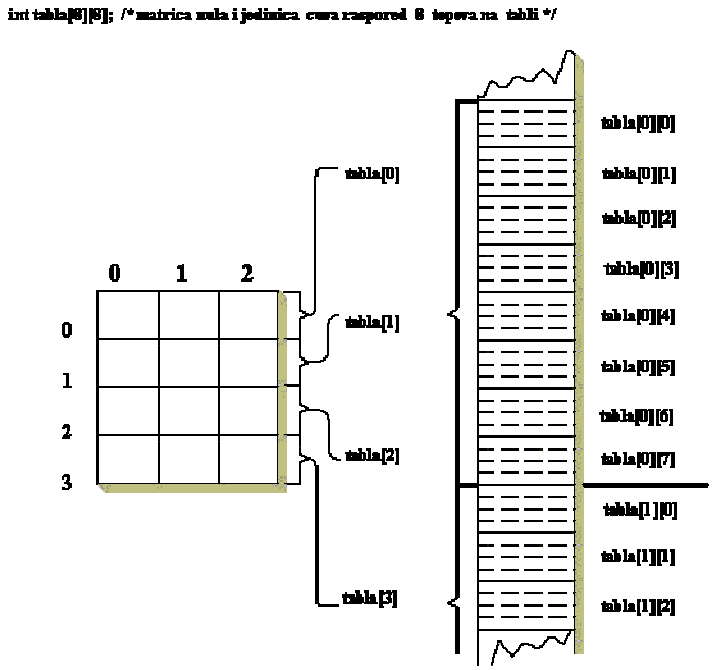
Višedimenzioni nizovi i matrice, pokazivači na funkcije, dinamička alokacija memorije

1. Napisati program koji sa standardnog ulaza učitava raspored 8 topova na šahovskoj tabli. Raspored se učitava u formi 8 linija sa po 8 brojeva po liniji. Ako na datom polu nema topa, učitava se 0, a inače 1. Program mora da ispita validnost unetog rasporeda (da li su učitani brojevi ili 0 ili 1, da li ima 8 topova) i ispita da li se u datom rasporedu dva topa tuku.




```
#include <stdio.h>
#include <stdlib.h>

main()
{ int tabla[8][8];  /*matrica nula i jedinica  cuva raspored  8  topova na  tabli */
  int suma;         /* pamti se suma reda/kolone kao test osobina iz zahteva zadatka */
   int i,j;          /*brojaci u petljama */

/*ucitavanje podataka o rasporedu topova  uz sumiranje broja topova i u slucaju greske
  stampa se odgovarajuca poruka i okoncava programa */
suma=0;
 for (i=0; i<8; i++ )
  for (j=0; j<8; j++)
    {  scanf("%d", &tabla[i][j]);

       /*test korektnosti ulaza */
       if (tabla[i][j] !=0 && tabla[i][j] !=1 )
          { printf ("\nNekorektni  ulazni podaci o rasporedu topova\n");
             exit(EXIT_FAILURE);
          }
       suma=suma + tabla[i][j];
    }
```

```
/*greska je ako je broj topova na tabli razlicit od 8 */
if (suma !=8)
  { printf ("\nNekorektni  ulazni podaci o broju topova\n");
              exit(EXIT_FAILURE);
    }

/*proveravanje da li se dva topa nalaze u istom redu , tj. da li je suma
   clanova nekog reda  veca od 1 */
for(i=0;i<8;i++)
  {
      suma=0;
      for (j=0;j<8;j++)  suma+=tabla[i][j];
      if (suma >1 )
       { printf ("\nTopovi se tuku\n");
              exit(EXIT_SUCCESS);
        }
    }


/*proveravanje da li se dva topa nalaze u istoj koloni , tj. da li je suma
   clanova neke kolone    veca od 1 */
for(j=0;j<8;j++)
  {
      suma=0;
      for (i=0;i<8;i++)  suma+=tabla[i][j];
      if (suma >1 )
       { printf ("\nTopovi se tuku\n");
              exit(EXIT_SUCCESS);
        }
    }


/*inace se topovi ne tuku */
printf ("\nTopovi se ne tuku\n");
exit(EXIT_SUCCESS);
}
```

Matrice

Već smo se upoznali sa jednodimenzionalnim nizovima kod kojih smo svakom elementu niza mogli pristupiti navodjenjem imena niza i jedne indeksne vrednosti. Ali, postoji mnogo problema gde bi se sa podacima jednostavnije operisalo ako bi se podaci predstavili tabelom ili dvodimenzionim nizom.

