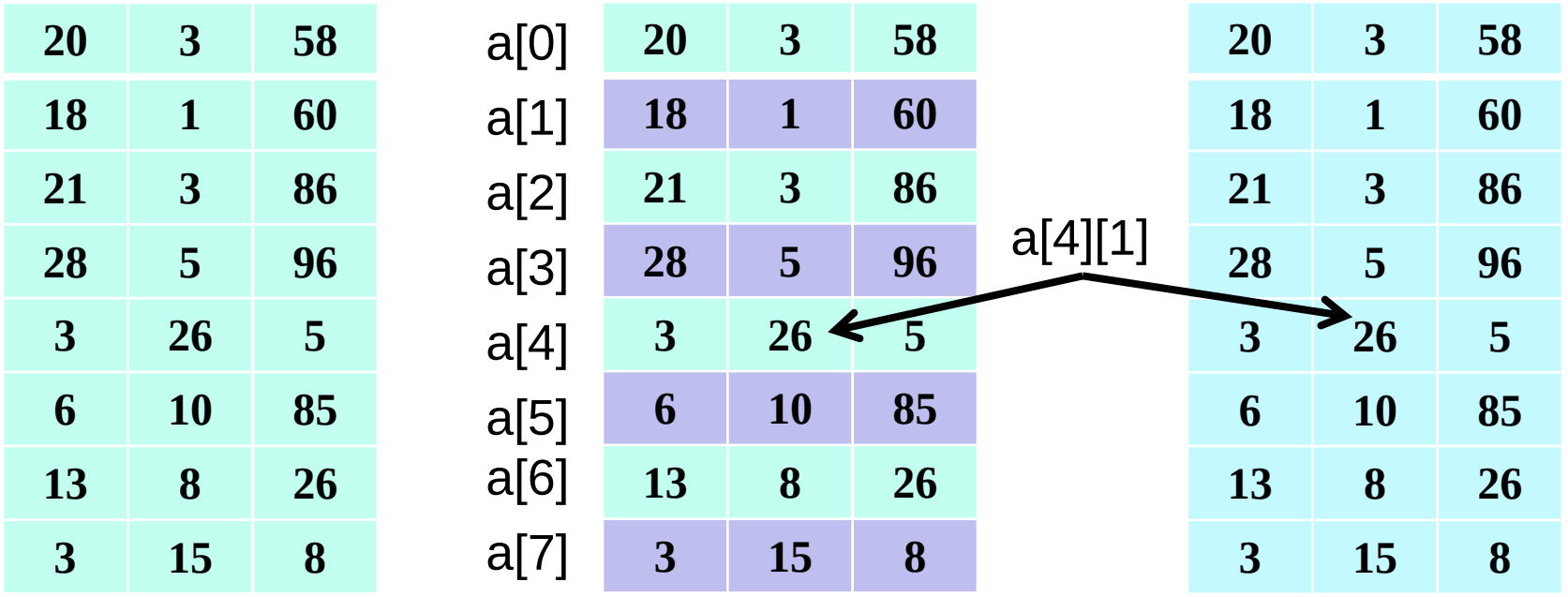
IX – Rad sa matricama

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

20	18	21	28	3	6	13	3

- Međutim, u velikom broju realnih problema znatno je bolje da se takvi podaci predstave u tabelarnoj formi, kao što je prikazano na slici:



IX – Rad sa matricama

- 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 prethodnoj slici (druga tabela).
- Tabela visine **m** (8 redova) i širine **n** (3 kolone), može se predstaviti kao niz dužine **m** čiji su elementi nizovi dužine **n**.

Primer: *Pogledajmo kako bi izgledao C kod kojim bi se deklarisala tabela **a** visine 8 i širine 3 elementa, pri čemu su elementi celi brojevi:*

```
typedef int vrsta[3];
typedef vrsta tabela[8];
tabela a;
```

- U prethodnom primeri definisan je tip podatka **vrsta** koji predstavlja niz od 3 elementa tipa **int**.
- Zatim je definisan tip podatka **tabela** koji predstavlja niz od 8 elemenata tipa **vrsta**.
- U poslednjem redu deklarisana je promenljiva **a** tipa **tabela**.
- Promenljiva **a** će predstavljati niz od **8** celobrojnih nizova dužine **3**.

