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 većini programskih jezika 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 C češće nazivaju strukturama i mogu se definisati korišćenjem rezervisane reči **struct** iza koje sledi naziv strukture i rezervisan znak **{**. Nakon toga ide spisak promenljivih članica. Definicija strukture se završava rezervisanim znakom **}**, nakon čega sledi znak **;**.

Moguće je strukturu definisati i kao tip korišćenjem rezervisane reči **typedef**. Razlika je u tome što se naziv tipa zadaje posle rezervisanog znaka **}**, nakon čega sledi znak **;**.

Sintaksa

```
struct <naziv_strukture>
{
   <tip_polja_1> <polje_1>;
   <tip_polja_2> <polje_2>;
                 ...
   <tip_polja_n> <polje_n>;
};

typedef struct
{
   <tip_polja_1> <polje_1>;
   <tip_polja_2> <polje_2>;
                 ...
   <tip_polja_n> <polje_n>;
} <naziv_tipa>;
```

RADNA VERZIJA

Primer

```
enum status_studenta {aktivan, neaktivan};

struct student
{
   char ime[20];
   char adresa[30];
   char indeks[6];
   enum status_studenta status;
   float prosek;
};

struct student student1, student2;
```

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 – strukturi *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 strukture mogu biti i složenog tipa. Tako, članice struktura mogu biti nizovi, matrice, ili neke druge strukture. 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.

```
enum status_studenta { aktivan, neaktivan};

struct adresa
{
   char ulica[20];
   int broj;
   char opstina[15];
};

struct ispit
{
   char predmet[20];
   int ocena;
};

struct student
{
   char ime[20];
   struct adresa prebivaliste;
   char indeks[6];
   enum status_studenta status;
   float prosek;
   struct ispit ispiti[100];
};
```

U prethodnom primeru definisana je struktura *adresa* koja sadrži tri polja (*ulica, broj* i *opstina*), a zatim je unutar strukture *student* deklarisano polje *prebivaliste* koje je tipa *adresa*. Dakle, unutar strukture *student* postoji polje *prebivaliste*, koje je takođe struktura. Takođe, promenljive članice strukture mogu biti i nizovnog tipa, kao što je to slučaj sa poljem *ispiti*

RADNA VERZIJA

unutar strukture *student*. Na primeru ovog polja možemo videti da strukture mogu biti i elementi nizova, pa je tako polje *ispiti* ustvari niz čiji su elementi slogovnog tipa *ispit*.

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

```
student1.prosek = 9.5;
student1.ispiti[2].ocena = 9;
student1.prebivaliste.broj = 62;
scanf("%s%f", student1.ime, &student1.prosek);
student2.prosek = student1.prosek;
printf("%4.2f\n", student2.prosek);
```

Pored pristupa pojedinačnim poljima radi čitanja i dodele vrednosti, moguće je korišćenje i čitavih struktura 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 **scanf** i **printf**, strukture se mogu koristiti i kao parametri funkcija, na potpuno isti način kao i svi ostali tipovi podataka.

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

$$c_x = a_y b_z - b_y a_z$$
$$c_y = a_z b_x - b_z a_x$$
$$c_z = a_x b_y - b_y a_x$$

```
#include <stdio.h>

typedef struct
{
   float x, y, z;
} vektor;
```

RADNA VERZIJA

```
vektor zbir(vektor a, vektor b)
{
   vektor c;

   c.x = a.x+b.x;
   c.y = a.y+b.y;
   c.z = a.z+b.z;

   return c;
}

float skalarni(vektor a, vektor b)
{
   return a.x*b.x+a.y*b.y+a.z*b.z;
}

vektor vektorski(vektor a, vektor b)
{
   vektor c;

   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;

   return c;
}

void stampaj_vektor(vektor a)
{
   printf("%10.2f%10.2f%10.2f\n", a.x, a.y, a.z);
}

main()
{
   vektor a, b, c;

   printf("Unesite vektor a:\n");
   scanf("%f%f%f", &a.x, &a.y, &a.z);

   printf("Unesite vektor b:\n");
   scanf("%f%f%f", &b.x, &b.y, &b.z);

   c = zbir(a, b);
   printf("Zbir vektora a i b je:\n");
   stampaj_vektor(c);

   printf("Skalarni proizvod vektora a i b je: %10.2f\n", skalarni(a,b));

   c = vektorski(a, b);
   printf("Vektorski proizvod vektora a i b je:");
   stampaj_vektor(c);
}
```

RADNA VERZIJA

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).

```
#include <stdio.h>

typedef struct
{
    char naziv[20];
    int bodovi;
    int kartoni;
} ekipa;

typedef ekipa niz_ekipa[100];

ekipa ucitaj_ekipu()
{
    ekipa e;
    printf("Unesite naziv ekipe:\n");
    scanf("%s", e.naziv);
    printf("Unesite broj ostvarenih bodova:\n");
    scanf("%d", &e.bodovi);
    printf("Unesite broj zutih kartona:\n");
    scanf("%d", &e.kartoni);

    return e;
}

void sortiraj(niz_ekipa lista, int n)
{
    int i, j;
    ekipa pom;

    for(i = 0; i < n-1; i++)
        for(j = i+1; j < n; j++)
            if(lista[i].bodovi < lista[j].bodovi)
            {
                pom = lista[i];
                lista[i] = lista[j];
                lista[j] = pom;
            }
}

int ferplej(niz_ekipa lista, int n)
{
    int i;
    int min_kartona, min_indeks;

    min_kartona = lista[0].kartoni;
    min_indeks = 0;

    for(i = 1; i < n; i++)
        if(lista[i].kartoni < min_kartona)
        {
            min_kartona = lista[i].kartoni;
            min_indeks = i;
        }
```

RADNA VERZIJA

```
    return min_indeks;
}

void stampaj_tabelu(niz_ekipa lista, int n)
{
    int i;
    printf("Naziv               Bodovi    Kartoni\n");
    for(i = 0; i < n; i++)
     printf("%20s%10d%10d\n", lista[i].naziv, lista[i].bodovi, lista[i].kartoni);
}

main()
{
    niz_ekipa lista;
    int n, i;

    printf("Unesite broj ekipa:\n");
    scanf("%d", &n);

    for(i = 0; i < n; i++)
        lista[i] = ucitaj_ekipu();

    sortiraj(lista, n);

    stampaj_tabelu(lista, n);

    i = ferplej(lista, n);

    printf("Ekupa nagradjena za fer plej je %s\n", lista[i].naziv);
}
```

RADNA VERZIJA