U svakodnevnom životu često vidjamo podatke koji su predstavljeni tabelom. Na primer, šahovska tabla je primer tabele sa 8 vrsta (redova) i 8 kolona (stubaca). Polja šahovske table se mogu deklarisati na sledeći način:

```
int tabla[8][8];
```

pri čemu prva dimenzija označava redove, a druga kolone dvodimenzionalne tabele.
Pri deklarisanju višedimenzionalne tabele (u programskim jezicima C i C++) se svaka dimenzija piše u zasebnom paru uglastih zagrada.

Svaki element matrice tabla[i][j] se opisuje pomoću dva indeksa i, j. Prvi indeks
označava vrstu, a drugi kolonu u kojoj se nalazi element tabla[i][j].

Na primer, ako želimo da pamtimo koliko koševa je ekipa 2 dala ekipi 5 na jednom košarkaškom turniru, ima smisla da napravimo dvodimenzionu tabelu
Turnir
i zapisaćemo koliko je koševa ekipa 2 dala ekipi 5 u elementu turnir[2][5]

Slično, broj koševa koje je ekipa 5 dala ekipi 2
zapisaćemo u elementu turnir[2][5]
Elementu dvodimenzionog niza pristupa se navodjenjem indeksa za vrste i indeksa za kolone.

U matematici se dvodimenziona tabela naziva matrica, te ćemo i mi koristiti povremeno taj termin. Pri opisivanju tj. zadavanju matrice
koriste se prirodni brojevi N i M koji definišu dimenziju matrice tj. kažemo da NxM matrica ima N redova i M kolona.
Ako N=M, matrica se naziva kvadratna.

Inicijalizacija dvodimenzionalne tabele može da se vrši na sledeći način:

```
int matrica[3][4] = {
 { 35, 46, 32, 40 }
 { 34, 38, 31, 33 }
 { 39, 37, 41, 36 }
 };
```

Drugi način inicijalizacije:

```
int matrica[3][4] = {35,46,32,40,34,38,31,33,39,37,41,36};
```

Kako indeksiranje nizova u C/C++ počinje od 0, vrste se u matrici indeksiraju u gornjem primeru od 0 do 2, a kolone se u primeru indeksiraju od 0 do 3.

Elemente koje ne navedemo automatski se postavljaju na 0. Na sledeći način možemo sve elemente tabele postaviti na 0:

```
 int tabele[2][2][2]={{0}};
```

Ako definišemo tabelu

```
 int tabele[2][2][2]={ {{1,2}, {3,4}} , { {5,6}, {7,8} } };
```

onda je istinito:

```
  tabele[0][0][0] == 1  && tabele[0][0][1] == 2  && tabele[0][1][0] == 3 &&
  tabele[0][1][1] == 4  && tabele[1][0][0] == 5  && tabele[1][0][1] == 6 &&
  tabele[1][1][0] == 7  && tabele[1][1][1] == 8
```

Dvodimenzioni niz se uvek može zameniti jednodimenzionim nizom. Ali to često zahteva dodatno (kodiranje.

Elementi niza takođe mogu biti nizovi. Na primer, konačan skup šahovskih tabli može se predstaviti modelom trodimenzionog niza.

Napomenimo da ako se pri inicijalizaciji višedimenzionalnih nizova, unutar vitičastih zagrada navede manje inicijalizacionih vrednosti
nego što je elemenata u vrsti, preostalim elementima se dodeljuje nula vrednost. Na primer, deklaracijom

```
  int tabela[3][4] = {
 { 35, 46},
 { 34, 38},
 { 39, 37}
 };
```

se inicijalizuju datim vrednostima prva dva elementa u vrsti, a poslednja dva elementa svake vrste se inicijalizuju nulom.

Ako se pri inicijalizaciji izostave unutrašnje zagrade i ostave samo spoljašnje zagrade, mogu se javiti dva slučaja. Prvi slučaj nastupa ako je
broj inicijalizacionih vrednosti jednak broju elemenata matrice. Tada će se vrste redom popuniti datim vrednostima. Ako je broj
inicijalizacionih vrednosti manji od broja elemenata dvodimenzionalnog niza, tada se matrica popunjava redom po vrstama dok ima inicijalizacionih
vrednosti. Potom se preostalim elementima dodeljuju nule. Na primer, deklaracijom

```
   int tabela[3][4] =
{ 35, 46, 34, 38, 39, 37};
```

se inicijalizuju datim vrednostima prva vrsta i prva dva elementa druge vrste, a ostali elementi se inicijalizuju nulom.