IX – Rad sa matricama

- Pojedinačnoj vrsti se može pristupiti korišćenjem zapisa **a[i]** gde je **i** redni broj vrste.
- Elementu u **i**-oj vrsti i **j**-oj koloni pristupa se korišćenjem zapisa **a[i][j]**
- Sa obzirom da je ovakav način pisanja prilično komplikovan, u C-u je omogućeno direktno definisanje višedimenzionalnih nizova.
- Dvodimenzionalne nizove najčešće nazivamo **matrice.**
- Matrica celih brojeva **a**, dimenzija 3x4, pri čemu je 3 broj vrsta, a 4 broj kolona, može se skraćeno deklarisati na sledeći način

typedef int matrica[3][4]; matrica a;

ili direktno kao

int a[3][4];

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

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

IX – Rad sa matricama

- Kod rešavanja mnogih problema postoji potreba za radom sa višedimenzionalnim strukturama podataka.
- U programskim jezicima se za rad sa ovakvim strukturama podataka definišu višedimenzionalna polja - jednorodne strukture podataka u okviru kojih se svakom elementu pristupa preko dva ili više indeksa.
- Posebno su značajne dvodimenzionalne strukture, koje odgovaraju pojmu matrica i omogućavaju rešavanje veoma široke klase problema
- Jednim parom zagrada definišu se jednodimenzionalni nizovi.
- Svaki par zagrada definiše jednu dimenziju niza.

Primer:
```
int a[100];      - jednodimenzionalna struktura
int b[3][5];     - dvodimenzionalna struktura
int c[7][9][2];  - trodimenzionalna struktura
```

- U jeziku C proizvoljni k-dimenzionalni niz poseduje veličinu svake od svojih k dimenzija.
- Višedimenzionalnim nizovima se pristupa preko dva ili više indeksa.

VIII – Višedimenzionalna polja

- U programskom jeziku C predstavljanje podataka datih u tabelarnoj formi se može obaviti na više načina:

```
1. int matrica1[3][4]={25,26,27,28,29,24,23,21,20,10, 11, 12};
2. int matrica2[3][4] = {
                          {25, 26, 27, 28}
                          {29, 24, 23, 21}
                          {20, 10, 11, 12}
                        };
3. int matrica3[ ][4] = {25,26,27,28,29,24,23,21,20,10, 11, 12};
```

- Deklaracija višedimenzionalnog polja ima oblik

mem_klasa tip ime[izraz_1][izraz_2]...[izraz_n];

gde je **mem_klasa** memorijska klasa, **tip** je tip podatka, **ime** je ime polja, a **izraz 1, . . . , izraz n** su konstantni celobrojni pozitivni izrazi koji određuju broj elementa polja vezanih uz pojedine indekse.

- Tako se prvi indeks kreće od 0 do **izraz_1-1**, drugi od 0 do **izraz_2-1**, n-ti indeks od 0 do **izraz_n-1.**

VIII – Višedimenzionalna polja

Primer: Kada polje **m** deklarišemo sa **static float x[3][4];** onda ono predstavlja matricu sa tri reda i četri kolone. Elemente matrice možemo prostorno zamisliti na sledeći način:

x[0][0]	x[0][1]	x[0][2]	x[0][3]
x[1][0]	x[1][1]	x[1][2]	x[1][3]
x[2][0]	x[2][1]	x[2][2]	x[2][3]

U prvom redu su elementi **x[0][i]**, u drugom redu **x[1][i]** a u trećem redu **x[2][i]** za i = 0, 1, 2, 3.

