Slogovi

U realnim problemima je često potrebno podatke organizovati u određene celine. Tako su, na primer, svi podaci o studentu (ime, prezime, adresa, broj indeksa...) organizovani u jednu celinu koja čini njegov lični dosije. Službe fakulteta mogu da manipulišu podacima studentskim dosijeima bez opasnosti od mešanja podataka o različitim studentima, do čega bi moglo da dođe ukoliko bi se, recimo, imena studenata čuvala u jednoj knjizi, adrese u drugoj, a ocene u trećoj. Pristup pojedinačnim podacima je olakšan samim tim što se svi podaci o jednom studentu nalaze na istom mestu. U sledećoj tabeli dat je primer jednog studentskog dosijea:

Ime i prezime	Petar Petrović
Adresa	Nikole Pašića 13/31, 34000 Kragujevac
Broj indeksa	17/08
Status	Apsolvent
Prosečna ocena	8.43

Ovakva organizacija podataka u programskom jeziku Pascal omogućena je korišćenjem slogovnog tipa podataka. **Slog** je skup više promenljivih koje su grupisane pod zajedničkim imenom radi lakše manipulacije. Slogovi pomažu u organizovanju složenih podataka tako što omogućavaju grupisanje međusobno povezanih promenljivih na takav način da se one posmatraju kao celina, a ne kao pojedinačni entiteti. Promenljive unutar sloga nazivamo **polja** ili **promenljive članice**.

Slogovi se u programskom jeziku Pascal mogu definisati korišćenjem rezervisane reči **record** iza koje sledi spisak promenljivih članica. Definicija sloga se završava rezervisanom rečju **end**, nakon čega sledi znak **;**.

Sintaksa

```
type <naziv_sloga>=record
                     <polje_1>:<tip_polja_1>;
                     <polje_2>:<tip_polja_2>;
                              ...
                     <polje_n>:<tip_polja_n>;
                   end;
```

Primer

```
type status_studenta=(aktivan,neaktivan);
     student=record
               ime:string[20];
               adresa:string[30];
               indeks:string[6];
               status:status_studenta;
               prosek:real;
             end;

var  student1, student2: student;
```

U prethodnom primeru definisan je slogovni tip *student* koji sadrži 5 polja: *ime*, *adresa*, *indeks*, *status* i *prosek*. Svako od polja je promenljiva koja ima svoj tip, ali sve one zajedno pripadaju jednom tipu podataka – slogu *student*. Polja mogu biti proizvoljnog tipa, bez obzira da li su oni standardni ili nestandardni. Tako je polje *status* deklarisano kao promenljiva tipa *status_studenta*, koji je prethodno definisan. Promenljive *student1* i *student2* su zatim deklarisane kao promenljive tipa *student*, na isti način kao što se deklarišu promenljive prostih tipova podataka.

Pored prostih tipova podataka, promenljive članice sloga mogu biti i složenog tipa. Tako, članice slogova mogu biti nizovi, matrice, ili neki drugi slogovi. U sledećem primeru ćemo adresu studenta predstaviti kao

složeni podatak koji sadrži ulicu, broj i mesto, a takođe ćemo dodati i podatke o položenim ispitima i ostvarenim ocenama.

```
const   MAX_PREDMETA=100;
type  status_studenta=(aktivan,neaktivan);
      adresa= record
                 ulica:string[20];
                 broj:integer;
                 opstina:string[15];
              end;
      ispit=  record
                 predmet:string[20];
                 ocena:integer;
              end;
      student=record
                 ime:string[20];
                 prebivaliste:adresa;
                 indeks:string[6];
                 status:status_studenta;
                 prosek:real;
                 ispiti:array[1..MAX_PREDMETA] of ispit;
              end;
```

U prethodnom primeru definisan je slog *adresa* koja sadrži tri polja (*ulica, broj* i *opstina*), a zatim je unutar sloga *student* deklarisano polje *prebivaliste* koje je tipa *adresa*. Dakle, unutar sloga *student* postoji polje *prebivaliste*, koje je takođe slog. Takođe, promenljive članice sloga mogu biti i nizovnog tipa, kao što je to slučaj sa poljem *ispiti* unutar sloga *student*. Na primeru ovog polja možemo videti da slogovi mogu biti i elementi nizova, pa je tako polje *ispiti* ustvari niz čiji su elementi slogovnog tipa *ispit*.