Primer upotrebe dvodimenzionog niza

```c++
#include <iostream>

using namespace std;

int main()
{
   int i,j;
   int tabela[3][3];  // tabela dimenzije 3x3
   for (i = 0; i < 3; i++)
      for (j = 0; j < 3; j++)
         tabela[i][j]=i*3+j;
   for (i = 0; i < 3; i++)
   {
      for (j = 0; j < 3; j++)
         cout <<tabela[i][j]<<" ";
      cout<<endl;
   }
   return 0;

}
```

ispis
0 1 2
3 4 5
6 7 8

Vidimo da dvodimenzioni niz se deklariše vrlo slično običnoj promenljivoj.
Tako ako hoćemo da deklarišemo 3x3 tabelu, to bismo uradili sledećom komandom:

```   int tabela[3][3];```

Ova komanda definiše u memoriji niz od 9 promenljivih pod nazivom "tabela".
Promenljive koje čine ovaj niz su numerisane od 0 do 8. Obratite pažnju, numerisanje uvek počinje od 0 i ide do proizvoda dimenzija minus 1.
```

Kako pristupamo ovim promenljivim? Tako što posle imena tabele u uglastim zagradama prosledimo kom elementu želimo da pristupimo.
Tako na primer, ako želimo da pristupimo prvom elementu gore levo u tabeli (koji ima indeks 0 u vrsti i indeks 0 u koloni),
to radimo tako što napišemo tabela[0][0].

Kada pristupimo elementu matrice možemo sa njim raditi sve što možemo i sa običnom promenljivom:
koristiti ga u izrazima, postavljati mu vrednost, itd.

Pogledajmo jedan primer, gde ćemo naznačiti u komentarima koje je stanje u memoriji nakon svake komande:

```c++
#include <iostream>

using namespace std;

int main()
{
	int tabela[3][3];
	// tabela[0][0], tabela[0][1], tabela[0][2]: nisu definisane vrednosti u 0. vrsti
	// tabela[1][0], tabela[1][1], tabela[1][2]: nisu definisane vrednosti u 1. vrsti
	// tabela[2][0], tabela[2][1], tabela[2][2]: nisu definisane vrednosti u 2. vrsti

	tabela[0][0] = 2;
	// tabele[0][0]: 2
	// nisu definisane vrednosti: tabela[0][1], tabela[0][2], tabela[1][0], tabela[1][1], tabela[1][2]
	// nisu definisane vrednosti: tabela[2][0], tabela[2][1], tabela[2][2]

	tabela[0][2] = 4;
	// tabela[0][0]: 2
	// tabela[0][1]: nije definisano
	// tabela[0][2]: 4

	tabela[0][1] = tabela[0][0] + tabela[0][2];
	// tabela[0][0]: 2
	// tabela[0][1]: 6
	// tabela[0][2]: 4

	return 0;
}
```

Kada hoćemo da pristupimo elementu tabele, ne moramo obavezno napisati samo broj u uglastim zagradama.
Možemo u njih staviti bilo koji izraz
koji kada se izračuna ima celobrojnu vrednost. Pogledajmo sledeći primer:

```c++
#include <iostream>

using namespace std;

int main()
{
	int indeks = 0;

	int tabela[3][3];
	// tabela[0][0], tabela[0][1], tabela[0][2]: nisu definisane vrednosti u 0. vrsti
```

```
        // tabela[1][0], tabela[1][1], tabela[1][2]: nisu definisane vrednosti u 1. vrsti
        // tabela[2][0], tabela[2][1], tabela[2][2]: nisu definisane vrednosti u 2. vrsti

        tabela[indeks][indeks] = 1;
        // tabela[0][0]: 1
        // nisu definisane vrednosti: tabela[0][1], tabela[0][2], tabela[1][0], tabela[1][1], tabela[1][2]
        // nisu definisane vrednosti: tabela[2][0], tabela[2][1], tabela[2][2]

        tabela[indeks + 1][indeks + 1] = 2;
        // tabela[0][0]: 1
        // tabela[1][1]: 2
        // ostale vrednosti u tabeli nisu definisane

        tabela[indeks + 2][indeks + 2] = 5;
        // tabela[0][0]: 1
        // tabela[1][1]: 2
        // tabela[2][2]: 5
        // ostale vrednosti u tabeli nisu definisane

}
```

Pri učitavanju matrice koja ima n vrsta i m kolona, sa tastature se redom učitavaju
najpre elementi prve vrste, potom druge,..., dok se ne učitaju elementi n-te vrste.

