Nizovi

Standardni i nestandardni prosti tipovi podataka (celobrojni, realni, logički, znakovni, nabrojivi) mogu biti sasvim dovoljni pri rešavanju manjih i jednostavnijih problema. Međutim, u slučaju rešavanja složenijih problema često je potrebno baratati velikim brojem podataka. Veliki broj podataka bi zahtevao postojanje isto tolikog broja promenljivih u koje bi se mogle smestiti njihove vrednosti.

Zamislimo program koji vrši obračun plata u nekom preduzeću. Ovakav program bi morao da ima bar onoliko realnih promenljivih koliko ima zaposlenih. Sigurno je da u firmama koje zapošljavaju hiljade radnika ovakav pristup ne bi bio moguć, jer bi zahtevao ogroman broj promenljivih u programu, što nikako nije praktično:

```
float pera,mika,laza,...,zoran1,zoran2,...,pera1254;
```

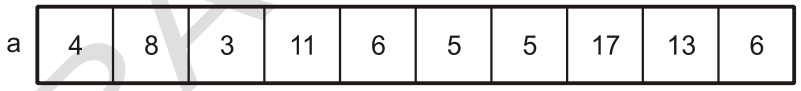
Međutim, znatno veći problem predstavlja činjenica da bi za izračunavanje plate više radnika bilo neophodno ponavljati istu operaciju nad različitim promenljivama (svaki radnik ima svoje promenljive), što onemogućava bilo kakvu automatizaciju čitavog procesa:

```
pera = pera_sati * cena_rada;
mika = mika_sati * cena_rada;
laza = laza_sati * cena_rada;
     ...
zoran1 = zoran1_sati * cena_rada;
zoran2 = zoran2_sati * cena_rada;
     ...
pera1254 = pera1254_sati * cena_rada;
```

Takođe, u trenutku pravljenja programa bilo bi neophodno znati tačan broj radnika u preduzeću, što u praksi nije moguće iz razloga što je broj radnika promenljiv od preduzeća do preduzeća, a takođe je promenljiv i u okviru jednog preduzeća tokom vremena.

Rešenje ovakvih problema u programskom jeziku C je moguće korišćenjem **nizovnog tipa podataka**, odnosno **nizova**. Niz je ograničen uređen skup promenljivih istog tipa, koje se nazivaju **elementima** ili **članovima** niza. Svakom elementu niza pridružen je indeks, uz pomoć koga je moguće pristupiti odgovarajućem elementu.

Na Slici ### je prikazan jednodimenzionalni niz celih brojeva *a*, koji ima 10 elemenata. U nastavku će biti reči isključivo o jednodimenzionalnim nizovima, koje ćemo skraćeno nazivati samo **nizovi**. Kada budemo govorili o višedimenzionalnim nizovima, to će biti posebno naglašeno.



Slika ### Jednodimenzionalni niz celih brojeva

Da bi bilo moguće definisati niz, neophodno je znati interval u kome se kreću indeksi niza, kao i tip elemenata niza. Nizovni tip se u programskom jeziku C definiše tako što se navede tip elemenata pa naziv niza iza kojeg u uglastim zagradama sledi dužina niza (broj elemenata niza). U programskom jeziku C indeksi niza uvek počinju od 0, pa je indeks poslednjeg elementa u nizu za jedan manji od broja elemenata. Ovakva definicija nizovnog tipa se može koristiti za definisanje novog tipa u okviru naredbe **typedef** ili za deklarisanje nizovnih promenljivih.

Sintaksa

```
typedef <tip_elemenata> <novi_tip>[<dužina_niza>];
```

ili

```
<tip_elemenata> <naziv_promenljive>[<dužina_niza>];
```

Primer

```
typedef int niz[10];
niz a;
```

Da bismo pročitali ili promenili vrednost nekog elementa, možemo mu pristupiti navođenjem indeksa elementa u uglastim zagradama iza promenljive nizovnog tipa:

```
p=a[0];      {"p" dobija vrednost prvog elementa niza "a"}
a[6]=12;       {sedmi element niza "a" dobija vrednost 12}
printf("%d", a[4]); {stampa se vrednost petog elementa niza "a"}
scanf("%d", &a[1]); {ucitava se vrednost i smesta u drugi clan niza "a"}
```

NAPOMENA: Indeks prvog elementa je 0.

