Elena Krelja-Kurelović, prof.

PASCAL

- Skripta sa zadacima i rješenjima -

SADRŽAJ:

I. UVOD U PASCAL

Krajem 60-tih godina Nicklaus Wirth definirao je jezik za programiranje i nazvao ga Pascal. Prvenstvena namjena Pascala je u rješavanju problema "algoritamske prirode" gdje dolazi do izražaja strukturno programiranje. *Strukturno programiranje* karakterizira to da se razvoj programa tj. rješenje problema odvija po dijelovima. Točnije, rješenje čini skup modula od kojih svaki predstavlja i izvršava pojedinačnu, nezavisnu funkciju programa.

Osnovne karakteristike Pascala:

1. Obavezno je deklariranje svih varijabli koje se koriste u programu
2. Određene ključne riječi (naredbe ili dijelovi naredbi), npr. BEGIN, IF, THEN, WHILE… "rezervirane" su i ne smiju se koristiti u druge svrhe
3. Standardni tipovi podataka su:
   - realni – ***real***
   - cjelobrojni – ***integer***
   - logički – ***boolean***
   - znakovni – ***char***
4. Postoje i složeni ili strukturirani tipovi podataka, a to su:
   - niz znakova - ***string***
   - polja – ***array***
   - slogovi – ***record***
   - skupovi – ***set of***
   - datoteke – ***file of***
5. Procedure i funkcijski potprogrami mogu pozivati sami sebe tj. rekurzivno

Pravila pisanja programa:

1. Program se sastoji od **zaglavlja** i **bloka naredbi**. U zaglavlju se definiraju svi potprogrami (funkcije i procedure) koji se koriste, varijable, konstante i novi tipovi podataka. To su sve informacije kompilatoru. Zaglavlje započinje s rezerviranom riječi ***PROGRAM***.
   Blok naredbi predstavlja pravi izvršni kod programa, te započinje rezerviranom riječi ***BEGIN*** a završava s ***END.***

Npr.

```
PROGRAM ime;
VAR a, b, c : tip_podataka;
      O, P : tip_podataka;
```

područje definiranja konstanti, deklariranja varijabli, tipova podataka i potprograma

```
BEGIN
      naredba;
      naredba;
      ...
      neredba;
END.
```

izvršni dio programa

2. U jednom redu piše se jedna naredba, a naredbe se razdvajaju znakom točka-zarez tj. **;**
3. Komentari se pišu unutar vitičastih zagrada

1. Tipovi podataka

Podaci mogu biti različitog tipa, kao npr. cijeli brojevi, realni brojevi, znakovi, logičke vrijednosti (istina i laž) i sl. pa time i varijable poprimaju vrijednosti točno određenog tipa podataka. Uobičajeni, jednostavni tipovi podataka u Pascalu, kao i u većini jezika za programiranje su:

Vrsta - tip podataka	Rezervirana riječ	Najmanja vrijednost	Najveća vrijednost
cijeli brojevi (- i +)	**shortint**	-128	127
cijeli brojevi (samo +)	**byte**	0	255
cijeli brojevi (- i +)	**integer**	-32768	32767
cijeli brojevi (samo +)	**word**	0	65535
cijeli brojevi (- i +)	**longint**	-2147483648	2147483647
realni brojevi	**real**	realni broj s 11-12 znamenki	
znakovi	**char**	bilo koji znakovi na tipkovnici Znak se piše unutar polunavodnika, npr. 'A'	

2. Deklariranje varijabli

Varijable koristimo da nam program bude rješiv nad različitim ulaznim podacima, te da možemo koristiti različite formule pri rješavanju zadatka. Kao što sama riječ kaže, vrijednosti varijable mogu se mijenjati, tj. nisu striktno unaprijed definirane. Varijabla ima svoje ***ime*** i njeno svojstvo je da se neka vrijednost pamti pod tim imenom, te da se ta *vrijednost smije promijeniti* tijekom izvršavanja programa.

Da bi u programu mogli koristiti varijable, prije njihova korištenja, moramo kompilatoru reći koje varijable ćemo koristiti i kakvog tipa će one biti. Definiranjem ***tipa podatka*** čije vrijednosti varijabla može poprimiti, ujedno definiramo i operacije koje možemo izvršavati nad tim varijablama.

Za deklariranje varijabli koristimo rezerviranu riječ ili naredbu **VAR.**

Sintaksa (pravilo) pisanja:

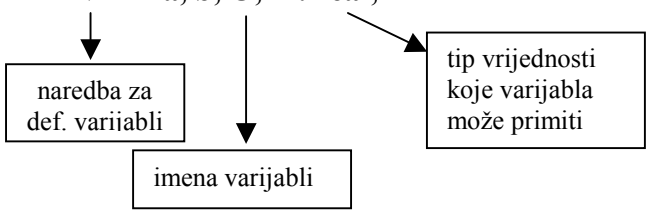
```
VAR   ime_var1, ime_var2 : tip_podatka1;
      ime_var3 : tip_podatka2;
```

Npr. Ako želimo napisati program za izračunavanje opsega i površine pravokutnika, potrebne su nam varijable:

a, b - za stranice pravokutnika
O, P - za opseg i površinu

Ako sve te varijable trebaju pamtiti realne brojeve, onda ćemo ih definirati da budu realnog tipa na slijedeći način:

```
VAR  a, b, O, P : real;
```



3. Definiranje konstanti

Kao i varijable, trebamo definirati i konstante koje ćemo u programu koristiti. Naredba za definiranje konstanti je **CONST.** Karakteristika konstanti je da se *njihova vrijednost ne smije mijenjati*.

Sintaksa (pravilo) pisanja:

CONST ime_konstante = vrijednost ;

Npr. **CONST Pi=3.14 ;**

Kada u programu napišemo ime konstante **Pi** uzima se njena vrijednost tj. 3.14 koju tijekom rada ne možemo mijenjati!

Zaglavlje programa sadrži:

PROGRAM ime_programa;
USES - *popis **modula** koji ćemo koristiti*;
CONST - *deklaracija **konstanti** koje ćemo koristiti*;
TYPE - *definiranje vlastitih, nestandardnih tipova podataka*;
VAR - deklaracija varijabli koje ćemo koristiti;
PROCEDURE - *definiranje **potprograma***;
FUNCTION - *definiranje **potprograma***;

MODUL (eng. Unit) je posebna kompilacijska cjelina koja se kao biblioteka, najčešće već gotovih potprograma, može uključiti u bilo koji program. Često se koristi ***modul*** **Crt** koji omogućava kontrolu nad ekranom, kodovima tipkovnice i bojama, prozorima i zvukovima, a on se poziva na sljedeći način:

```
PROGRAM proba;
  USES Crt;
  VAR a, b : integer;
       x, y : real;
  itd.
```

II. PISANJE PROGRAMA U PASCAL-u

Najpoznatiji program Pascal je Borlandov ***Turbo Pascal***, ali za njega treba imati licencu. Jedna od besplatnih verzija ovog programa je i ***Virtual Pascal*** koji se može downloadati sa adrese: http://www.vpascal.com/download.html. Rađen je za DOS operativni sustav, pa ga pokrećemo kao i ostale DOS programe. Samo pisanje programskog kôda radimo u ***editoru*** Pascala. Kada želimo testirati program, on se najprije mora kompilirati tj. prevesti u strojni jezik. Možemo ga kompilirati (**Compile**) samo u radnu memoriju ili na disk čime dobivamo izvršnu tj. ∗.exe datoteku tog našeg programa. Program pokrećemo naredbom (**Run**). Izbornik pozivamo tipkom **Alt**.

1. OSNOVNE NAREDBE

1.1. Pridruživanje vrijednosti

Da bi nekoj varijabli dodijelili vrijednost, koristimo znak za prodruživanje, a to je **:=** tj. "dvotočka i jednako".

Npr. A := 10 čitaj*: A ima vrijednost 10*
X := 3
ili X := X+A čitaj*: X uvećaj za vrijednost variajble A i rezultat spremi u varijablu X*
nova vrijednost varijable **X** je 13 *(jer je 3+10=13)*

VAŽNO: Izraz koji se nalazi s desne strane znaka **:=** najprije se izračuna, a zatim dodjeli varijabli s lijeve strane!