Na primer

```c++
#include <iostream>

using namespace std;

int main()
{
        int tabela[10][10];
   int i,j,n,m;

        cout << "Unesite dimenzije matrice" <<endl;
        cin >>n>>m;
        cout <<"Unesite matricu"<<endl;

   for(i=0;i<n;i++)
      for(j=0;j<m;j++)
         cin >> tabela[i][j];

        return 0;
}```

Uradimo još jedan primer da bi se uhodali u radu sa matricama. Stranica dnevnika sa završnim ocenama učenika na kraju školske godine može
može se predstaviti matricom.
U svakoj vrsti te matrice nalaze se sve završne ocene jednog učenika. U svakoj koloni su sve završne ocene iz nekog školskog predmeta.
Recimo da se unosi broj vrsta i broj kolona te matrice , a zatim i elementi matrice tj. ocene za sve učenike iz svih predmeta..
Ono što hoćemo da uradimo je da ispišemo indeks školskog predmeta koji ima najmanju prosečnu ocenu i da ispišemo taj prosek.
```

Pogledajmo kako bismo to uradili:

```c++
#include <iostream>

using namespace std;

int main()
{
   int tabela[10][10];
   int i,j,n,m;

   cout << "Unesite dimenzije matrice" <<endl;
   cin >>n>>m;
   cout <<"Unesite matricu"<<endl;

   for(i=0; i<n; i++)
      for(j=0; j<m; j++)
         cin >> tabela[i][j];

   /*lazno pretpostavimo da trazeni prosek je 0.0 i da je u pitanju predmet iz 0. kolona*/
   float minprosek=0.0;
   int minj=0;

   /* racunamo prosek elemenata 1. kolone*/
   for(i=0; i<n; i++) minprosek=minprosek+tabela[i][0];
   minprosek=minprosek/n;

   //za ostale kolone trazimo prosek na slican nacin
   //potom se prosek tekuce kolone uporedjuje sa najmanjim prosekom
   for(j=1; j<m; j++)
   {
      float tek=0.0;
      for(i=0; i<n; i++) tek=tek+tabela[i][j];
      tek=tek/n;
      if (tek<minprosek)
      {
         minprosek=tek;
         minj=j;
      }
   }

   cout << endl<< "Najmanji prosek ima " << minj << ". predmet" << endl;
   cout << "Prosek tog predmeta je " <<minprosek<<endl;

   return 0;
}
```

2. /*pokazivaci na f-je */

Napisati C program koji ispisuje na standardnom izlazu tri HTML tabele. Svaka tabela ima dve kolone. Prva kolona u sve tri tabele su brojevi od 1..n. U prvoj tabeli druga kolona su kvadrati brojeva od 1..n, a u drugoj tabeli druga kolona su prigušene oscilacije brojeva od 1..n, u trećoj tabeli druga kolona su kubovi brojeva od 1..n. Broj n se unosi sa standardnog ulaza.

```
#include <stdio.h>
#include <math.h>
/*f-ja koja formulom po argumentu tipa double opisuje prigusene oscilacije */
double priguseneOscila(int );
/*f-ja koja racuna kvadrat argumenta */
double kvadrat(int);
/*f-ja koja racuna kub argumenta */
double kub(int);
/*f-ja koja stampa HTML tabelu sa dve kolone */
void stampati( double (*ptrFunk) (int), int n)
{  int i; //brojac u ciklusu

   /*zaglavlje tabele*/
printf("\n<TABLE><TR><TH>I kolona</TH><TH>II kolona</TH></TR>\n");

for(i=1;i<=n;i++) printf("\n<TR><TD>%d</TD><TD>%lf</TD></TR>", i, (*ptrFunk)(i));
printf("</TABLE>");
}
main( )
{  int n;
   scanf("%d",&n); printf("<HTML>\n<HEAD></HEAD><BODY>\n");
//ispis 1. tabele
   stampati(priguseneOscila,n);
//ispis 2. tabele
   stampati(kvadrat,n);
//ispis 3. tabele
   stampati(kub,n);

printf("\n</BODY></HTML>\n");
   return 0;
}
double priguseneOscila(int x)
{ return exp(- 0.1 * x) * sin(x); }
double kvadrat(int x)
{ return (x)*(x);
}
double kub(int x)
{ return (x)*(x)*(x);
}
```