Na Slici ### je prikazan način indeksiranja niza realnih brojeva *x*, pri čemu su indeksi celobrojnog tipa na intervalu od 0 do 9:

x	4.2	8.8	3.14	11.0	6.02	-5.0	2.73	17.1	-13.1	6.84
	x[0]	x[1]	x[2]	x[3]	x[4]	x[5]	x[6]	x[7]	x[8]	x[9]

Slika ### Indeksiranje celim brojevima na intervalu od 0 do 9

Za pristupanje određenom elementu niza, pored konstanti se mogu koristiti i promenljive i izrazi, ali pri tome treba voditi računa da njihov tip mora biti celobrojan i unutar definisanog intervala niza:

```
p=2;              {celobrojna promenljiva dobija vrednost 2}
a[p+3]=7;         {ispravno, jer je indeks 5 unutar opsega 0..9}
a[p+8]=3;         {neispravno, jer indeks 10 izlazi iz opsega 0..9}
a[1+a[p+3]]=a[2]; {ispravno}
```

NAPOMENA: Jednodimenzionalni nizovi se vrlo često nazivaju i vektorima iz razloga što se određena veličina, odnosno vektor položaja u *n*-dimenzionalnom prostoru, može predstaviti nizom dužine *n*, pri čemu svaki element niza predstavlja odgovarajuću koordinatu vektora položaja. Na primer, vektor položaja tačke u trodimenzionalnom prostoru se može predstaviti realnim nizom dužine 3, pri čemu bi prvi element označavao *x*, drugi *y*, a treći *z* koordinatu vektora. Zahvaljujući ovome, nad nizovima se mogu vršiti vektorske operacije kao što su sabiranje vektora, vektorski proizvod, itd.

Primer 1

Napisati program u kome korisnik unosi broj radnih sati s_i koje je ostvario svaki od n radnika nekog preduzeća, cenu radnog sata *c*, a zatim se izračunava plata svakog radnika po obrascu $p_i = s_i \cdot c$.

```
#include <stdio.h>

main()
{
   int i, n;
   float s[100], p[100];
   float c;
```

```
   printf("Unesite broj radnika:");
   scanf("%d", &n);

   //Unos broja radnih sati za svakog radnika
   for(int i = 0; i < n; i++)
   {
      printf("Unesite broj radnih sati %d. radnika:\n", i+1);
      scanf("%f", &s[i]);
   }

   printf("Unesite cenu radnog sata:\n");
   scanf("%f", &c);

   //Racunanje i stampanje plata
   for(int i = 0; i < n; i++)
   {
      p[i] = s[i] * c;
      printf("Plata %d. radnika je %.2f dinara.\n", i+1, p[i]);
   }
}
```

Primer 2

Napisati program koji učitava dva realna niza istih dužina, a zatim pozivanjem potprograma izračunava njihov zbir i skalarni proizvod. Da se podsetimo, zbir dva niza (vektora) *a* i *b* dužine *n* je takođe niz dužine *n*, pri čemu su članovi novog niza *s* definisani kao:

$$s_i = a_i + b_i \qquad i = 1..n$$

Skalarni proizvod dva niza (vektora) *a* i *b* dužine *n* je skalar *p* definisan kao:

$$p = a_1 \cdot b_1 + a_2 \cdot b_2 + ... + a_n \cdot b_n = \sum_{i=1}^{n} a_i b_i$$

```
#include <stdio.h>

typedef float niz[100];

void zbir(niz a, niz b, niz c, int n)
{
   int i;

   for(i = 0; i < n; i++)
      c[i] = a[i] + b[i];
}

float skalarni(niz a, niz b, int n)
{
   int i;
   float suma = 0.0;

   for(i = 0; i < n; i++)
      suma += a[i] * b[i];

   return suma;
}

main()
{
   int n, i;
   niz a, b, s;
```

```
   float p;

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

   printf("Unesite elemente prvog niza:\n");
   for(i = 0; i < n; i++)
      scanf("%f", &a[i]);

   printf("Unesite elemente drugog niza:\n");
   for(i = 0; i < n; i++)
      scanf("%f", &b[i]);

   zbir(a, b, s, n);
   p = skalarni(a, b, n);

   printf("Zbir vektora je:\n");
   for(i = 0; i < n; i++)
      printf("%10.4f", s[i]);
   printf("\n\n");

   printf("Skalarni proizvod vektora je: %10.4f", p);
}
```

Pretraživanje nizova

Pretraživanje je proces koji za cilj ima pronalaženje elementa niza, koji zadovoljava unapred definisani kriterijum. Najčešće je to traženje elementa koji sadrži određeni podatak, koji nazivamo **ključ**. U zavisnosti od tipa strukture podataka u kojoj se vrši pretraživanje, algoritmi mogu biti manje ili više kompleksni.