- Element na mestu (i,j) matrice **x** je **x[i][j].**
- Deklaracija se može čitati i ovako: definisan je niz u kome se nalaze elementi 3 niza i to svaki sa 4 elementa tipa int.
- Dvodimenzionalni nizovi se često koriste za rad sa matricama.
- U tom slučaju nije potrebno razmišljati o tome kako je niz složen u memoriji, jer se elementima pristupa preko dva indeksa: prvi je oznaka **reda**, a drugi je oznaka **kolone** matrice.
- Ako je broj redova i kolona isti radi se o **kvadratnoj matrici**

VIII – Višedimenzionalna polja

- Memorijski raspored elemenata dvodimenzionalnog niza, koji opisuju neku matricu, je takav da su elementi složeni po redovima matrice; najpre prvi red, zatim drugi, itd..

Adresa	*Sadržaj*
1000	x[0][0]
1004	x[0][1]
1008	x[0][2]
1012	x[0][3]
1016	x[1][0]
1020	x[1][1]
1024	x[1][2]
1028	x[1][3]
1032	x[2][0]
1036	x[2][1]
1040	x[2][2]
1044	x[2][3]

Prvi red matrice

Drugi red matrice

Treći red matrice

x[0][0]	x[0][1]	x[0][2]	x[0][3]
x[1][0]	x[1][1]	x[1][2]	x[1][3]
x[2][0]	x[2][1]	x[2][2]	x[2][3]

VIII – Višedimenzionalna polja

- Višedimenzionalni niz se može inicijalizirati već u samoj deklaraciji.

Primer:
```
int x[3][4] = { { 1, 21, 14, 8},
                {12, 7, 41, 2},
                { 1,  2,  4,  3} };
```

- Navođenje unutrašnjih velikih zagrada je opciono, pa se može pisati i sledeća deklaracija: **int x[3][4] = {1, 21, 14, 8, 12, 7, 41, 2, 1, 2, 4, 3};**
- Ovaj drugi način inicijalizacije se ne preporučuje, jer je zbunjujući jer je teže uočiti raspored elemenata po redovima i kolonama.
- Elementima se pristupa preko indeksa niza.

Primer: *Izračunati sumu svih elemenata matrice x*;

```
int i, j, brojredova=3, brojkolona=4;
int sum=0;
for (i = 0; i < brojredova; i++)
   for (j = 0; i < brojkolona; j++)
   sum += x[i][j];
printf("suma elemenata matrice = %d", sum);
```

VIII-Prenos višedimenzionalnih nizovu u F-ju

- Kod deklarisanja parametara f-je, koji su višedimenzionalni nizovi pravilo je da se ne navodi prva dimenzija (kao i kod 1-dimenzionalnih nizova), ali ostale dimenzije treba deklarisati, kako bi program prevodilac "znao" kojim su redom elementi složeni u memoriji.

Primer: *Definisana je f-ja* ***sum_mat_el()*** *kojom se računa suma elemenata dvodimenzionalne matrice, koja ima 4 kolone*:

```
int sum_mat_el, int x[][4], int brojredova)
{
int i, j, sum=0;
for (i = 0; i < brojredova; i++)
   for (j = 0; i < 4; j++)
      sum += x[i][j];
return sum;
}
```

- U definiciji f-je naveden je argument koji opisuje broj redova matrice.
- Broj kolona je fiksiran već u deklaraciji niza na vrednost 4.
- Ima ograničenu upotrebu jer se može primeniti samo na matrice koje imaju 4 kolone.

VIII - Primer učitavanja matrice

Primer: *Učitavanje matrice*

```
for ( red=1; red<=n; red++ )
   for ( kol=1; kol<=m; kol++ )
   {
      printf("mat[%d][%d]=",red-1,kol-1);
      scanf("%d", &mat[red-1][kol-1]);
   }

ili:

for ( red=0; red<n; red++ )
   for ( kol=0; kol<m; kol++ )
   {
      printf("mat[%d][%d]=",red,kol);
      scanf("%d", &mat[red][kol]);
   }
```