3. Napisati C program koji iz datoteke ulaz.txt ucitava dimenizju niza, a potom i niz celih brojeva (ciji broj nije unapred poznat), a na standardni izlaz ispisuje dimenziju podniza koji se sastoji samo od pozitivnih brojeva. Zadatak realizovati upotrebom dinamickog niza.

```
#include <stdio.h>
#include <stdlib.h>
#define GRESKA_DODELA  0 /*status prve poruke o gresci*/
#define GRESKA_PROMENA 1 /*status II poruke o gresci */
#define GRESKA_DATOTEKA 2 /*status III poruke o gresci */

char *poruke[]={  "\nNesupesna dodela memorije\n",
"\nNeuspesna promena dinamicki dodeljene memorije\n",
   "\nGreska pri otvaranju datoteke ulaz.txt\n"};

int sazima_niz(int a[], int n)
{  int i;  //brojac
```

```
   int *ptr1, *ptr2; /*pokazivac ptr1 pratice adrese svih elemenata polaznog niza,
                pokazivac ptr2 ce se koristiti za formiranje novog niza*/

ptr1=ptr2=a;

    for(i=0; i<n; i++)
          { if(*ptr1>0)    /*ako je tekuci clan niza pozitivan */
             { *ptr2=*ptr1;
               ptr2++;          /*ulazi u sastav novog niza*/
             }
       ptr1++; //prelazi se na poziciju sledeceg clana polaznog niza
          }

          return (ptr2-a); /*dobija se broj clanova novog niza */
}

void ucitavanje(int a[], FILE *f); //ucitavanje niza iz datoteke
void greska(int status); /*ispis poruka o gresci */
void stampaNiz(int a[], int n)
{ int i;
  printf("\nNovi niz\n");
  for(i=0; i<n; i++)  printf("%d ", a[i]);
  printf("\n\n");
}


int main( )
{  FILE *f;
    int *a, n; //niz iz datoteke i broj clanova
    int n2; //dimenizija sazetog niza


          f=fopen("ulaz.txt", "r");
          if (f==NULL) greska(GRESKA_DATOTEKA);
          fscanf(f, "%d", &n);

          a=(int *)malloc(n* sizeof(int));
    if (a==NULL)  greska(GRESKA_DODELA);
    ucitavanje(a,f);
          n2=sazima_niz(a,n);

          //ako se promenila dimenzija niza n, onda se
          //vrsi (re)alokacija za niz manje dimenzije n2
                    if (n2<n)
                              if( (a=(int *)realloc(a, n2*sizeof(int))) ==NULL) greska(GRESKA_PROMENA);

          stampaNiz(a,n2);

          //oslobadja se memorija za sazeti niz
          free(a);

          fclose(f);

          return 0;
}
```

```
void ucitavanje(int a[], FILE *f)
{
        int i=0;
        /*citanje brojeve iz datoteke, fscanf vraca broj uspesnih
        ucitavanja ili EOF pri nailasku na kraj datoteke */
        while (fscanf(f, "%d", &a[i++]))
          if (feof(f)) break; /*prekida se citanje nailaskom na EOF */
}

void greska(int status)
{
          puts(poruke[status]);
          exit(1);
}
```