1.2. Naredba za unos

Ova naredba omogućuje pisanje općenitijih programa jer se vrijednosti varijablama zadaju izvan programa, u fazi njegova izvršavanja. Moramo paziti da unesene vrijednosti odgovaraju definiranom tipu podataka za te varijable.
Općenito značenje naredbe je ***prekid izvršavanja programa i čekanje da se unesu vrijednosti navedenih varijabli***. Potom će utipkane vrijednosti biti dodijeljene odgovarajućim varijablama i program će se nastaviti izvršavati.

Naredba je: **READ** (*ime varijable, ime varijable…*);
ili **READLN** (ime variajble…);

Razlika između READ i READLN je samo u tome što se kod READLN kursor premješta na početak novog retka.
Npr. Ako želimo unijeti vrijednosti za stranice pravokutnika – varijable a i b, u programu ću upisati naredbu:

READ (a,b); ili
READLN (a,b);

1.3. Naredba za ispis

Da bi vidjeli rješenja našeg problema, program nam mora ispisati određene vrijednosti, a to čini naredbom za ispis. Tom naredbom možemo na ekranu vidjeti ispisan neki tekst ili vrijednosti varijabli.

Naredba je: **WRITE** (ime varijable ili tekst u polunavodnicima);
ili **WRITELN** (ime varijable ili tekst u polunavodnicima); - kursor se premješta u novi redak

a) ***Ispis teksta***

Npr. WRITELN (' Opseg i površina pravokutnika ');
WRITE (' PROGRAMIRANJE ');

tekst obavezno staviti u **polunavodnike**!

b) Ispis vrijednosti varijabli

Npr. WRITE (O, P); -ispisat će se vrijednost varijable O (opseg) i vrijednost varijable P (površina)

c) Kombiniranje ispisa teksta i vrijednosti varijabli ili vrijednosti aritm. izraza

Npr. WRITELN ('Rješenje je' , X);

WRITELN ('Zbroj brojeva je:' , A+B)

d) ***Formatirani ispis*** – koristimo kod realnih brojeva

Npr. varijabla x:=3.1438963125 a želimo ispis samo na 2 delimale

WRITE (X:5:2); ⟶ - za ispis vrijednosti varijable X rezervirano je **ukupno 5 mjesta**, a od toga **2 mjesta za decimalni dio** (ispisalo bi 3.14)

Zadatak: Što će ispisati ovaj program?

```
PROGRAM vjezba_1;
  VAR a, b: integer;
BEGIN
   a := 2;
   b := 3;
   writeln ( 'Vrijednosti varijabli su: '  , a, b);
   writeln ( 'Zbroj je: '  , a+b );
   writeln ( 'a+b' );
   b := b + 7;
   writeln ( 'Nova vrijednost varijable b je '  , b);
   readln;
END.
```

Zadaci:

1. Napiši program koji će izračunati opseg i površinu pravokutnika za bilo koje vrijednosti stranica.

```
PROGRAM pravokutnik;
  VAR a, b, O, P : real;
BEGIN
  write ('Upiši vrijednosti stranica pravokutnika:');
  readln (a,b);
  O := 2 * (a+b);
  P := a * b;
  writeln ('Opseg je: ' , O);
  writeln ('Površina je: ' , P);
  readln;
END.
```

2. Napiši program koji će za bilo koji upisani broj ispisati njegovog prethodnika i slijedbenika.
   Upute: prethodnik := broj - 1;
   slijedbenik := broj +1;

3. Napiši program koji će vrijednost u kunama pretvoriti u EURE. Vrijednost tečaja se upisuje prilikom startanja programa jer je promjenjiva!

```
PROGRAM mjenjacnica;
  VAR iznos, kn, t : real;
BEGIN
  write ('Upisi vrijednost tecaja za EURO ');
  readln (t);
  write ('Upisi iznos u kn ');
  readln (kn);
  iznos:=kn/t;
  writeln ('Iznos u EURIMA je ' , iznos);
  readln;
END.
```

2. ARITMETIČKI IZRAZI

Služe da bismo mogli izračunati tj. obraditi ulazne podatke u cilju dobivanja izlaznih podataka tj. rezultata. Pri tome se koristimo standardnim operacijama i funkcijama.

1. ***Standardne operacije*** :

- +, -, * ⇨ definirane su na INTEGER i REAL tipu podataka
- / ⇨ "uobičajeno" dijeljenje je definirano samo na REAL tipu podataka
- DIV cjelobrojno dijeljenje
- MOD ostatak cjelobrojnog dijeljenja

} - ove operacije su definirane samo na INTEGER tipu podataka

Npr. ako je A:=17, a B:= 3 i ako je: C := A **DIV** B; tj. C:=17 DIV 3
D := A **MOD** B; tj. D:=17 MOD 3

Rezultat je: C:= 5, a D:=2

Napomena: Sve varijable u ovom primjeru **moraju biti deklarirane kao cijeli broj** tj. INTEGER ili neki njegov podtip, a nipošto kao REAL!

2. ***Standardne funkcije***:

Popis najvažnijih funkcija:

Poziv funkcije	Broj/varijabla uz funkciju je:	Rezultat je	Opis
ord(x)	integer	integer	redni broj
pred(x)	integer	integer	prethodnik
succ(x)	integer	integer	sljedbenik
random(x)	integer	integer	slučajni boj u intervalu od 0 do x
abs(x)	integer, real	integer/real	apsolutna vrijednost broja x
sqr(x)	integer, real	integer/real	kvadrat broja x
frac(x)	real	real	decimalni dio vrijednosti realnog br. x
int(x)	real	real	cijeli dio vrijednosti br. x, prikazan kao realni br.
round(x)	real	integer	zaokružena vrijednost broja x prikazana kao cijeli br.
trunc(x)	real	integer	cijeli dio realnog broja x
sqrt(x)	integer, real	real	drugi korjen broja x

Napomena: Pascal nema funkciju koja bi računala $\mathbf{x^y}$, pa se to rješava korištenjem eksponencijalne i logaritamke funkcije ovako: **exp (y * ln(x))**

Zadatak: Napiši program koji će za neki realan broj posebno ispisati njegov cijeli i decimalni dio.

```
PROGRAM cijeli_decim;
 VAR x, d: real;
       c : integer;
BEGIN
 write ('Upisi realan broj ') ;
 readln (x);
 c:=trunc(x);
 d:=frac(x);  ------------>   ili   d:= x-c;
 writeln ('Cijeli dio broja je: ', c);
 writeln ('Decimalni dio broja je: ', d:4:3);  ------> formatirani ispis varijable realnog tipa
 readln;
END.
```

Zadaci za vježbu - aritmetički izrazi

I. Pretvori matematički izraz u aritmetički izraz koji odgovara Pascalu:

1) $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

2) $D = \sqrt{a^2 + b^2 + c^2}$

3) $x = \frac{a+b}{a-c} + \frac{a}{c-b}$

4) $y = \frac{|a-b|}{2a} + \frac{3(a+4c)}{b^2}$

5) $x_1 = -\frac{p}{2} + \sqrt{\left(\frac{p}{2}\right)^2 - q}$

6) $V = h + \frac{hr}{P-r}$

II. Izračunaj rezultat aritmetičkih izraza vodeći računa o prioritetu operacija:

1) y := a MOD b + c - a DIV d ako je: a=15, b=4, c=3, d=2
2) x := ROUND(a * b ^ 2 / (b-a)) ako je: a= 2, b=6
3) z := b - TRUNC((a + b) / b + a) ako je: a= 1, b=2

3. Programi linearne strukture

1. Napiši program koji će izračunati zbroj kvadrata dva broja.
2. Napiši program koji će za bilo koji dvoznamenkasti broj:
   a) ispisati njegovu znamenku na mjestu desetica i znamenku na mjestu jedinica.
   b) te znamenke zbrojiti.
3. Napiši program koji će izračunati prosječnu potrošnju benzina nekog automobila, ako se unose podaci o broju prijeđenih kilometara i količini potrošenog benzina.