Svakom polju sloga se može direktno pristupiti navođenjem naziva promenljive slogovnog tipa iza koje sledi tačka, a zatim naziv željenog polja. Na primer, podacima sloga *student* se može pristupiti radi čitanja ili upisivanja vrednosti na sledeći način:

```
student1.ime:="Petar Petrovic";
student1.prebivaliste.ulica:="Nikole Pasica";
student1.ispiti[2].ocena:=9;
readln(student1.prosek);
student2.prosek:=student1.prosek;
writeln(student2.prosek);
```

Pored pristupa pojedinačnim poljima radi čitanja i dodele vrednosti, moguće je korišćenje i čitavih slogova u operaciji dodele. Tako se, na primer, može napisati:

```
student2:=student1;
```

čime se praktično vrednosti svih polja promenljive *student1* dodeljuju odgovarajućim poljima promenljive *student2*.

Polja određenog tipa se mogu koristiti u svim izrazima tog tipa. Na primer, polja realnog tipa se mogu koristiti u realnim izrazima:

```
student1.prosek:=student2.prosek+0.5;
```

Kao što smo mogli videti na primerima naredbi **readln** i **writeln**, slogovi se mogu koristiti i kao parametri funkcija i procedura, na potpuno isti način kao i svi ostali tipovi podataka. Međutim, slogovi ne mogu biti rezultat funkcija.

Primer 1

Napisati program koji za dva vektora u prostoru računa njihov zbir, skalarni i vektorski proizvod. Da se podsetimo, zbir vektora $\vec{a}$ i $\vec{b}$ je vektor $\vec{c}$, čije su komponente

$c_x = a_x + b_x$

$c_y = a_y + b_y$

$c_z = a_z + b_z$

Skalarni proizvod vektora $\vec{a}$ i $\vec{b}$ je skalar s, pri čemu je

$s = a_x b_x + a_y b_y + a_z b_z$

dok je vektorski proizvod ova dva vektora vektor $\vec{c}$ čije su komponente

$c_x = a_y b_z - b_y a_z$

$c_y = a_z b_x - b_z a_x$

$c_z = a_x b_y - b_y a_x$

```
program Vektori;
type vektor=record
              x,y,z:real;
            end;
var  a,b,c:vektor;

   procedure zbir(a,b:vektor; var c:vektor);
   begin
      c.x:=a.x+b.x;
      c.y:=a.y+b.y;
      c.z:=a.z+b.z;
   end;

   function skalarni(a,b:vektor):real;
   begin
      skalarni:=a.x*b.x+a.y*b.y+a.z*b.z;
   end;

   procedure vektorski(a,b:vektor; var c:vektor);
   begin
      c.x:=a.y*b.z-b.y*a.z;
      c.y:=a.z*b.x-b.z*a.x;
      c.z:=a.x*b.y-b.y*a.x;
   end;

   procedure stampaj_vektor(a:vektor);
   begin
      writeln('{',a.x,',',a.y,',',a.z,'}');
   end;

begin
   writeln('Unesite vektor a:');
   readln(a.x,a.y,a.z);

   writeln('Unesite vektor b:');
   readln(b.x,b.y,b.z);

   zbir(a,b,c);
   writeln('Zbir vektora a i b je:');
   stampaj_vektor(c);

   writeln('Skalarni proizvod vektora a i b je:');
   writeln(skalarni(a,b));
```

```
   vektorski(a,b,c);
   writeln(Vektorski proizvod vektora a i b je:');
   stampaj_vektor(c);
end.
```