4. ***Napisati C program koji će sabrati dva broja koji se zadaju kao dva parametra komandne linije***

```
/*u ovom zadatku  broj u dekadnom zapisu se u funkciji sabiranja transformise u nisku cifara u kojoj se najpre
pojavljuju cifre manje tezine (sto je suprotan redosled od onoga koji je unet komandnom linijom).
Izabran je redosled cifara koji je vise odgovarao prilikom racunanja. */

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define znak(cifra)  ((cifra) +'0')
#define  cifra(x)     ((x)-'0')
void saberi(char *a1, int n, char *a2, int m, char *s, int *ns);   /*a1+a2->s */
/*Kako u ovom zadatku koristimo nisku cifara , u potprogramima nisu bili nuzni n,m, ali ih prenosimo da bi kod bio
jasniji i da bi se mogao primeniti i ako se radi sa npr. (int *) , umesto (char *) */
char* reverse( char s[] ) ;
int main(int argc, char *argv[])
{
   int n=strlen(argv[1]);  /*br.cifra prvog "broja" */
   int m=strlen(argv[2]);   /*br.cifara drugog "broja" */
   int nzbir;              /*br cifara zbira */
   char *zbir;              /*zbir oba broja */

/*inicijalizacije */
nzbir= (n>m ? n : m) +1;  /*nzbir=max(n,m)+1 */
if(argc !=3)
{ fprintf(stderr,"\nNekorektan broj argumenata komandne linije u pozivu programa\n");
  exit(1);
}
 /*sabiranje */
zbir=(char *)malloc(nzbir*sizeof(char) );
saberi( reverse(argv[1]) ,n, reverse(argv[2]),m,zbir,&nzbir);
printf("\n%s+%s=  %s\n\n", reverse(argv[1]),reverse(argv[2]), reverse(zbir));
free(zbir);
return 0;
}
void saberi(char *a1, int n, char *a2, int m, char *s, int *ns)
{
   int prenos;  /*prenos u susedni razred cifara, a koji je nastao
                    pri sabiranju cifri oba broja tekuceg razreda */
   int c;      /*rezultat sabiranja tekuce cifre prvog broja i tekuce cifre drugog broja*/
   int i;      /*brojac u petlji */
```

```
   prenos=0;
    /*sabiranje velikih brojeva se odvija implementacijom algoritma naucenom u mladjim razredima osnovne skole */
    /*Sabiraju se cifre oba broja iz tekuceg razreda,  pocev od razreda jedinica ,ka razredima vece tezine. */
    /*Vodi se racuna o eventualnom nastalom prenosu u sledeci razred cifara, */
    /*kao i da ako je neki od brojeva "kraci",  postavljaju mu se zamisljene nule  kao cifre tekuceg razreda .*/
    /*Te zamisljene  nule su vodece nule,  i zato ne uticu na izmenu vrednosti broja */

for(i=0; i<*ns; ++i) {
      c= (i<n ? cifra(a1[i]) : 0) + (i<m ? cifra(a2[i]) : 0) + prenos;
      s[i]= znak(c % 10) ;              /*formira se cifra rezultata ,kao vrednost arapske cifre u masinskom setu znakova */
      prenos= c < 10 ? 0 : 1;                 /* prenos=c/10; */
   }    //kraj for petlje

    if(prenos > 0) {
      s[*ns]=znak(prenos);
      *ns=*ns+1;
   }
   s[*ns]='\0';  /*zbog rada sa niskom cifara */
}     //kraj potprograma
char* reverse( char s[] )
{
  int indeks,br; /* brojacke promenljive */
  char  Temp;/* posrednik u razmeni dva karaktera sa iz leve i desne polovine stringa */
    br=strlen(s)-1;
      for( indeks=0; indeks< br; indeks++, br-- )
 {   Temp=s[indeks];   s[indeks]=s[br];   s[br]=Temp;  }
 return s;
}
```

5. Napisati C program koji će pomnožiti dva broja koji se zadaju kao dva parametra komandne linije.

6. Napisati funkciju koja omogucava racunanje proizvoda dve kvadratne matrice dimenzija n×n. Napisati program koji omogucava unosenje dve kvadratne matrice i stampanje proizvoda te dve matrice.

7. Napisati program koji sa ulaza ucitava matricu dimenzija m × n i na standardnom izlazu ispisuje:
(a) Vrednost najveceg elementa na sporednoj dijagonali.
(b) Indeks kolone koja sadrzi najmanji element matrice.
(c) Indeks vrste koja sadrzi najveci element matrice.
(d) Broj negativnih elemenata matrice.
Dimenzije m i n se zadaju kao argumenti komandne linije.