Sekvencijalno pretraživanje

Sekvencijalno pretraživanje je sigurno najjednostavniji algoritam za pretraživanje nizova. Ovaj algoritam se naziva još i **linearno pretraživanje** iz razloga što se traženje određenog elementa vrši tako što se ispituje jedan po jedan element niza sve dok se ne nađe element koji zadovoljava zadate kriterijume ili se ne dođe do kraja niza.

Primer 1

Posmatrajmo niz celih brojeva *a* dužine *n*. Napisati program koji nalazi poziciju prvog pojavljivanja zadatog celog broja *t* u nizu *a*. Ukoliko niz ne sadrži traženi element, funkcija treba da vrati -1.

```
#include <stdio.h>

typedef int niz[100];

int trazi(niz a, int n, int t)
{
   int i;
   int indeks = -1;

   for(i = 0; i < n; i++)
      if(a[i] == t)
      {
         indeks = i;
         break;
```

```
     }

   return indeks;
}

main()
{
   int i, broj, trazeni, indeks;
   niz a;

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

   printf("Unesite elemente niza:\n");
   for(i = 0; i < broj; i++)
   {
      printf("Unesite %d. element niza:\n", i+1);
      scanf("%d", &a[i]);
   }

   printf("Unesite element koji trazite:\n");
   scanf("%d", &trazeni);

   indeks = trazi(a, broj, trazeni);

   if(indeks > - 1)
      printf("Trazeni element se nalazi na poziciji %d.\n", indeks);
   else
      printf("Trazeni element ne postoji.\n");
}
```

Primer 2

Napisati program koji u nizu realnih brojeva *a* dužine *n* određuje sumu svih elemenata koji zadovoljavaju kriterijum $|a_i| > 3$.

```
#include <stdio.h>

typedef float niz[100];

float suma(niz a, int n)
{
   int i;
   float s;

   s = 0;
   for(i = 0; i < n; i++)
      if(a[i] < -3 || a[i] > 3)
         s += a[i];

   return s;
}

main()
{
   int i, broj;
   niz a;

   printf("Unesite broj elemenata niza:\n");
```

```
   scanf("%d", &broj);

   for(i = 0; i < broj; i++)
   {
      printf("Unesite %d. element niza:\n", i+1);
      scanf("%f", &a[i]);
   }

   printf("Suma elemenata koji zadovoljavaju kriterijum je %f\n", suma(a, broj));
}
```

Sortiranje nizova

Sortiranje je proces kojim se niz prevodi u stanje u kome su elementi raspoređeni u rastućem ili opadajućem redosledu. Za niz kažemo da je rastući ukoliko je svaki element niza veći od svog prethodnika, a manji od svog sledbenika. Suprotno, niz je opadajući ako je svaki element manji od svog prethodnika, a veći od svog sledbenika. Sortirane nizove nazivamo još i uređenim nizovima.

Sortiranje izborom uzastopnih minimuma

Postoje različite vrste algoritama za sortiranje, koji se bitno razlikuju po principu rada i brzini izvršavanja. Jedan od najjednostavnijih algoritama za sortiranje niza u rastući poredak je svakako **izbor uzastopnih minimuma**, poznat i kao **selection sort**. Ovaj algoritam uređuje niz u rastući tako što traži najmanji element i smešta ga na početak niza. Zatim se za ostatak niza ponavlja isti postupak, sve dok se ne dođe do podniza koji ima samo jedan element. Na sličan način vrši se i sortiranje niza u opadajući poredak, sa jedinom razlikom što se u tom slučaju vrši izbor uzastopnih maksimuma.