VIII - Primer ispisivanja matrice

Primer: Ispisivanje matrice

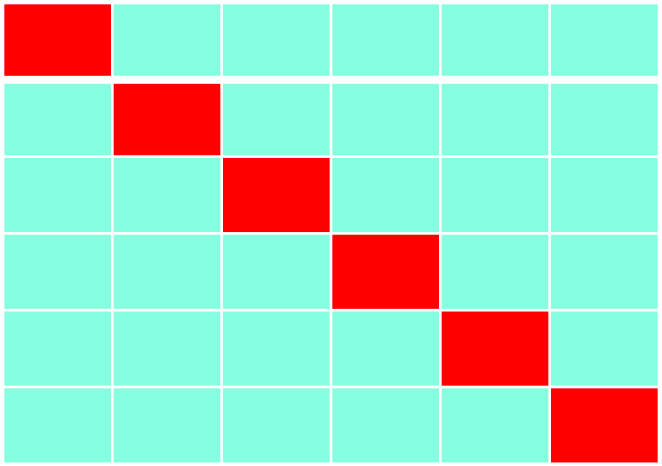
```
for ( red=1; red<=n; red++ )
{
   for ( kol=1; kol<=m; kol++ )
      printf(" %4d",mat[red-1][kol-1]);
      printf("\n");
}

ili:

for ( red=0; red<n; red++ )
{
   for ( kol=0; kol<m; kol++ )
      printf(" %4d",mat[red][kol]);
      printf("\n");
}
```

VIII – Rad sa matricama

Primer: *manipulacije glavnom dijagonalom*



mat[0][0]
mat[1][1]

mat[i][i]
mat[n-1][n-1]

→ **mat[i][j], i=j**

Ispis elemenata na glavnoj dijagonali:

```
printf("Glavna dijagonala: ");
for ( i=0; i<n; i++ ) printf(" %d", mat[i][i]);
```

Suma elemenata na glavnoj dijagonali:

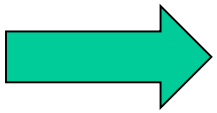
```
for ( s=i=0; i<n; i++ ) s+=mat[i][i];
printf("Suma elemenata na GD: %d",s);
```

IX - Transponovanje matrice

Primer: *transponovanje matrice*

1	2	3
4	5	6
7	8	9

Polazna matrica



1	4	7
2	5	8
3	6	9

Transportovana matrica

Transponovanje matrice:

```
for ( i=0; i<n; i++ )
  for ( j=i+1; j<n; j++ )
  {
    pom=mat[i][j]);
    mat[i][j]=mat[j][i];
    mat[j][i]=pom;
  }
```

Ispis transponovane matrice na osnovu originalne matrice:

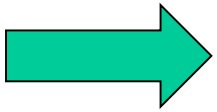
```
printf("Transponovana:\n ");
for ( i=0; i<n; i++ )
{
  for ( j=0; j<n; j++ )
  printf(" %4d", mat[j][i]);
  printf("\n");
}
```

IX - Množenje dve matrice

1	2	3
4	5	6
1	0	2

*

1	2	3
4	2	1
7	1	0



30	9	5
66	24	17
15	4	3

Matrica a (m*n)

Matrica b (n*p)

Matrica c (m*p)

Množenje matrica:

```
for ( i=0; i<m; i++ )          /* red matrice c */
  for ( j=0; j<p; j++ )        /* kolona matrice c */
    for ( k=0; k<n; k++ )
      c[i][j]+=a[i][k]*b[k][j];
```

Rešenje:

c[0][0]=1*1+2*4+3*7=**30**, c[0][1]=1*2+2*2+3*1=**9**, c[0][2]=1*3+2*1+3*0=**5**

c[1][0]=4*1+5*4+6*7=**66**, c[1][1]=4*2+5*2+6*1=**24**, c[1][2]=4*3+5*1+6*0=**17**