   Formula glasi: $\mathbf{prosj.potrošnja} = \frac{\mathbf{litre\ \ benzina}}{\mathbf{km}} * 100$
4. Napiši program koji će izračunati koliko korisnik jedne kartice može poslati SMS poruka ako se unese stanje na kartici i cijena koštanja 1 SMS poruke.
5. Napiši program koji će upisani kut u stupnjevima pretvoriti u radijane. Formula glasi:

   $\boldsymbol{radijani = kut * \frac{\pi}{180}}$.
6. Napiši program koji će računati udaljenost dviju točaka u ravnini. Unose se koordinate točaka T1 i T2. Formula glasi: $\boldsymbol{d = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}}$
7. Napiši program koji će računati površinu trokuta ako su poznati njegovi vrhovi. Unose se koordinate vrhova V1, V2 i V3. Formula glasi:

   $\boldsymbol{p = \frac{1}{2}\left|x1(y2 - y3) + x2(y3 - y1) + x3(y1 - y2)\right|}$
8. Napiši program koji će uneseno vrijeme u minutama pretvoriti u sate i minute. Npr. za 146 min. treba ispisati rješenje 2 sata i 26 min.
9. Napiši program koji će razlomak - posebno se učitava brojnik a posebno nazivnik, ispisati u nepravom obliku. Pretpostavka: brojnik je veći od nazivnika.

   npr. brojnik = 7, nazivnik = 3 ➔ rezultat je: 2 i 1 / 3
   brojnik =19, nazivnik = 4 ➔ rezultat je: 4 i 3 / 4
10. Napiši program koji će izračunati slobodan prostor na disketi ako se na nju snimi jedna datoteka određene veličine. Veličinu datoteke izrazit ćemo u KB, a veličinu praznog prostora na disketi izrazi:
    a) u KB
    b) u byte

    Disketa je kapaciteta 1.44 MB.
    Odnos: 1 MB = 1024 KB kao i
    1 KB = 1024 byte.

III. NAREDBE IZBORA

Kada u programu moramo odabrati jedan put kojim će se program nastaviti, ovisno o rezultatu uvjeta koji smo postavili, koristimo naredbu za selekciju ili izbor. Primjenom te naredbe moguće je provjeriti postavljeni uvjet i na temelju njegova ishoda odlučiti što dalje činiti. Zbog toga se kaže da ova naredba čini osnovu pisanja "inteligentnijih" programa.

Naredba za izbor ima dva oblika:
1. **IF – THEN** (AKO je uvjet zadovoljen–TADA učini…)
2. **IF – THEN –ELSE** (AKO je uvjet zadovoljen – TADA učini jedno – INAČE učini drugo)

Pravilo pisanja naredbe:

1. **IF** logički izraz (uvjet) **THEN** naredba;

2. **IF** logički izraz (uvjet) **THEN** naredba1 **ELSE** naredba2;

Za pisanje logičkih izraza tj. uvjeta koji moraju bili ISTINITI ili LAŽNI koristimo se slijedećim ***logičkim relacijama***:
- < "manje od"
- > "veće od"
- = "jednako"
- <= "manje ili jednako"
- >= "veće ili jednako"
- <> "različito"

No, možemo u logičkim izrazima imati i ***logičke operacije***:
- **NOT** logička negacija (ISTINU mijenja u LAŽ i obratno)
- **AND** logičko "I"
- **OR** logičko "ILI"
- **XOR** isključivo "ILI"

Značenje ovih logičkih operacija isto je kao i u matematici i kod logičkih sklopova ili "vrata". Izraze povezane ovim operatorima moramo staviti u zagrade!

Logička funkcija:
- **ODD** (**c**); *gdje je* **c** *varijabla/broj* cjelobrojnog tipa!

- rezultat te funkcije je **true** ako je **c** ***neparan broj*** ili **false** ako je **c** ***paran broj***

Prioriteti izvršavanja operacija (aritmetičkih i logičkih)

1. izraz u zagradi
2. aritmetička funkcija
3. negativan predznak
4. NOT
5. *, /, DIV, MOD i AND
6. +, -, OR i XOR
7. relacije <, >, =, <=, >=, <>

Zadaci:

1. Ako želimo ispitati je li učitani broj paran ili neparan, to možemo učiniti ovako:

```
PROGRAM par_nepar;
  VAR broj: integer;
BEGIN
  write ('Upisi broj: ');
  readln (broj);
  IF odd(broj) THEN writeln('Broj je neparan') ELSE writeln('Broj je paran');
  readln;
END.
```

ako je broj **neparan**, rezultat funkcije će biti **TRUE** pa će se izvršiti **naredba uz THEN**, a u suprotnom će rezultat funkcije biti **FALSE** pa će se izvršiti **naredba uz ELSE**

2. Treba učitati 2 broja i ispisati manji pa veći broj.

```
PROGRAM manji_veci;
  VAR a, b: integer;
BEGIN
  write ('Upisi 2 broja: ');
  readln (a,b);
  IF a<b THEN write(a,b) ELSE write(b,a);
  readln;
END.
```

3. Ako je učitani broj djeljiv sa 3, ispisati rezultat djeljenja, a ako nije broj treba umanjiti za ostatak djeljenja s 3 i ispisati taj dobiveni rezultat koji je ujedno i prvi manji broj djeljiv sa 3.

```
PROGRAM djeljiv;
  VAR broj, a : integer;
BEGIN
  write ('Upisi broj: ');
  readln (broj);
  a:= broj mod 3;
  IF a = 0 THEN write(a) ELSE
                          BEGIN
                            broj:=broj-a;
                            write ('Prvi manji broj djeljiv sa 3 je ' , broj);
                          END;
  readln;
END.
```

1. Programi strukture grananja

1. Napiši program koji će riječima ispisati da li je učitani broj paran ili neparan, ali bez korištenja funkcije odd!

2. Napiši program koji će izračunati drugi (kvadratni) korjen broja ukoliko se on može izračunati, a u suprotnom će se ispisati poruka "*Nedozvoljena vrijednost*".

3. Napiši program za ovako definiranu funkciju: $z = \begin{cases} x^2 & za \quad x \geq y \\ x * y & za \quad x < y \end{cases}$

4. Napiši program koji će ispitati je li godina prijestupna. Godina je prijestupna ako je djeljiva s 4 i nije djeljiva s 100 ili je djeljiva s 400.

5. Napiši program koji će od tri broja ispisati najveći.

6. Napiši program koji će tri broja poredati po veličini, od manjeg prema većem.

7. Napiši program koji će ispitati da li učitane stranice čine trokut. Stranice čine trokut ako vrijedi da je a+b>c ili a+c>b ili b+c>a.

8. Napiši program koji će ispitati da li stranice a, b i c čine pravokutan trokut. Trokut je pravokutan ako vrijedi da je kvadrat nad hipotenuzom jednak zbroju kvadrata nad katetama tj. $c^2=a^2+b^2$

9. Napiši program koji će ispitati kakav je trokut ako znamo vrijednosti njegovih stranica. Trokut može biti raznostraničan, jednakokračan i jednakostraničan.

10. Napiši program koji će simulirati rad logičkog **ILI** sklopa.
11. Napiši program koji će isptati je li učitani broj ocjena (1-5) i ako je odgovor potvrdan, da li je ta ocjena prolazna ili nije.

12. Napiši program koji će izračunati ocjenu učenika prema postotku bodova na testu. Unosi se broj bodova učenika i ukupni broj bodova na testu.
    a) 50% - 60% dovoljan
    b) 61% - 75% dobar
    c) 76% - 90% vrlo dobar
    d) 91% - 100% odličan

IV. NAREDBE ZA PONAVLJANJE

U programima često postoji potreba da se dio programa izvede više puta. Pascal ima 3 naredbe za ponavljanje ili kako ih često nazivamo "**petlje**". One omogućuju ponavljanje grupe naredbi i provjeru uvjeta okončanja. To su:

1. **FOR** petlja
2. **WHILE** petlja
3. **REPEAT** petlja

FOR petlja omogućuje bezuvjetno ponavljanje naredbi točno određeni broj puta, pa kažemo da je to prebrojivo ponavljanje.
WHILE i **REPEAT** petlje ponavljaju naredbe ovisno o tome je li postavljen uvjet ispunjen ili nije, pa kažemo da je to uvjetovano ponavljanje.

1. PREBROJIVO PONAVLJANJE

1.1. FOR petlja

Ponavlja naredbu ili veći broj naredbi zadani broj puta, pa mora imati "brojač" koji broji koliko puta se je ponavljanje izvršilo. Za brojač trebamo definirati početnu vrijednost – od nje počinje brojati i završnu vrijednost. Kada brojač dostigne zadanu završnu vrijednost, ponavljanje se prekida. Tada se program nastavlja sa prvom naredbom nakon FOR petlje.