Primer 1

```
#include <stdio.h>

typedef int niz[100];

void rastuci(niz a, int n)
{
   int i, j;
   int pom;

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

void opadajuci(niz a, int n)
{
   int i, j;
   int pom;

   for(i = 0; i < n - 1; i++)
      for(j = i+1; j < n; j++)
         if(a[j] > a[i])
```

```
        {
           pom = a[i];
           a[i] = a[j];
           a[j] = pom;
        }
}

void stampaj(niz a, int n)
{
   int i;

   for(i = 0; i < n; i++)
      printf("%5d", a[i]);
   printf("\n");
}

main()
{
   int i, broj;
   niz a;

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

   for(i = 0; i < broj; i++)
   {
      printf("Unesite %d. element niza:\n", i+1);
      scanf("%d", &a[i]);
   }

   rastuci(a, broj);

   printf("Niz u rastucem poretku je:\n");
   stampaj(a, broj);

   opadajuci(a, broj);

   printf("Niz u rastucem poretku je:\n");
   stampaj(a, broj);
}
```


Primer 2

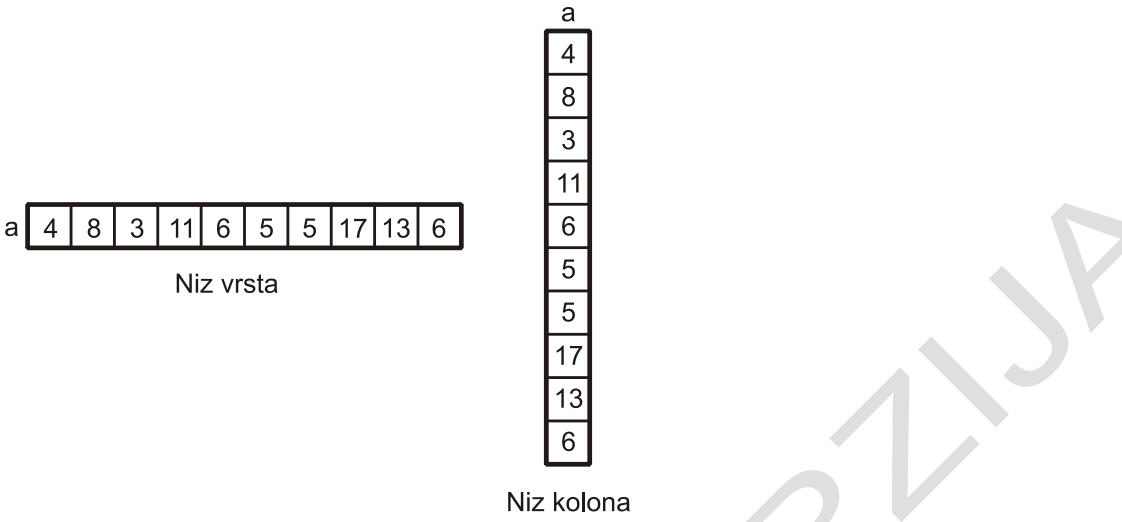
Napisati program koji na osnovu broja osvojenih poena formira rang listu studenata koji su polagali ispit. Svaki student ima broj indeksa (ceo broj) i broj poena (float). Napisati i funkciju koja će za zadati broj indeksa ispisati broj poena koje je taj student osvojio.

```
#include <stdio.h>

typedef int niz_indeksa[100];
typedef float niz_poena[100];

void sortiraj(niz_indeksa indeksi, niz_poena poeni, int n)
{
   int i, j;
   int pom_ind;
   float pom_poen;
```

```
   for(i = 0; i < n-1; i++)
      for(j = i+1; j < n; j++)
         if(poeni[j] > poeni[i])
         {
            pom_ind = indeksi[i]; indeksi[i] = indeksi[j]; indeksi[j] = pom_ind;
            pom_poen = poeni[i]; poeni[i] = poeni[j]; poeni[j] = pom_poen;
         }
}

void stampaj(niz_indeksa indeksi, niz_poena poeni, int n)
{
   int i;
   printf("R.br.     Indeks    Poeni\n");

   for(i = 0; i < n; i++)
      printf("%10d%10d%10.2f\n", i+1, indeksi[i], poeni[i]);
}

void student(int indeks_studenta, niz_indeksa indeksi, niz_poena poeni, int n)
{
   int i;
   printf("'R.br.     Indeks    Poeni\n");

   for(i = 0; i < n; i++)
      if(indeksi[i] == indeks_studenta)
      {
         printf("%10d%10d%10f\n", i+1, indeksi[i], poeni[i]);
         break;
      }
}

main()
{
   niz_indeksa indeksi;
   niz_poena poeni;
   int broj, i, ind;

   printf("Unesite broj studenata koji su polagali ispit:\n");
   scanf("%d", &broj);

   for(i = 0; i < broj; i++)
   {
      printf("Unesite indeks %d. studenta:\n", i+1);
      scanf("%d", &indeksi[i]);
      printf("Unesite broj poena koje je osvojio %d. student:\n", i+1);
      scanf("%f", &poeni[i]);
   }

   sortiraj(indeksi, poeni, broj);

   stampaj(indeksi, poeni, broj);

   printf("Unesite broj indeksa studenta:\n");
   scanf("%d", &ind);

   student(ind, indeksi, poeni, broj);
}
```