8. NCP koji sa standardnog ulaza učitava kvadratnu matricu kojom je predstavljena neka relacija i ispisuje da li relacija je refleksivna, simetrična, tranzitivna. Dakle, elementi matrice su 0 i 1 (vrednost 0 u i-tom redu i j-toj koloni označava da i-ti element skupa nije u relaciji sa j-tim elementom). U prvoj liniji standardnog ulaza nalazi se dimenzija matrice n, a zatim se učitava n redova sa po n elemenata. Elementi matrice su razdvojeni blanko znakom. Dimenzija matrice jeste celobrojna, ali nije unapred poznata.

```
#include <stdio.h>
#include <stdlib.h>
/* Funkcija ucitaj_matricu() ucitava matricu u zadatom formatu sa standardnog  ulaza. Argumenti funkcije su:
  pn - pokazivac na lokaciju koja ce sadrzati dimenziju matrice
  pA - pokazivac na lokaciju koja ce opisivati matricu, pri cemu je matrica
       predstavljena poljem pokazivaca na redove matrice (za element matrice
       se koristi char tip zato sto element moze imati samo vrednosti 0 ili 1*/
void ucitaj_matricu (int *pn, char ***pA);
/* Funkcija proveri_refleksivnost() proverava refleksivnost relacije predstavljene matricom. Argumenti funkcije su:
  n - dimenzija matrice
  A - matrica predstavljena poljem pokazivaca
```

```
Funkcija vraca 1 ukoliko je relacija refleksivna, odn. 0 ako nije. */
int proveri_refleksivnost (int, char **);

/* Funkcija proveri_simetricnost() proverava simetricnost relacije predstavljene matricom. Argumenti funkcije su:
  n - dimenzija matrice
  A - matrica predstavljena poljem pokazivaca
Funkcija vraca 1 ukoliko je relacija simetricna, odn. 0 ako nije. */
int proveri_simetricnost (int, char **);
/* Funkcija proveri_tranzitivnost() proverava tranzitivnost relacije predstavljene matricom. Argumenti funkcije su:
  n - dimenzija matrice
  A - matrica predstavljena poljem pokazivaca
Funkcija vraca 1 ukoliko je relacija tranzitivna, odn. 0 ako nije. */
int proveri_tranzitivnost (int, char **);

main ()
{
int n;   /* Dimenzija matrice. */
char **A;   /* Matrica predstavljena poljem pokazivaca. */
int i;   /* Brojac u petlji. */
/* Ucitavanje  matrice */
  printf ("Uneti matricu u zadatom formatu:\n");
  ucitaj_matricu (&n, &A);
/* Provera relacija predstavljenih matricom. Ako je svaka od njih zadovoljena, ispisuje se odgovarajuca poruka. */
  if (proveri_refleksivnost (n, A))
    printf ("Matrica je refleksivna\n");
  if (proveri_simetricnost (n, A))
    printf ("Matrica je simetricna\n");
  if (proveri_tranzitivnost (n, A))
    printf ("Matrica je tranzitivna\n");
/* Oslobadjanje memorije koju je zauzimala matrica. */
  for (i = 0; i < n; i++)
    free (A[i]);
  free (A);
  /* Zavrsava se program. */
  exit (EXIT_SUCCESS);
}
void  ucitaj_matricu (int *pn, char ***pA)
{
  int clan;   /* Tekuci element matrice koji se ucitava. */
  int i, j;   /* Brojaci u petljama. */
  /* Ucitava se dimenzija matrice. */
  scanf ("%d", pn);
  /* Alocira se memorija za polje pokazivaca na redove matrice, kao i za      same redove. */
  *pA = (char **) malloc (*pn * sizeof (char *));
  for (i = 0; i < *pn; i++)
    (*pA)[i] = (char *) malloc (*pn * sizeof (char));
  /* Ucitava se jedan po jedan element matrice. Ucitavanje se vrsi u int  promenljivu, a potom se vrednost te
promenljive prepisuje u element   matrice koji je char tipa.       Ovako je jednostavnije, jer se ucitavanje vrsi
formatirano (pomocu scanf() funkcije, a elementi matrice mogu biti  samo 0 i 1 tako da je za njihovo predstavljanje
upotrebljen char tip da  se ne bi rasipala memorija. Vredi razmisliti o daljem poboljsanju efikasnosti  tako sto bi se
elementi matrice smestali u pojedinacne bitove. */
  for (i = 0; i < *pn; i++)
    for (j = 0; j < *pn; j++)
      {
 scanf ("%d", &clan);
 (*pA)[i][j] = (char) clan;
```

```
    }
}
int proveri_refleksivnost (int n, char **A)
{
  int i; /* Brojac u petlji. */
/* Proverava se refleksivnost i iz petlje se izlazi cim se ustanovi da relacija nije zadovoljena. */
  for (i = 0; i < n; i++)
    if (!A[i][i])
      return 0;
  /* Ako se proslo kroz citavu petlju, relacija je zadovoljena. */
  return 1;
}
int proveri_simetricnost (int n, char **A)
{
  int i, j;   /* Brojaci u petljama. */
  /* Proverava se simetricnost i iz petlji se izlazi cim se ustanovi da relacija nije zadovoljena. Pri proveri se prelazi
samo deo matrice
     iznad gornje dijagonale, jer je tako dovoljno. Uociti da pri ispitivanju  simetricnosti elementi moraju biti razliciti.
*/
  for (i = 0; i < n - 1; i++)
    for (j = i + 1; j < n; j++)
      if (A[i][j] != A[j][i])
 return 0;
  /* Ako se proslo kroz obe petlje, relacija je zadovoljena. */
  return 1;
}
int proveri_tranzitivnost (int n, char **A)
{
  int i, j, k;   /* Brojaci u petljama. */
  /* Proverava se tranzitivnosti i iz petlji se izlazi cim se ustanovi da relacija nije zadovoljena. Uociti da pri ispitivanju
tranzitivnosti
     elementi moraju biti razliciti. Tranzitivnost se moze matematicki  interpretirati i na nesto drugaciji nacin*/
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      if (i != j)
 for (k = 0; k < n; k++)
   if (k != i && k != j)
     if (A[i][k] && A[k][j] && !A[i][j])
       return 0;
  /* Ako se proslo kroz sve tri petlje, relacija je zadovoljena. */
  return 1;
}
```

