.	správny výsledok a zdôvodnenie	3	3
3.	vytvorenie požadovaného programu	4	4
			10 bodov

TEST Č. 4.: VEKTORY A MATICE.

1. Doplňte nasledujúci program tak, aby vypočítal hodnotu mnohočlena $A_{n+1}x^n + A_n x^{n-1} + ... + A_2x + A_1$ v bode x:

```
program HODNOTA ;
    type pole: array [ 1..20 ] of integer ;
    var  x: real ;  n,j: integer ;  A: pole ;
    function VYPOCET: .................
        var i: integer ;  h: .............
    begin
        h:= .......
        for i:= .....  to  ......  do  h:= h*x + A[i] ;
        ....................................
    end ;
begin
     read( n ) ;
     read( x ) ;
     for j:=1 to  ......  do read( A[i] );
     write( VYPOCET( . . . . . ) )
end.
```

4 body

2. Procedúru MAXPOLA prepíšte na funkciu MAXPOLA tak, aby vracala najväčší prvok a jeho miesto v postupnosti.

```
procedure MAXPOLA(a:postupnost;n:index;var indx:index;var max:integer);
    var i :index;                                  ..........
  begin                                            ..........
    max:=a[1];indx:=1;                             ..........
    for i:=2 to n do                               ..........
     if a[i]>max then                              ..........
            begin                                  ..........
              max:=a[i]; indx:=i                   ..........
            end;                                   ..........
  end;                                             ..........
```

3 body

3. Napíšte procedúru, ktorá pre vstupné celé číslo N vypíše nasledujúcu štruktúru od 1 do N^2 :

napr. pre N=4

```
 1  2  3  4
 8  7  6  5
 9 10 11 12
16 15 14 13
```

6 bodov

TEST Č. 4.: VEKTORY A MATICE - ŠPECIFIKAČNÁ TABUĽKA.

PRÍKLAD		BODY	SPOLU
1.	Doplnenie parametrov v hlavičke funkcie a správne volanie funkcie	1	
	doplnenie deklarácie premennej „h“ a jej správne nastavenie	1	
	doplnenie parametrov cyklov v deklarácii funkcie VYPOCET a v hlavnom programe	1	
	definovania hodnoty funkcie VYPOCET	1	4
2.	hlavička funkcie	1	
	deklarácia premenných	1	
	príkaz priradenia MAXPOLA:=	1	**3**
3.	správne cykly v procedúre	3	
	premenné v procedúre a v hlavnom programe	2	
	hlavný program	1	**6**
			13 bodov

TEST Č. 5.: REŤAZCE.

1. Čo vypíše program PRVY a program DRUHY ? Napíšte zdôvodnenie.

```
Program PRVY;
   var a,b: string;
  procedure UROB( x,y: string);
     var p: string;
  begin
      p:=a; a:=b; b:=p;
  end;
begin
   readln(a,b);
   UROB(a,b);
   writeln('a=',a,' b=',b)
end.
.....
.....
.....
```

```
Program DRUHY;
   var a,b: string;
  procedure UROB( x,y: string);
    var p: string;
  begin
      p:=a; a:=b; b:=p;
  end;
begin
   readln(a,b);
   UROB(a,b);
   writeln('a=',a,' b=',b)
end.
.....
.....
.....
```

3 body

2. Napíšte procedúru NAPIS_ODZADU, tak aby vypísala dlhé číslo v obrátenom poradí. Uvedomte si typ premennej cislo

```
program  odzadu_cislo;
    var cislo: string;
  procedure napis_odzadu(c:string);
  .....
    begin
  .....
  .....
  .....
  .....
  .....
    end;
  begin
    writeln('dlhe cislo ');readln(cislo);
    napis_odzadu(cislo);
  end.
```

4 body

3. Daný program vypíše stĺpec pozostávajúci zo slov danej vety. Dopíšte procedúru CITAJ_VETU a VYPIS_STLPEC.

```
program stlpec;
   var veta:string;
  procedure  CITAJ_VETU;
      begin
          .....
          .....
      end;
  procedure  VYPIS_STLPEC;
    var i: .....;
        znak: .....
        slovo:.....
```

```
begin
  slovo:='';
  for i:=1 to length(veta)+1 do
    begin
      znak:=.....;slovo:=.....;
      if (znak=' ') or (i=length(veta)) then
          begin
            writeln(slovo);
            slovo:= .....
          end
    end;
end;
begin
   CITAJ_VETU;
   VYPIS_STLPEC;
end.
```

5 bodov

TEST Č. 5.: REŤAZCE - ŠPECIFIKAČNÁ TABUĽKA

PRÍKLAD		BODY	SPOLU
1.	objavenie rozdielu slovné zdôvodnenie	1 2	3
2.	deklarácia premenných správne oddeľovanie cifier konštrukcia nového čísla príkaz výstupu	1 1 1 1	4
3.	telo procedury CITAJ_VETU deklarácie premenných telo procedúry VYPIS_STLPEC	1 1 3	5
			12 bodov