Primer 2

Napisati program koji učitava podatke o fudbalskim ekipama, broj bodova koje su osvojili u toku sezone, kao i ukupan broj žutih kartona koji su dobili igrači svakog tima, a zatim na osnovu broja bodova štampa tabelu i određuje tim koji će biti nagrađen za "fer plej" (tim koji ima najmanje žutih kartona).

```
program Fudbal;
const MAX_EKIPA=100;
type ekipa=record
             naziv:string[20];
             bodovi:integer;
             kartoni:integer;
           end;
     niz_ekipa=array[1..MAX_EKIPA] of ekipa;
var  lista:niz_ekipa;
     n,i:integer;

   procedure ucitaj_ekipu(var e:ekipa);
   begin
     writeln('Unesite naziv ekipe:');
     readln(e.naziv);
     writeln('Unesite broj ostvarenih bodova:');
     readln(e.bodovi);
     writeln('Unesite broj zutih kartona:');
     readln(e.kartoni);
   end;

   procedure sortiraj(var lista:niz_ekipa; n:integer)
   var  i,j:integer;
        pom:ekipa;
   begin
     for i:=1 to n-1 do
       for j:=i+1 to n do
         if lista[i].bodovi<lista[j].bodovi then
           begin
             pom:=lista[i];
             lista[i]:=lista[j];
             lista[j]:=pom;
           end;
   end;

   function ferplej(lista:niz_ekipa; n:integer):integer;
   var  i:integer;
        min_kartona,min_indeks:integer;
   begin
     min_kartona:=lista[1].kartoni;
     min_indeks:=1;

     for i:=2 to n do
       if lista[i].kartoni<min_kartona then
         begin
           min_kartona:=lista[i].kartoni;
           min_indeks:=i;
         end;
```

```
      ferplej:=min_indeks;
   end;

   procedure stampaj_tabelu(lista:niz_ekipa; n:integer);
   var  i:integer;
   begin
      writeln('Naziv               Bodovi    Kartoni');
      for i:=1 to n do
         writeln(lista[i].naziv:20,lista[i].bodovi:10,lista[i].kartoni:10);
   end;

begin
   writeln('Unesite broj ekipa:');
   readln(n);

   for i:=1 to n do
      ucitaj_ekipu(lista[i]);

   sortiraj(lista,n);

   stampaj_tabelu(lista,n);

   i:=ferplej(lista,n);

   writeln('Ekipa nagradjena za fer plej je ',lista[i].naziv);
end.
```

Naredba WITH

U prethodni primerima smo imali prilike da vidimo kako se pristupa promenljivama članicama nekog sloga. Međutim, često ponavljanje naziva slogovne promenljive prilikom pristupanja njenim članicama može biti veoma zamorno. Tako, na primer, ukoliko želimo da svim poljima sloga *student* dodelimo odgovarajuće vrednosti, morali bismo pri svakoj dodeli da ponavljamo naziv slogovne promenljive:

```
student1.ime:="Ivan Ivanovic";
student1.adresa:="Kralja Aleksandra 46";
readln(student1.prosek);
student1.status:=aktivan;
```

Slično bismo radili i prilikom očitavanja vrednosti ovih promenljivih:

```
writeln(student1.ime);
writeln(student1.adresa);
student1.prosek:= student2.prosek+0.5;
```

Programski jezik Pascal obezbeđuje pojednostavljenje u pisanju ovakvih segmenata programa u vidu komande **with**. Ukoliko se iza komande **with** navede određena slogovna promenljiva, u bloku koji sledi iza naredbe nije neophodno pisati naziv ove promenljive prilikom pristupanja njenim članicama.

Sintaksa

```
with <slogovna_promenljiva> do
   begin
      ...
   end;
```

Sada se prethodni segmenti koda mogu napisati kao

```
with student1 do
   begin
      ime:="Ivan Ivanovic";
      adresa:="Kralja Aleksandra 46";
      readln(prosek);
      status:=aktivan;
   end;
```

odnosno

```
with student1 do
   begin
      writeln(ime);
      writeln(adresa);
      prosek:= student2.prosek+0.5;
   end;
```

Naredba **with** može imati i ugnježdenu strukturu. Na primer:

```
with a do
   with b do
      begin
         ...
      end;
```

Ukoliko promenljive *a* i *b* sadrže polja istog naziva, a to polje se javlja unutar bloka naredbi, onda se ovo polje vezuje za poslednju navedenu slogovnu promenljivu unutar naredbe **with**. Takođe, ukoliko van naredbe **with** postoji promenljiva istog naziva kao i polje sloga navedenog unutar naredbe **with**, pominjanje tog naziva u bloku naredbe **with** će se odnositi isključivo na polje tog sloga, a ne na promenljivu van naredbe **with**.

Prethodna struktura se može kraće zapisati kao

```
with a, b do
   begin
      ...
   end;
```