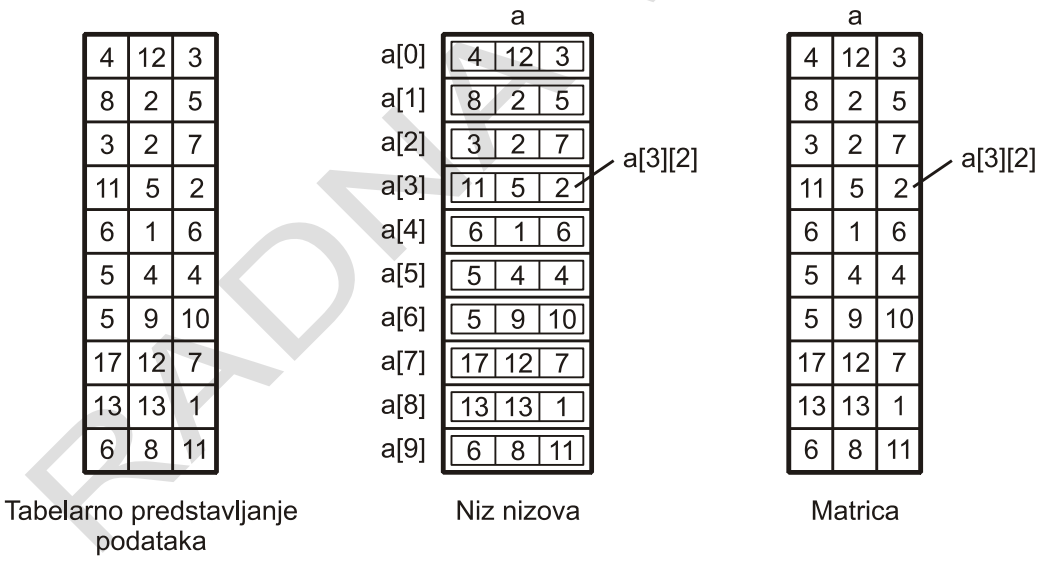
Matrice

Do sada smo imali prilike da se upoznamo sa jednodimenzionalnim nizovima čiji su elementi skalarne veličine (celi brojevi, realni brojevi, znakovi,...). Ovakvi nizovi se mogu šematski prikazati kao horizontalna ili vertikalna lista podataka, kao što je prikazano na Slici ###.



Slika ### Šematski prikaz niza u vidu horizontalne (niz vrsta) ili vertikalne (niz kolona) liste podataka

Međutim, u velikom broju realnih problema je pogodnije podatke predstaviti u tabelarnoj formi, kao što je prikazano na Slici ###.