9. Napisati C program koji sa standardnog ulaza učitava kvadratnu matricu čiji elementi su celi brojevi i ispisuje da li matrica je ortogonalna. U prvoj liniji standardnog ulaza nalazi se dimenzija matrice n, a zatim se učitava n redova sa pon elemenata. Elementi matrice su razdvojeni blanko znakom. Dimenzija matrice jeste celobrojna, ali nije unapred poznata. Može se pretpostaviti da sve linije sem eventualno prve su u ispravnom formatu.

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
void ucitaj_matricu (unsigned *dim, int ***matrica);
/* ucitavanje  matrice  u formulisanom formatu sa standardnog  ulaza
  dim <-> pokazivac na lokaciju koja sadrzi dimenziju matrice
  matrica< -> pokazivac na lokaciju koja opisuje  matricu, pri cemu matrica se
 predstavlja poljem pokazivaca na vrste matrice koje sadrze int  clanove*/
```

```
int ortogonalna( int **matrica , int dim);
/*funkcija ortogonalna proverava ortogonalnost matrice.
 matrica - pokazivac na lokaciju koja ce opisivati matricu */

/* glavna funkcija */
main ()
{
  int **a;   /* matrica kao polje pokazivaca. */
  unsigned Dim;         /* dimenzija matrice. */
  int i;   /* lokalni brojac  */
  /* ucitavanje  matrice */
  printf ("Uneti matricu u opisanom obliku:\n");
  ucitaj_matricu (&Dim, &a);
  /* Proverava se ortogonalnost matrice i ispisuje se odgovarajuca poruka. */
  if (ortogonalna(a,Dim))
    printf ("Matrica je ortogonalna\n");
 else printf ("Matrica nije ortogonalna\n");
    /* Oslobadja se memorija koju je zauzimala matrica. */
  for (i = 0; i < Dim; i++)
    free (a[i]);
  free (a);
  return (EXIT_SUCCESS);
}
void ucitaj_matricu (unsigned *dim, int  ***matrica)
{
  int pom;  /* element matrice koji se ucitava. */
  int i,j;        /* brojaci  */
  int c;          /*karakter prve linije */
  int preth;      /*indikator nepraznosti prve linije */
/* ucitavanje  dimenzije  matrice. */
for(preth=0,*dim=0;(c=getchar() ) !='\n' ;*dim=*dim*10+(c-'0'),preth=1 )
  {if ( !isdigit(c) )
     { fprintf(stderr,"Nekorektan unos dimenzije"); exit(1); }
  }
if (!preth)
  {fprintf(stderr,"Nekorektan unos dimenzije \n"); exit(1); }
/* alociranje memorija za polje pokazivaca na vrste matrice, a potom i za
 same vrste */
  if  (  (*matrica = (int **) malloc (*dim * sizeof (int *)) ) == NULL)
            fprintf(stderr,"Ne moze da se alocira prostor\n");
  for (i = 0; i < *dim; i++)
    (*matrica)[i] = (int *) malloc (*dim * sizeof (int));
  /* ucitavanje elemenata matrice.      */
  for (i = 0; i < *dim; i++)
    for (j = 0; j < *dim; j++)
      {
 scanf ("%d", &pom);
 (*matrica)[i][j] = pom;
      }
}

int ortogonalna( int  **matrica , int   dim)
{
 int i,j,k,pom ,ort=1 ;
for (i=0;i< dim && ort; i++)
  for (j=0;j< dim && ort; j++ )
{ pom=0;
```

```
  for (k=0; k< dim; k++)
   /*izracunati clan [i,j] matrice (A)*(A^T)   */
   pom+= matrica[i][k] * matrica[j][k];
/*test (van)dijagonalnih clanova proizvoda */
ort= ( (i==j) && (pom==1) ) || ( (i!=j) && (pom==0) );
}
return ort;
}
```