1. Treba napisati program koji će 10 puta na ekranu ispisati znak #, u istom redu.

```
PROGRAM  znakovi;
 VAR b : integer;
BEGIN
 FOR b:= 1 TO 10 DO
        write ('#');
 readln;
END.
```

ova varijabla je brojač

varijabli **b** pridružili smo početnu vriiednost i definirali završnu

2. Treba napisati program koji će ispisati sve borjeve od 1 do 20, ali unatraške!

```
PROGRAM Unatrag;
 VAR b: integer;
BEGIN
 FOR b:=20 DOWNTO 1 DO
        writeln (b);
 readln;
END.
```

DOWNTO omogućuje brojanje u natrag, pa *početna vrijednost mora biti veća od završne*!

3. Napiši program koji će ispisati brojeve od 1 do N i pored svakog broja njegov kvadrat.
4. Napiši program koji će tražiti unos N ocjena i prebrojati koliko ima odičnih ocjena.
5. Program koji zbraja sve brojeve od 1 do nekog broja N kojeg zadajemo prilikom izvršavanja programa:

```
PROGRAM zbroji;
  VAR  b, n, z : integer;
BEGIN
  z:=0;
  writeln ('Upiši do kojeg zbroja želiš ići: ');
  read (n);
  FOR b:=1 to n DO
        z:=z+b;
  writeln ('Zbroj brojeva je: ' , z);
  readln;
END.
```

početna vrijednost varijable u koju ćemo spremati ZBROJ

iza **DO** ne pišemo **;**

6. Program koji računa prosjek N unesenih ocjena.

```
PROGRAM prosjek_ocj;
  VAR b, n, ocj, z: integer;
        pr : real;
BEGIN
  writeln ('Koliko ocjena želiš upisati? ');
  readln (n);
  FOR b:=1 TO n DO
      BEGIN
        write ('Upiši ocjenu: ');
        readln (ocj);
        IF (ocj<1) AND (ocj< 5) THEN
            BEGIN
                write ('Upisani broj nije ocjena, upiši ponovo!');
                readln (ocj);
            END;
        z:=z+ocj;
      END;
 pr:= z/n;
  writeln ('Prosjek tvojih ocjena je ' , pr);
  readln;
END.
```

b = je brojač
n = broj ocjena koje ćemo upisati
z = zbroj ocjena
pr = prosjek ocjena

početak FOR petlje

provjerava je li upisani broj za varijablu **ocj** zaista ocjena tj. broj od 1 do 5

kraj FOR petlje

7. Napiši program koji će računati prosjek samo pozitivnih ocjena, a unosi se N ocjena.
8. Napiši program koji će izračunati faktorijel broja N.

2. UVJETOVANO PONAVLJANJE

2.1. WHILE petlja

While = " sve dok"

Naredba ili više njih se ponavljaju sve dok je postavljeni uvjet istinit tj. zadovoljen. Kada uvjet postane lažan, program se nastavlja naredbom iza WHILE petlje.

Pravilo pisanja naredbe:

a) ponavljanje 1 naredbe sve dok je uvjet istinit

```
WHILE uvjet DO
        naredba;
```

b) ponavljanje više naredbi sve dok je uvjet istinit

```
WHILE uvjet DO
    BEGIN
        naredba1;
        naredba2;
    END;
```

Ova se petlja često koristi da kontrolu ulaznih vrijednosti.

Npr. Ako trebamo upisivati ocjene, ta bi se provjera učinila ovako:

```
PROGRAM ocjena;
VAR ocj: integer;
BEGIN
writeln ('Upiši ocjenu');
readln (ocj);
WHILE (ocj<1) OR (ocj>5) DO
   BEGIN
        write ('Upiši ocjenu od 1 do 5');
        readln (ocj);
   END;
writeln ('Upisani broj je ocjena');
readln;
END.
```

Ove naredbe WHILE petlje između BEGIN i END će se ponavljati sve dok je uvjet **istinit**.

Ako je uvjet odmah lažan, cijela WHILE petlja se preskače

2.2. REPEAT – UNTIL petlja

Ova petlja je suprotna od prethodne u tom smislu što omogućuje ponavljanje naredbe ili više naredbi *sve dok je uvjet lažan* tj. sve dok uvjet ne postane istinit. Kada uvjet postane istinit, prekida se izvođenje ove petlje i program se nastavlja slijedećom naredbom.

Pravilo pisanja naredbe:

```
REPEAT
  naredba1;
  naredba2;
  naredba3;
UNTIL uvjet;
```

Uz ovu petlju **ne trebamo** pisati BEGIN i END ako želimo ponavljati više naredbi

naredbe će se ponavljati sve dok je ovaj uvjet **lažan**

I ovu naredbu možemo koristiti za provjeru ulaznih vrijednosti, ali moramo paziti kako ćemo postaviti uvjet:

Za isti primjer, rješenje bi bilo ovakvo:

```
PROGRAM ocjene;
VAR ocj : integer;
BEGIN
REPEAT
   writeln('Upiši ocjenu');
   readln(ocj);
UNTIL (ocj>0) AND (ocj<6);        ⟶ uvjet je suprotan od onoga kod WHILE!!!
writeln('Upisani broj je ocjena');
readln;
END.
```

Razlika između ovih dviju petlji je da će se korištenjem **REPEAT-UNTIL** petlje naredbe unutar nje **barem jednom izvršiti** jer se uvjet ispituje tek na kraju petlje!
Kod **WHILE** petlje uvjet se ispituje odmah na početku, pa ako je lažan, naredbe unutar petlje se neće niti jednom izvršiti.

Koristan primjer:

```
REPEAT UNTIL KeyPressed;
```

Da bi mogli koristiti KeyPressed moramo odmah na početku definirati da ćemo koristiti modul CRT tj. odmah nakon PROGRAM ime_programa; pišemo
USES Crt;

- tom naredbom zamjenjujemo onaj Readln na kraju programa tj. njome kažemo računalu da čeka dok ne pritisnemo bilo koju tipku na tipkovnici
- može služiti i kao pauza bilo gdje unutar programa

3. Programi s naredbama za ponavljanje

a) **FOR petlja**:

1. Napiši program koji će ispisati sve parne brojeve od 1 do N.
2. Napišpi program koji će prebrojati koliko puta se pojavljuje znamenka 3 u brojevima od 1 do N, gdje je $N < 100$.
3. Napiši program koji će tražiti unos prva dva člana aritmetičkog niza i ispisati:
   a) prvih N članova
   b) sumu prvih N članova
   c) u prvih N članova, koliko ima neparnih brojeva
4. Napiši program kojim će se unositi broj neopravdanih sati izostanka za N učenika i izračunati koliko učenika ima neku odgojnu mjeru zbog neopravdanih izostanaka.
5. Napiši program koji će za brojeve od 1 do N računati drugi korijen i prebrojati koliko tih korijena su cijeli brojevi, te ih ispisati.

b) **Petlja u petlji (FOR)**:

6. Napiši program koji će ispisati zvjezdice u N redova, u sljedećem obliku:

npr. za N=8 ispis treba biti:

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

```
PROGRAM zvjezdice;
VAR a, b, n : integer;
BEGIN
write ('Upiši broj redaka');
readln (n);
FOR a:=1 TO n DO
   BEGIN
      FOR b:=1 to a DO
           write ('*')
      writeln;
   END;
readln;
END.
```

7. Napiši program koji će ispisati tablicu množenja za N<= 10.
8. Napiši program koji će naći koji broj u intervalu od 1 do N ima najveći broj djelitelja.

c) **WHILE i REPEAT petlje**:

9. Napiši program koji će ispisati redni broj člana aritmetičkog niza i njegovu vrijednost kada suma članova tog niza postane veća od 100. Prva dva člana aritmet. niza se upisuju.
10. Napiši program koji će simulirati rad bankomata na način da se upisuje stanje na računu (jednom - na početku korištenja programa) i upisuju se iznosi za isplatu korisniku. Program treba dozvoliti isplatu i izračunati novo stanje sve dok je to stanje "u plusu" ili se upiše broj 1 koji označava kraj korištenja "bankomata".
11. Program ispisuje sve brojeve iz intervala od 1 do N koji se mogu napisati kao zbroj kvadrata dvaju cijelih brojeva. Npr: $2=1^2+1^2$, $5=2^2+1^2$, $8=2^2+2^2$, $10=3^2+1^2$ itd.

```
PROGRAM zbroj_kvadr;
VAR a, i, n, x, y : integer;
BEGIN
write ('Upisi gornju granicu:');
readln (n);
FOR i:=1 TO n DO
      BEGIN
      a:= round(sqrt(i));
      x:=x+1;
      WHILE (x>=1) AND (x<=a) DO
            BEGIN
             FOR y:=1 TO x DO
                  IF i=sqr(x)+sqr(y) THEN writeln(i, '=', x, '*', x, '+', y, '*', y) ;
             x:=x+1;
             END;
      END;
readln;
END.
```