c[2][0]=1*1+0*4+2*7=**15**, c[2][1]=1*2+0*2+2*1=**4**, c[2][2]=1*3+0*1+2*0=**3**

IX-Pokazivači i višedimenzionalni nizovi

- Dvodimenzionalni nizovi jesu **jednodimenzionalni** nizovi jednodimenzionalnih nizova.
- Na primer, ako je **r** ime dvodimenzionalnog niza (matrice), tada je **r[i]** jednodimenzionalni niz koji sadrži elemente i-te vrste matrice **r.**
- Dvodimenzionalni nizovi (matrice) se u memoriji registruju po vrstama, koristeći uzastopne memorijske lokacije.
- Na primer, dvodimenzionalni niz **int a[3][2]** se u memoriji raspoređuje u sledećem poretku:

  a[0][0], a[0][1], a[1][0], a[1][1], a[2][0], a[2][1] .
- Ako se deklariše pokazivač **p** pomoću **int *p** posle dodele **p=a** pointer **p** uzima za svoju vrednost adresu elementa koji leži u nultoj vrsti i nultoj koloni matrice **a** (tj. adresu prvog elementa matrice **a**).
- Takođe, važe sledeće pointerske jednakosti:
  p=&a[0][0], p+1=&a[0][1], p+2=&a[1][0], p+3 =&a[1][1], p+4=&a[2][0], p+5=&a[2][1].
- Na sledećoj slici prikazano je smeštanje elemenata matrice **x[2][3]** sa odgovarajućim vrednostima pokazivača na te vrednosti

IX-Pokazivači i višedimenzionalni nizovi

x[0][0]	x[0][1]	x[0][2]	x[0][3]
x[1][0]	x[1][1]	x[1][2]	x[1][3]
x[2][0]	x[2][1]	x[2][2]	x[2][3]

Adresa	Sadržaj	Pointer
1000	x[0][0]	p=&x[0][0]
1004	x[0][1]	p+1=&x[0][1]
1008	x[0][2]	p+2=&x[0][2]
1012	x[0][3]	p+3=&x[0][3]
1016	x[1][0]	p+4=&x[1][0]
1020	x[1][1]	p+5=&x[1][1]
1024	x[1][2]	p+6=&x[1][2]
1028	x[1][3]	p+7=&x[1][3]
1032	x[2][0]	p+8=&x[2][0]
1036	x[2][1]	p+9=&x[2][1]
1040	x[2][2]	p+10=&x[2][2]
1044	x[2][3]	p+11=&x[2][3]

IX-Pokazivači i višedimenzionalni nizovi

- U našem slučaju, vrste matrice **a** date su izrazima **a[0], a[1] i a[2].**
- Sa obzirom na poznata svojstva jednodimenzionalnih nizova, važe sledeće jednakosti:

  a[0]=&a[0][0], a[1]=&a[1][0], a[2]=&a[2][0] .
- Osim toga, važe sledeće jednakosti:

a[i]=&a[i][0], a[i]+j=&a[i][j], a[i][j]=*(a[i]+j), &a[i][j] = a+i*2+j

- Takođe je

  a[j][k]=*(a+j*duzina_vrste +k).
- Neka je **mat[][7]** dvodimenzionalni niz.
- Tada važi:

  mat[i][j]=*(mat[i]+j)=(*(mat+i))[j]=*(mat+i*7+j)=
  ***(&mat[0][0]+i*7+j)= *((*(mat+i))+j).**
- U gornjim primerima zagrade su neophodne zbog toga što operator selekcije ima viši prioritet u odnosu na operator indirekcije.

IX – Rad sa poljima i matricama

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);
}
```

IX – Rad sa poljima i matricama

Primer 2 : *Napisati program koji pozivanjem potprograma vrši transponovanje kvadratne matrice celih brojeva* ***a*** *dimenzija* ***n x n***