Slika ### Tabelarno predstavljanje podataka

U programskom jeziku C predstavljanje podataka datih u tabelarnoj formi se može obaviti na više načina. Jedan od njih i ređe korišćeni način je posmatranje tabelarnih podataka kao niza nizova kao što je prikazano na Slici ###. Tabela visine *m* i širine *n*, može se predstaviti kao niz dužine *m* čiji su elementi nizovi dužine *n*. Pogledajmo kako bi izgledao C kod kojim bi se deklarisala tabela *a* visine 10 i širine 3 elementa, pri čemu su elementi celi brojevi:

```
typedef int vrsta[3];
```

```
typedef vrsta tabela[10];
tabela a;
```

U prethodnom kodu definisan je tip podatka *vrsta* koji predstavlja niz od 3 elementa tipa *int*. Zatim je definisan tip podatka *tabela* koji predstavlja niz od 10 elemenata tipa *vrsta*. u poslednjem redu deklarisana je promenljiva *a* tipa tabela. Na ovaj način promenljiva *a* će predstavljati niz od deset celobrojnih nizova dužine 3. Pojedinačnoj vrsti se može pristupiti korišćenjem zapisa

```
a[i]
```

gde je *i* redni broj vrste. Elementu u *i*-toj vrsti i *j*-toj koloni se može pristupiti korišćenjem zapisa

```
a[i][j]
```

S obzirom da je ovakav način pisanja prilično komplikovan, u programskom jeziku C omogućeno je direktno definisanje višedimenzionalnih nizova. Dvodimenzionalne nizove najčešće nazivamo **matrice**. Matrica celih brojeva *a*, dimenzija 10x*3*, pri čemu je 10 broj vrsta, a 3 broj kolona, može se skraćeno deklarisati na sledeći način

```
typedef int matrica[10][3];
matrica a;
```

ili direktno kao

```
int a[10][3];
```

Sada se elementu u *i*-toj vrsti i *j*-toj koloni može pristupiti korišćenjem zapisa

```
a[i][j]
```

S obzirom da se u većini programskih jezika elementi matrice u memoriji pakuju vrsta po vrsta, uobičajeno je da prvi indeks predstavlja vrstu, a drugi kolonu. Vrste u matricama često nazivamo i redovima.

Na sličan način mogu se definisati i višedimenzionalni nizovi. Tako, na primer, možemo definisati i trodimenzionalni niz koji bi predstavljao niz matrica. Sledeći kod pokazuje kako bi se definisao niz od 5 matrica dimenzija 10x3:

```
int a[5][10][3];
```

a njegovim elementima bi se pristupalo korišćenjem zapisa

```
a[t][i][j]
```

pri čemu je *t* redni broj matrice, *i* redni broj vrste, a *j* redni broj kolone u kojoj se element nalazi.

Primer 1

Napisati program koji učitava matricu realnih brojeva *a* dimenzija *m* x *n* i računa sumu svih elemenata matrice.

```
#include <stdio.h>

main()
{
   float a[100][100];
   int m, n, i, j;
   float suma;

   printf("Unesite broj vrsta i kolona:\n");
   scanf("%d%d", &m, &n);
```

```
   printf("Unesite elemente u matricnom obliku:\n");
   for(i = 0; i < m; i++)
      for(j = 0; j < n; j++)
         scanf("%f", &a[i][j]);

   suma = 0.0;
   for(i = 0; i < m; i++)
      for(j = 0; j < n; j++)
         suma += a[i][j];

   printf("Suma elemenata matrice je %f\n", suma);
}
```

Primer 2

Napisati program koji pozivanjem odgovarajućeg potprograma vrši transponovanje kvadratne matrice celih brojeva *a* dimenzija *n* x *n*. Da se podsetimo, transponovanje matrice podrazumeva zamenu vrsta i kolona, tj. zamenu mesta elementima a[i][j] i a[j][i], za svako *i* i *j*.

```
#include <stdio.h>

typedef float matrica[100][100];

void transponuj(matrica a, int n)
{
   int i, j;
   float pom;

   for(i = 0; i < n; i++)
      for(j = 0; j < i; j++)
      {
         pom = a[i][j];
         a[i][j] = a[j][i];
         a[j][i] = pom;
      }
}

void stampaj(matrica a, int n)
{
   int i, j;

   for(i = 0; i < n; i++)
   {
      for(j = 0; j < n; j++)
         printf("%10.2f", a[i][j]);
      printf("\n");
   }
}

main()
{
   matrica a;
   int n, i, j;

   printf("Unesite velicinu matrice:\n");
   scanf("%d", &n);

   printf("Unesite elemente u matricnom obliku:\n");
```

```
   for(i = 0; i < n; i++)
      for(j = 0; j < n; j++)
         scanf("%f", &a[i][j]);

   transponuj(a, n);

   stampaj(a, n);
}
```