12. Program skraćuje razlomak $\frac{\mathbf{A}}{\mathbf{B}}$ tražeći najvećeg zajedničkog djelitelja.

I način:

```
PROGRAM razlomak;
VAR a, b, i, x, a1, b1 : integer;
BEGIN
write ('Upisi brojnik ');
readln (a);
write ('Upisi nazivnik ');
readln (b);
IF a<b THEN x:=a  ELSE x:=b;
FOR i:=2 TO x DO
      IF (a mod i=0)   AND (b mod i=0) THEN
         BEGIN
            a1:= a div i;
            b1:= b div i;
          END;
IF (a1<>0) AND (b1<>0) THEN writeln ('Skraceni razlomak je ', a1, '/', b1)
                              ELSE writeln ('Razlomak se ne moze skratiti');
readln;
END.
```

II način – **Euklidov algoritam** za nalaženje najvećeg zajedničkog djelitelja:

Ako su A i B brojevi za koje treba naći najveći zajednički djelitelj, najprije se definiraju 2 pomoćne varijeble X i Y kojima se inicijalno pridruže vrijednosti A i B. Zatim treba ***ponavljati*** sljedeći postupak ***sve dok je*** **X<>Y**:

a) ako je X>Y, nova vrijednost za X je X-Y
b) ako je X<Y, nova vrijednost za Y je Y-X

Na kraju ovog postupka, kada bude ispunjeno da je **X=Y**, dobit ćemo najveći zajednički djelitelj za brojeve A i B, a to je X ili Y.

```
PROGRAM euklid;
VAR a, b, x, y : integer;
BEGIN
write ('Upisi brojnik ');
readln (a);
write ('Upisi nazivnik ');
readln (b);
x:=a;
y:=b;
WHILE x<>y DO
          IF x>y THEN x:=x-y  ELSE y:=y-x;
a:=a div x;
b:=b div x;
writeln ('Skraceni razlomak je ', a, '/', b) ;
readln;
END.
```

13. Program provjerava je li učitani broj prosti - prim broj (djeljiv samo s 1 i sa samim sobom).

```
PROGRAM prosti_br;
VAR a, b, x : integer;
BEGIN
write ('Upisi jedan cijeli broj ');
readln (a);
b:=2;
x:=round(sqrt(a));
WHILE  (a mod b<>0) AND (b<=x) DO
          b:=b+a;
IF b<X THEN writeln ('Broj nije prosti br.')  ELSE writeln ('Broj je prosti broj');
readln;
END.
```

14. Napiši program koji će ispisati sve proste - prim brojeve od 1 do N.

V. NIZOVI ZNAKOVA - *STRING*

Ponekad u programima moramo raditi i sa podacima koji nisu brojevi, kao npr. ako želimo upisati nečije *ime i prezime* i neke operacije s time napraviti. Tada se radi o programiranju tzv. "nebrojčanih" problema.

NIZ je struktura podataka koja je definirana nad *znakovnim* tipom kao osnovom, pa sadrži bilo kakav skup znakova koji čini jednu cjelinu. Inače, znakovi i njihovo uređenje definirani su ASCII kodom znakova.

Na primjer, nizovi znakova su:

a) '----------'
b) 'S.S. Jurja Dobrile'
c) 'OK'
d) '0123456789'

Kao što se vidi iz primjera, svaki niz znakova, ako se koristi unutar programa, mora se staviti u ***polunavodnike***.
Osim toga, svaki niz znakova ima svoju ***duljinu***. To je maksimalan broj znakova koje taj niz može poprimiti. Duljina niza mora biti u ***intervalu od 1 do 255 znakova***. Taj podatak nam je bitan da bi mogli definirati varijablu tipa **niz** tj. **string**.

Deklariranje nizovne varijable:

Varijablama tipa ***STRING*** (NIZ) možemo dodijeliti vrijednost *niza znakova*. To raadimo kao i za varijeble ostalih tipova:

VAR ime, skola : **STRING[25]**;
predmet : **STRING[15]**;

u uglatim zagradama
pišemo **max. duljinu** niza

Pridruživanje nizovnih vrijednosti:

1. *Naredbom za **dodjeljivanje***

Naredbom za dodjeljivanje **:=** dodjeljujemo vrijednost koju pišemo s desne strane te naredbe varijabli koju naznačimo s lijeve strane. Odnosno, postupak je isti kao i dodjeljivanje vrijednosti bilo kojoj drugoj varijabli.

pr. ime := 'Marko';
predmet := 'informatika';

VAŽNO: Ne smijete zaboraviti vrijednost tj. niz znakova napisati u **polunavodnicima**!

2. *Naredba za **unos** nizovnih vrijednosti*

Naredba za unos je READ ili READLN, pa se koristi i za unos nizovnih vrijednosti. Naredba se piše prema poznatom pravilu, a unutar zagrada pišemo ime nizovne varijable čija će se vrijednost unijeti preko tipkovnice u trenutku izvršavanja programa.

npr. READLN (ime); program će čekati da unesemo neko ime preko tipkovnice
VAŽNO: tada ne pišemo polunavodnike jer direktno unosimo **vrijednost**

Kako doći do pojedinog znaka unutar niza?

Pošto svaki znak unutar niza ima svoje mjesto, a mjesta počinju od broja 1, pa do max. broja znakova u nizu, nije problem doći do bilo kojeg znaka unutar niza.

npr. ime := 'Marko'
ime[1] := 'M' vrijednost varijable **ime** na mjestu 1 je **M**
ime[2] := 'a' vrijednost varijable **ime** na mjestu 2 je **a**
ime[3] := 'r' vrijednost varijable **ime** na mjestu 3 je **r**
ime[4] := 'k' vrijednost varijable **ime** na mjestu 4 je **k**
ime[5] := 'o' vrijednost varijable **ime** na mjestu 5 je **o**

ovaj broj upućuje na kojem mjestu unutar znakovne varijable treba gledati vrijednost!

Kako doznati točnu duljinu niza znakova u nekoj varijabli?

Funkcija **LENGTH** = engl. duljina izračunava duljinu niza znakova za varijablu koju stavimo u zagradu. Pošto je to funkcije ona ne može stajati samostalno kao naredbe! Zato je pišemo ovako:

d:=**length** (ime); *gdje je* **d** – varijabla tipa INTEGER u koju će se spremiti vrijednost duljine niza
ime – varijabla tipa STRING čija nas duljina zanima

npr. ako je

a) ime := 'Marko' *onda će nakon* d:=length(ime) *biti* d:=5
b) ime := 'Kristijan' *onda će nakon* d:=length(ime) *biti* d:=9

Zadatak 1: Napiši program koji će učitati ime bilo koje osobe i ispisati ga svako slovo u novom redu.

```
PROGRAM nizovi;
USES crt;
VAR b,d : integer;
      ime: STRING[15];
BEGIN
write ('Upiši bilo koje ime ');
readln (ime);
d:=length (ime);
FOR b:=1 TO d DO
      writeln (ime[b]);
repeat until keypressed;
END.
```

→ ***ispisuje slovo po slovo, tj. ono na mjestu brojača!***

Zadatak 2: Napiši program koji će bilo koje upisano ime ispisati naopačke i sve u jednom redu.

Spajanje 2 ili više niza znakova

Nizove znakova spajamo znakom +. Npr. ako imamo dvije nizovne varijable:

```
a := 'dobar ';
b := 'dan';
```

i ako je c := a+b *onda će* c := 'dobar dan'

Zadatak 3: Napiši program kojim će se upisati nečije ime pa prezime, to zatim spojiti u jednu varijablu ali tako da se ispiše prvo prezime pa ime (sve u jednom redu).

Zadatak 4: Napiši program koji će učitati neko ime i ispisati ga ovako:
Npr, ako učitamo MARKO, treba ispisati :

```
M
MA
MAR
MARK
MARKO
```

Funkcija **COPY**

Piše se i poziva ovako: **a:=copy(niz, pocetak, kraj);**

npr. dio:=**copy**(recenica, 1, 5);

- ova funkcija ***kopira niz znakova*** iz varijable **recenica** počevši od **1**. znaka pa do **5**. znaka i rezultat stavlja u varijablu **dio**.