```
#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);
}
```

IX-Sabiranje i oduzimanje matrica

Primer: *Učitaj dve matrice od* ***n*** *redova i* ***m*** *kolona i ispiši njihov zbir i razliku. Samo matrice koje imaju isti broj redova i kolona se mogu sabirati i oduzimati.*

$$c = a + b = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ & & \vdots & \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ & & \vdots & \\ b_{m1} & b_{m2} & \cdots & b_{mn} \end{bmatrix} =$$

$$\begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \dots & a_{1n}+b_{1n} \\ a_{21}+b_{21} & a_{22}+b_{22} & \dots & a_{2n}+b_{2n} \\ & & \vdots & \\ a_{m1}+b_{m1} & a_{m2}+b_{m2} & \dots & a_{mn}+b_{mn} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ & & \vdots & \\ c_{m1} & c_{m2} & \cdots & c_{mn} \end{bmatrix}$$

IX-Sabiranje i oduzimanje matrica

```
#include<conio.h>
#include<stdio.h>
int a[10][10],b[10][10],c[10][10],d[10][10];
int i,j,n,m;
void main(void)
{
clrscr();
printf ("upisi broj redova matrica: ");
scanf("%d",&n);
printf("upisi broj kolona matrica: ");
scanf("%d",&m);
printf("upisi elemente prve matrice:\n");
for(i=0;i<n;i++)
          for(j=0;j<m;j++)
          {
             printf("a[%d][%d]: ",i,j);
             scanf("%d",&a[i][j]);
          }
printf("upisi elemente druge matrice: \n");
for(i=0;i<n;i++)
     for(j=0;j<m;j++)
{
          printf("%d b[%d][%d]:",i,j);
          scanf("%d",&b[i][j]);
}
for(i=0;i<n;i++)
    for(j=0;j<m;j++)
    {
       c[i][j]=a[i][j]+b[i][j];
       d[i][j]=a[i][j]-b[i][j];
     }
clrscr();
printf("\n zbroj ucitanih matrica je: \n");
for(i=0;i<n;i++)
{
   for(j=0;j<m;j++)
      printf("%d  ",c[i][j]);
printf("\n");}
printf("\n razlika ucitanih matrica je: \n");
for(i=0;i<n;i++)
{
   for(j=0;j<m;j++)
      printf("%d  ",d[i][j]);
printf("\n");         }
getch();
}
```

IX – Množenje matrica

Primer: *Učitati dve matrice i pomnožiti ih.*

Napomena: Množiti se mogu jedino dve matrice kod kojih je broj redova druge matrice jednak broju kolona prve matrice. Ako je A matrica tipa (n,m), a B tipa (m,k) onda će C=A*B biti tipa (n,k).

$$c = a \cdot b = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ & & \vdots & \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1k} \\ b_{21} & b_{22} & \cdots & b_{2k} \\ & & \vdots & \\ b_{m1} & b_{m2} & \cdots & b_{mk} \end{bmatrix} =$$

$$\begin{bmatrix} a_{11}b_{11}+a_{12}b_{21}+\ldots+a_{1m}b_{m1} & \cdots & a_{11}b_{1k}+a_{12}b_{2k}+\ldots+a_{1m}b_{mk} \\ a_{21}b_{11}+a_{22}b_{21}+\ldots+a_{2m}b_{m1} & \cdots & a_{21}b_{1k}+a_{22}b_{2k}+\ldots+a_{2m}b_{mk} \\ & \vdots & \\ a_{n1}b_{11}+a_{n2}b_{21}+\ldots+a_{nm}b_{m1} & \dots & a_{n1}b_{1k}+a_{n2}b_{2k}+\ldots+a_{nm}b_{mk} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ & & \vdots & \\ c_{m1} & c_{m2} & \cdots & c_{mn} \end{bmatrix}$$

IX - Množenje matrica

```
#include<conio.h>
#include<stdio.h>
int a[10][10],b[10][10],c[10][10],d[10][10];
int i,j,n,m,k,l;
void main(void){
clrscr();
printf ("upisi broj redova prve matrice: ");
scanf("%d",&n);
printf("upisi broj kolona prve matrice: ");
scanf("%d",&m);
printf("upisi broj kolona druge matrice: ");
scanf("%d",&k);
printf("upisi elemente prve matrice:\n");
for(i=0;i<n;i++)
  for(j=0;j<m;j++)
  {
    printf("a[%d][%d]: ",i,j);
    scanf("%d",&a[i][j]);
  }
printf("upisi elemente druge matrice: \n");
for(i=0;i<m;i++)
  for(j=0;j<k;j++)
  {
  printf("b[%d][%d]:",i,j);scanf("%d",&b[i][j]);
  }
for(i=0;i<n;i++)
  for(j=0;j<k;j++)
  {
    c[i][j]=0;
    for(l=0;l<m;l++)
      c[i][j]=c[i][j]+a[i][l]*b[l][j];
  }
printf("\n proizvod ucitanih matrica je: \n");
  for(i=0;i<n;i++)
  {
    for(j=0;j<k;j++)
          printf("%d  ",c[i][j]);
          printf("\n");
  }
getch();
}
```

IX – Pretraživanje matrice (dijagonale)

Primer: *Učitati kvadratnu matricu reda* ***n****. Ispisati matricu u obliku tablice i elemente na glavnoj i sporednoj dijagonali*. Elementi glavne dijagonale su oni elementi matrice kojima je jednak indeks reda i kolone. Za elemente sporedne dijagonale važi da je suma indeksa reda i kolone jednaka n+1,tj.da se indeks reda povećava, a indeks kolone smanjuje za 1

```
#include<conio.h>
#include<stdio.h>
int a[10][10],b[10],c[10],i,j,n;
void main(void)
{
clrscr();
printf ("upisi red matrice: ");
scanf("%d",&n);
printf("upisi elemente matrice:\n");
for(i=0;i<n;i++)
  for(j=0;j<n;j++)
  {
    printf("a[%d][%d]: ",i,j);
    scanf("%d",&a[i][j]);
  }
for(i=0;i<n;i++)
  {
    b[i]=a[i][i];
    c[i]=a[i][n-i-1];
  }
for(i=0;i<n;i++)
  {
    for(j=0;j<n;j++)
      printf("%d  ",a[i][j]);
      printf("\n");
  }
printf("\n elementi na glavnoj dijagonali su: ");
  for(i=0;i<n;i++)
    printf("%d  ",b[i]);
printf("\n elementi sporedne dijagonale su: ");
for(i=0;i<n;i++)
  printf("%d  ",c[i]);
getch();
}
```

IX – Rad sa poljima i matricama

Primer: *Učitaj jednu kvadratnu matricu reda* ***n****. Ispiši najveći i najmanji element u svakom redu i koloni. Dobijene elemente smestiti u jednodimenzionalna polja.*

```
#include<conio.h>
#include<stdio.h>
int a[10][10],maxr[10],maxs[10],minr[10];
int mins[10],i,j,n;
void main(void)
{
clrscr();
printf ("upisi red matrice: ");scanf("%d",&n);
printf("upisi elemente matrice:\n");
for(i=0;i<n;i++)
  for(j=0;j<n;j++)    {
            printf("a[%d][%d]: ",i,j);
            scanf("%d",&a[i][j]);            }
for(i=0;i<n;i++)       {
  minr[i]=a[i][0];     maxr[i]=a[i][0];
  mins[i]=a[0][i];     maxs[i]=a[0][i];
  for (j=1;j<n;j++) {
    if(a[i][j]<minr[i])minr[i]=a[i][j];
      if(a[i][j]>maxr[i])maxr[i]=a[i][j];
      if(a[j][i]<mins[i])mins[i]=a[j][i];
        if(a[j][i]>maxs[i])maxs[i]=a[j][i];
        } }
for(i=0;i<n;i++)       {
            for(j=0;j<n;j++)
              printf("%d  ",a[i][j]);
              printf("\n");         }
for(i=0;i<n;i++)
printf("\n najveci u redu %d je %d, a najmanji
%d ",i+1,maxr[i],minr[i]);
for(i=0;i<n;i++)
printf("\n najveci u stupcu %d je %d, a
najmanji %d ",i+1,maxs[i],mins[i]);
getch();
}
```

Hvala na pažnji !!!

Pitanja

? ? ?