Funkcija **POS**

Piše se i poziva ovako : **p:=poz (b, a);**

Napomena: Varijable **a** i **b** su string, a **p** je integer!

- **rezultat** te funkcije je broj tj. ***pozicija pojavljivanja*** podniza **b** u nizu **a**

Npr.
```
a:='trokut';
b:='kut';
p:=poz (b, a);
write ('pozicija je: ', p);
```
→ - podniz '**kut**' se pojavljuje u nizu '**trokut**' i to počevši od pozicije 4 u nizu a pa je vrijednost varijable **p** je **4**

1. Standardne procedure

VAŽNO: Standardne procedure ***se ponašaju kao obične naredbe***, pa ih i pišemo kao naredbe što znači da njihov rezultat ne možemo pridružiti nekoj drugoj varijabli!

1) DELETE

Piše se i poziva ovako: **delete (niz, poz, duljina);**

- ova procedura ***briše*** željeni broj znakova, počevši od željene pozicije u nizu, npr. ako imamo dio programa

```
a:='prabaka';
delete (a, 1,3);
write (a);
```
→ obrisat će 3 znaka u nizu **a** počevši od 1. pozicije

→ rezultat je niz **baka**

ili
```
a:='ABCDEFGH';
delete (a, 3, 4);
write (a);
```
→ ra~~čuna~~lo tj. počevši od 3. mjesta briše 4 sljedeća znaka

2) INSERT

Piše se i poziva ovako: **insert (umet, a, poz);**

- ova procedura ***umeće*** neki niz znakova označen kao **umet** u nizovnu varijablu npr. imena **a** počevši od njene pozicije **poz**

Npr.
```
a:='prabaka';
insert ('STARA', a, 4);
write (a);
```
} ispisat će se **praSTARAbaka**

3) STR

Piše se i poziva ovako: **str (x, niz);**

- ova procedura ***pretvara*** realan ili cijeli ***broj* X *u niz znakova***

Npr. Ako imamo neki cijeli broj, pa trebamo ispisati posebno njegove znamenke, najlakše je to napraviti tako da broj pretvorimo u niz znakova, pa da ovisno o duljini toga niza (broja znamenki) ispišemo svaku znamenku posebno.

```
PROGRAM znamenke;
VAR x, d: integer;
      br: string[10];
BEGIN
write ('Upisi broj');
readln (x);
str (x, br);
d:=length (br);
FOR i:=1 TO d DO
    writeln (br[i]);
readln;
END.
```

1. ako smo za x upisali broj 9632
2. varijabla **br** će biti **'9632'** - obratiti pažnju na polunavodnike koji označavaju niz znakova
3. **d** je duljina niza **br**
4. **for** petlja omogućuje da dođemo ***do svakog znaka u nizu***, te se oni ispisuju svaki u svom redu

4) VAL

Piše se i poziva ovako: **val (niz, X, n);**

- ovo je suprotna procedura od STR. Ona ***niz znakova pretvara u broj*** tj. zapisuje ga u brojčanu varijablu **X**. Varijabla **n** je cjelobrojna (integer) a potrebna je jer se u nju upisuje 0 (nula) ako je pretvorba uspješna ili neki drugi broj (pozicija greške) ako pretvorba nije bila uspješna.

Npr. Ako iz prethodnog primjera trebamo sve znamenke broja međusobno zbrojiti – ako je broj x:= 9632 zbroj njegovih znamenki je 20. Da bi to dobili moramo broj X pretvoriti u niz, pa onda doći do svake znamenke, pa znamenke pretvoriti u brojčanu vrijednost i zbrojiti ih. Program bi bio sljedeći:

```
PROGRAM zbroj_znamenki;
VAR x, d, zbroj,a, n : integer;
      br: string[10];
BEGIN
write ('Upisi broj');
readln (x);
str (x, br);
d:=length (br);
FOR i:=1 TO d DO
   BEGIN
    val (br[i], a, n);
    zbroj:=zbroj+a;
   END;
writeln ('Zbroj znamenki je: ', zbroj);
readln;
END.
```

1. ako smo za x upisali broj 9632
2. varijabla **br** će biti **'9632'** - obratiti pažnju na polunavodnike koji označavaju niz znakova
3. **d** je duljina niza **br**
4. **for** petlja omogućuje da dođemo do svakog znaka - znamenke u nizu
5. tu ***znamenku pretvorimo u broj*** procedurom **val**
6. znamenku zbrojimo

2. Programi sa stringovima

1) Prebrojati i ispisati riječi u rečenici koja se učitava preko tipkovnice.

```
PROGRAM riječi;
VAR ulaz : string[100];
        rijec: string[15];
        d,b,p: integer;
BEGIN
writeln('Upisi recenicu ');
readln(ulaz);
b:=1;
REPEAT
    d:=length(ulaz);
     p:=pos('  ', ulaz);
     if p<>0 then
                    begin
                        rijec:=copy(ulaz, 1, p-1);
                        writeln(rijec);
                        ulaz:=copy(ulaz, p+1, d);
                        b:=b+1;
                     end
                else writeln(ulaz);
UNTIL p=0;
writeln('Broj rijeci je: ',b);
readln;
END.
```

Objašnjenje: Riječi međusobo razdvajamo jednim razmakom, pa program traži razmake i ispisuje tekst prije razmaka, a ulaz skraćuje za onaj dio koji se ispisao. Postupak se ponavlja sve dok ima razmaka u ulaznom nizu znakova.

2) Zadana su 3 broja aritmetičkog niza **a,b,c** (brojevi se uvećavaju uvijek za isti korak **b-a** ili **c-b**). Treba izračunati koliko se puta javlja **2** kao znamenka u brojevima, počevši od prvog pa do stotog elementa aritmetičkog niza. *Pr. aritmetičkog niza*: 2, 5, 8, 11, 14, 17…

```
PROGRAM znamenka2:
VAR a,b,c,k,bb,d,i,j : integerM;
        broj : string;
BEGIN
write('Upisi brojeve aritmetickog niza a,b,c : ');
readln(a,b,c);
k:=b-a;
FOR i:=1 to 100 DO
     BEGIN
          str(a, broj);
          d:=length(broj);
          FOR j:=1 to d DO
                IF broj[j]='2' THEN bb:=bb+1;
          a:=a+k;
      END;
writeln('Znamenka 2 se pojavljuje ' , bb,  ' puta');
END.
```

broj **a** pretvaramo u niz i spremamo rezultat u varijablu **broj** da bi mogli analizirati znamenke toga broja

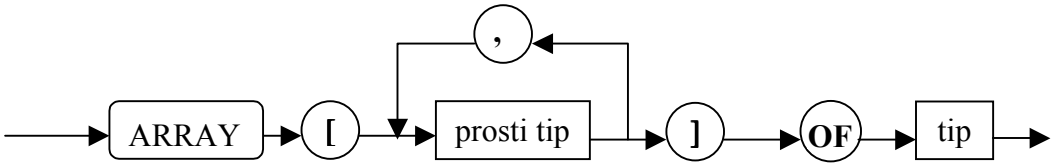
VI. **POLJA -** *ARRAY*

Prosti tip podataka je skup nedjeljivih elemenata, pa govorimo o pojedinačnim podacima i pojedinačnim varijablama.

Polazeći od prostih tipova podataka moguće je definirati složenije tipove podataka koje nazivamo **strukturalni tip podataka**. Njihova karakteristika je da se sastoje od skupova vrijednosti čija struktura ima određeni smisao. Takvi tipovi podataka potrebni su nam za rješavanje složenijih zadataka tj. svakodnevnih programerskih problema.
U Pascalu možemo koristiti sljedeće strukturalne tipove podataka:

1. **Polja ili niz (ARRAY)**
2. **Skup (SET)**
3. **Zapis ili slog (RECORD)**
4. **Datoteka (FILE)**

POLJE (**ARRAY**) je struktura podataka koja se sastoji od ***skupa elemenata*** (komponenata) ***istog tipa***. Tip tih elemenata može biti bilo koji prosti ili strukturalni tip podataka osim datoteke (FILE).



Zašto koristiti polje? Uzmimo primjer da moramo svakodnevno bilježiti temperature zraka u nekom gradu (za 1 mjesec) da bi izračunali najveću, najmanju i prosječnu temperaturu. Ako koristimo primitivne tipove podataka, trebali bi onoliko varijabli koliko ima dana u tom mjesecu. Međutim, znamo da su ***sve te varijable istoga tipa***. Umjesto da svaku varijablu označimo svojim imenom, možemo sve te varijable istoga tipa označiti jednim ***zajedničkim imenom*** čime koristimo varijablu tipa **polje**. Dakle, polje je cjelina tih elemenata, a do pojedinog elementa u polju dolazimo koristeći ime njegove varijable i poziciju tog elementa u polju. Tu poziciju zovemo **indeks**. Indeks nam ujedno govori o maksimalnom broju elemenata u polju. No, i sam ***broj indeksa*** nije ograničen, pa ovisno o tome govorimo o **jednodimenzionalnim** (*vektor*) - niz, **dvodimenzionalnim** (*matrica*) i višedimenzionalnim poljima. To moramo odrediti na početku programa.

JEDNODIMENZIONALNO POLJE je najjednostavniji oblik polja sa samo jednim indeksom, a matematički ga uspoređujemo sa vektorom. Takao polje možemo definiarti na sljedeći način:

TYPE ime_tip= **ARRAY[**v1..v2**] OF** T1;

gdje je: **v1..v2** - ***interval indeksa*** s time da je **v1** donja a **v2** gornja granica indeksa; kao indeks može se pojaviti bilo koji redni (ordinalni) tip podataka tj. svi prosti tipovi osim "real" (ponoviti s učenicima)

T1 - tip elemenata u polju

Deklariranje varijable ovoga tipa činimo na sljedeći način:

VAR ime_var : **ARRAY[v**1..v2**] OF** T1;

ili **VAR** ime_var : ime_tip;

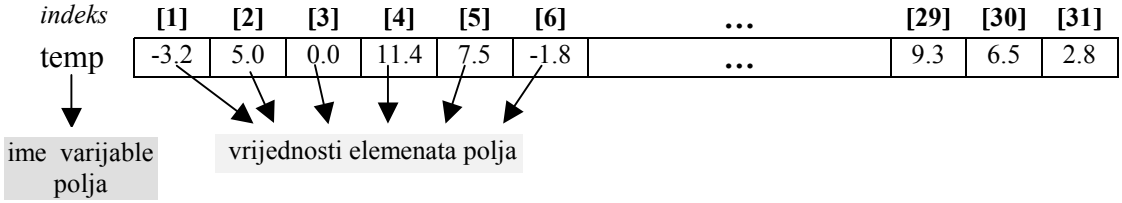
Npr. Ukoliko želimo riješiti prethodno spomenuti problem sa temperaturama, možemo deklarirati sljedeću varijablu:

VAR temp : **ARRAY** [1..31] **OF** real;

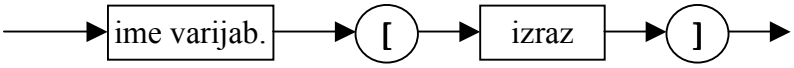
ili, kao indeks možemo koristiti i znakovni (char) tip podataka:

VAR slovo : **ARRAY** ['A'..'Z'] **OF** boolean;

Slikovito, varijablu **temp** koju smo prethodno deklarirali možemo prikazati ovako:



Do svakog elementa u polju možemo doći korištenjem imena varijable i pozicije - **indeksa** tog elementa u polju.



temp[1], temp[2], temp[3] … temp[i], temp[i+1], temp[i+j] …
slovo['A'], slovo['B'], … slovo[chr(90)], slovo[succ('C')] …

Takve varijable zovemo ***indeksirane varijable***, a njih koristimo kao i bilo koju drugu pojedinačnu varijablu. Evo nekoliko primjera:

1) Želimo li nekom elementu polja pridružiti određenu vrijednost, to ćemo učiniti ovako:
   temp[4] := 11.4 ; ili **slovo['G'] :=** false;
2) Želimo li šestom elementu polja **temp** učitati vrijednost preko tipkovnice
   Read (**temp[6]**) ; ili Write (abs(**temp[2]**));

Na isti način koristimo ostale naredbe, operacije i funkcije nad pojedinim elementima polja. Iako na početku bloka programa (deklaracijski dio) moramo odrediti najveću duljinu polja i njegovu dimenziju, u obradi ne moramo koristiti sve njegove elemente, nego samo određeni potrebni dio.

Vrijednosti elemenata polja u početku nisu definirane. Ukoliko želimo unijeti vrijednosti za sve ili većinu elemenata polja, to činimo unutar naredbe za ponavljanje. Isti slučaj je sa ispisivanjem vrijednosti pojedinih elemenata polja.

Pr. Treba upisati izmjerene dnevne temperature za mjesec siječanj i ispisati ih, po danima.

```
PROGRAM temperature;
USES crt;
VAR temp : ARRAY[1..31] OF real;
      i : integer;
BEGIN
writeln ('Upisite izmjerene temperature:');
{upis svih elemenata polja}
FOR i:=1 TO 31 DO
      readln (temp[i]);
clrscr;
{ispis svih elemenata polja}
FOR i:=1 TO 31 DO
      writeln ('Temperatura',  i,  '. dana bila je',  temp[i]:6:2);
readln;
END.
```

1. Programi s poljima

1. Za upisane dnevne temperature treba izračunati prosječnu temperaturu i ispisati koliko je puta upisana temperatura veća od prosječne. Unaprijed se ne zna broj dana za koje se upisuje temperatura, ali je najveći broj dana 31.

```
PROGRAM temperature;
VAR temp : ARRAY[1..31] OF real;;
      i, n, b : integer;
      sum, pros : real;
BEGIN
write ('Upisi broj dana – najvise 31: ');
readln (n);
writeln ('Upisi izmjerene temperature:');
FOR i:=1 TO n DO
      readln (temp[i]);
sum:=0;
FOR i:=1 TO n DO
      sum:=sum+ temp[i];
pros:=sum/n;
b:=0;
FOR i:=1 TO n DO
      IF temp[i] > pros THEN b:=b+1;
writeln ('Prosjecna temperatura iznosi:', pros:6:2);
writeln ('Broj temperatura vecih od prosjeka iznosi:', b);
readln;
END.
```

Dopunite ovaj zadatak tako da:

a) ispišete, za svaku upisanu temperaturu koliko odstupa od prosjeka

b) ispišete razliku prve i zadnje upisane temperature
c) pronađete i ispišete najmanju temperaturu i koliko puta se pojavljuje
d) ispišete temperature dana s neparnim indeksima, ali u redoslijedu obrnutom od upisivanja
e) ispitate jesu li uzastopno, u 2 dana izmjerene iste temperature i kada

c)
```
PROGRAM min_temp;
VAR temp : ARRAY[1..31] OF real;
        i, n, b : integer;
        min : real;
BEGIN
{… upis elemenata u polje …}

min:= temp[1];
FOR i:=2 TO n DO
        IF temp[i] < min THEN min:= temp[i];
b:=0;
FOR i:=1 TO n DO
        IF temp[i] = min THEN b:=b+1;
write ('Najmanja temperatura je', min:6:2);
writeln ('izmjerena je',  b,  ' puta');
readln;
END.
```

d)
```
FOR i:=n  DOWNTO 1  DO
        IF odd(i) THEN writeln (i, '. dana izmjerena temp. iznosi ',  temp[i]:6:2);
```

e)
```
PROGRAM iste_temp;
VAR temp : ARRAY[1..31] OF real;
        j, n, b: integer;
BEGIN
{upis elemenata niza …}

FOR j:=1 TO n-1 DO
   IF temp[j] = temp[j+1] THEN writeln ('Iste temperature izmjerene su', j, 'i', j+1, '. dana')
                          ELSE b:=b+1;
IF b=n-1 THEN writeln ('Nisu izmjerene iste uzastopne temperature');
readln;
END.
```

2. Sortiranje elemenata polja od manjega prema većemu.

```
PROGRAM sort;
VAR  br:= array[1..100] of real;
        i, j, n :  integer;
        x : real;
BEGIN
```

```
write (‘Koliko brojeva zelite upisati? ’);
REPEAT
readln (n);
UNTIL (n>0) and (n<101);
writeln (‘Upisite ’, n, ‘brojeva’);
FOR i:=1 TO n DO read (br[i]);
FOR i:=1 TO n-1 DO
     FOR j:=i+1 TO n DO
          IF br[i]>br[j] THEN
                         BEGIN
                            x:= br[j];
                            br[j]:= br[i];
                            br[i]:= br[j];
                         END;
writeln (‘Sortirani brojevi su: ’);
FOR i:=1 TO n DO write (br[i]:5);
readln;
END.
```

upis elemenata polja - for petlja ima samo ovu 1 naredbu

Uspoređuju se elemeti polja i to tako da se element u gornjoj petlji (**i**) usporedi sa svim sljedećim elementima u polju tj. doljnjoj petlji (**j**) i ako je prvi element veći od ovog drugog, oni zamijene mjesta

ispis elemenata polja - for petlja ima samo ovu 1 naredbu

3. Ispiši sve prim ili proste brojeve od 1 do N.

- to su oni brojevi koji su djeljivi samo sa 1 i sa samim sobom kao npr. 1, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31…

a) Algoritam "**Eratostenovo sito**" koristi **polje logičkih vrijednosti** s ***indeksima*** od **2 do maxint** (najveći integer broj). Početno se vrijednosti svim elementima polja postave na true. Zatim se sa FOR i WHILE petljom "križaju" tj. postavljaju na false svi elementi polja čiji su indeksi djeljivi s tom vrijednosti brojača petlje što znači da nisu prosti brojevi.

```
PROGRAM prosti_br;
VAR n : integer;
      i, j : 2 .. maxint;
      prosti: array[2 .. maxint]  of boolean;
BEGIN
write ('Upisi gornju granicu: ');
readln (n);
FOR i:=2 TO n DO prosti[i]:= true;
FOR j:=2 TO trunc(sqrt(n)) DO
      IF prosti[j] THEN
         BEGIN
         i:=2*j;
         WHILE i<=n  DO
                  BEGIN
                     prosti[i]:= false;
                     i:=i+j;
                  END;
         END;
write ('Prosti brojevi do zadane granice su: ');
FOR i:=2 TO n DO
```

označavaju se oni elementi polja čiji su indeksi djeljivi s brojem **i**

```
        IF prosti[i] THEN write prosti[i];
    readln;
    END.
```

npr. želimo li ispis svih prostih brojeva do 13 postupak je sljedeći:

	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]
prosti	true	true	true	true	true	true	true	true	true	true	true	true

koraci: **j:=2**; prosti[2]= true (da); i:=4;
i< n (da) pa se izvršava WHILE petlja ➔ prosti[4] postavlja na fasle, i:=4+2 tj. 6
6<13 (da) pa prosti[6] postavlja na false; i:=6+2 tj. 8
8<13 (da) pa prosti[8] postavlja na false; i:=8+2 tj. 10
10<13 (da) pa prosti[10] postavlja na fasle; i:=10+2 tj. 12
12<13 (da) pa prosti[12] postavlja na false; i:=12+2 tj. 14
14<13 (NE) pa se WHILE petlja više ne izvršava nego ide sljedeća naredba, tj. FOR petlja koja j:=3 i postupak se ponavlja
j:=3; prosti[3] je true (da) i:=2*3 tj. 6
6<13 (da) pa prosti[6] postavlja na false; i:=6+3 tj. 9
9<13 (da) pa prosti[9] postavlja na false; i:=9+3 tj. 12
12<13 (da) pa prosti[12] postavlja na false; i:=12+3 tj. 15
15<13 (NE) pa se WHILE petlja prekida
j:=4 …

	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]
prosti *za j:=2*	true	true	~~true~~ false	true	~~true~~ false	true	~~true~~ false	true	~~true~~ false	true	~~true~~ false	true

VII. POTPROGRAMI - PROCEDURE i FUNKCIJE

Potprogrami su složene naredbe koje grupiraju niz naredbi u 1 cjelinu i omogućuju da se jednom definirana cjelina (potprogram) može pozivati i izvršavati neograničeni broj puta.

Kada koristiti potprograme*?*

Ako imamo niz naredbi koje rade neki zadatak i to trebamo koristiti nekoliko puta unutar programa. Npr. nekoliko puta u programu moramo zamijeniti varijablama vrijednosti. Iako je taj dio programskog koda kratak, nepotrebno ga je ponavljati nekoliko puta. Dovoljno je definitati proceduru koja će to raditi jer je možemo onda pozvati onoliko puta koliko je to potrebno.
Primjena potprograma uvijek je vezana za dijelove programa koji se ponavljaju, pa ih izdvajamo u zasebne cjeline.

Potprograme ***definiramo u zaglavlju*** programa tj. odmah iza deklaracije varijabli.

***Kako definirati potprogram**?*

Razlikujemo **PROCEDURE** i **FUNCTION**, a iako su u principu slične, razlika je u tome što funkcija ima samo jednu izlaznu vrijednost tj. rezultat dok procedura može imati više izlaznih vrijednosti i što se rezultat funkcije mora pridružiti nekoj varijabli! Deklariraju se isto, kao u sljedećem primjeru.

1. pr. Imamo 3 broja koja trebamo poredati po veličini, od manjega prema većemu. Koristit ćemo proceduru za zamjenu brojeva.

```
PROGRAM poredaj;
VAR : a,b,c : integer;

PROCEDURE zamijeni (VAR i,j : integer);
    VAR x: integer;
    BEGIN
       x:=j;
       j:=i;
       i:=x;
    END;

{pocetak glavnog dijela programa}
BEGIN
write ('Upisi 3 broja: ');
readln (a, b, c);
IF a>b THEN zamijeni(a,b);
IF a>c THEN zamijeni(a,c);
IF b>c THEN zamijeni(b,c);
write ('Poredani brojevi su: ' , a, b, c);
readln;
END.
```

zaglavlje PROCEDURE sadrži ***ime*** s kojom će se pozivati iz glavnog programa. a u zagradama ako piše **VAR** te varijable su ***ulazno-izlazne*** , a varijable bez VAR su samo ulazne
- ispod je deklarirana varijabla X koja je *lokalna* tj. vrijedi samo u toj proceduri

- **poziv procedure** sa konkretnim vrijednostima varijabli
- te varijable se "***prenose***" u proceduru tako da prva varijabla postaje **i**, a druga **j**, a na kraju procedura vraća vrijednosti i, j ovim varijablama koje su tu napisane
- ***tipovi varijabli*** kod poziva procedure i tip varijabli u samoj proceduri moraju biti podudarni

2. pr. Poznato je da Pascal nema gotovu ***funkciju za potenciranje*** (već samo za izračunavanje kvadrata broja) ali je možemo sami napraviti. To ćemo učiniti kada u istom programu trebamo više puta izračunati X^Y.

```
PROGRAM potenciranje;
VAR x, y, r : integer;

FUNCTION  pot(a,b: integer): integer;
    VAR c, i: integer;
    BEGIN
        c:=1;
        for i:=1 to b do
            c:=c*a
        pot:=c;
    END;
```

Zaglavlje funkcije sadrži njeno ime i u zagradi ulazne varijable, a nakon toga moramo odrediti kojeg tipa je njen rezultat

U ovoj funkciji broj **a** množimo **b** puta sam sa sobom i rezultat stavljamo u varijablu **c**
- vrijednost funkcije je jednaka vaijabli **c**

```
BEGIN
writeln('Upisi brojeve X i Y da bi se izracunalo X^Y');
readln(x,y);
r:=pot(x,y);
writeln('Rezultat je ', r);
readln;
END.
```

poziv funkcije sa vrijednostima varijabli **x** i **y** a rezultat toga spremit će se u varijablu **r**

Funkcije i procedure možemo definirati i pozivati **rekurzivno** što znači da one pozivaju same sebe. Najbolji primjer za to je izračunavanje faktorijela. Npr. faktorijel od 6 ili
6! = 6*5*4*3*2*1 ili općenito: **n! = n*(n-1)*(n-2)*…*3*2*1** i po definiciji **0! = 1**

```
PROGRAM faktorijel;
VAR a: integer;
      r: longint;

FUNCTION fakt(n: integer) : longint;
    BEGIN
        IF n>0 THEN fakt:=n*fakt(n-1) ELSE fakt:=1;
    END;

BEGIN
write('Upisi broj za izracunavanje faktorijela ');
readln(a);
r:=fakt(a);
writeln(a, '! = ' , r);
readln;
END.
```

-vidimo da se s desne strane pridruživanja opet poziva ista funkcija, ali s vrijednošću **n -**1 a to prestaje tek kada je n=0
- na kraju se sve te vrijednosti pomnože da bi dobili rezultat funkcije