

# Programiranje podatkovna struktura polje

## Podatkovna struktura *polje*

- Polje je *podatkovna struktura* gdje isto ime dijeli više podataka
- Svi podaci u nekom polju moraju biti istog tipa
- Elementima polja se pristupa koristeći indeks, koji mora biti nenegativni cijeli broj (konstanta, varijabla, cjelobrojni izraz).
- Indeks elemenata je broj između 0 i broja elemenata minus jedan, uključivo, tj.  
 $\text{indeks} \in [0, \text{BrojElementata} - 1]$ .
- Indeks može biti nenegativni cijeli broj (konstanta, varijabla, cjelobrojni izraz)  
 $x[0] \quad x[9] \quad x[n] \quad x[\text{MAX}] \quad x[n+1] \quad x[k/m+5]$
- Elementi polja susjedni po indeksu susjedni su i u memoriji računala

2

## Definicija i deklaracija polja

- Prilikom definicije polja moraju se u uglatim zagradama navesti maksimalni broj elemenata (cjelobrojna ili simbolička konstanta, ili konstantni izraz npr. `MAXTEXT+1`).

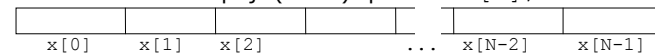
```
int x[20] – polje od 20 cijelih brojeva (članova)
char znakovi[2] – polje od 2 znaka
float niz[MAX] – MAX je simbolička konstanta
int x[100], p[MAX]; cjelobrojna polja: x od 100, a p od MAX članova
char tekst[80+1]; znakovno polje od 81 člana
```

- Indeks elementa polja kreće se u rasponu  $[0, \text{BrojElementata}-1]$
- Polje može biti dvodimenzionalno i višedimenzionalno npr.  
`float y[3][4]; double d[5][6][7];`
- Broj dimenzija nije ograničen npr.  
`int a[3][3][3][3][3][3][3][3][3][3][3][3][3][3][3];`  
**Oprez: Veličina polja  $4 * 3^{14} = 19131876$  bajta**

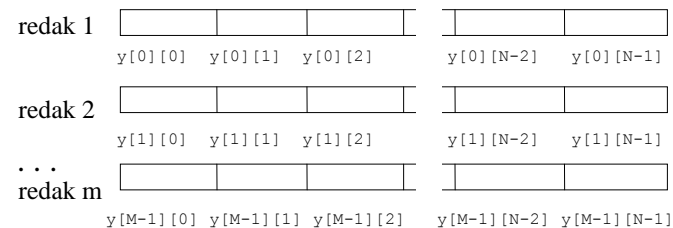
3

## Indeksiranje članova polja Jednodimenzionalna i dvodimenzionalna polja

Jednodimenzionalno polje (vektor) npr. `int x[N];`



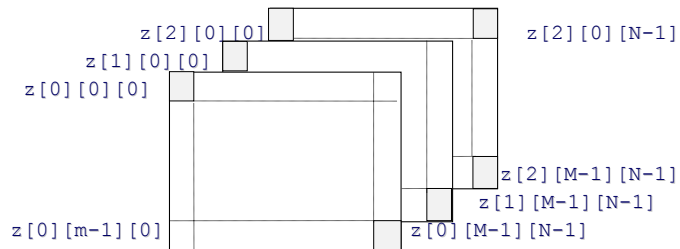
Dvodimenzionalno polje (tablica, matrica) npr. `float y[M][N];`



4

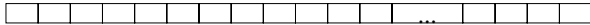
## Indeksiranje članova polja Trodimenzionalna polja

Primjer: `char z[3][M][N];`



5

## Smještaj polja u memoriji računala

- Memorija računala je jednodimenzionalna!  

- Jednodimenzionalno polje: element za elementom
- Dvodimenzionalno polje: redak za retkom  
`int a[3][4];`  
 3 susjedna jednodimenzionalna polja svako veličine 4
- Trodimenzionalno polje: sloj za slojem  
`double b[2][3][4];`  
 2 susjedna dvodimenzionalna polja veličine 3x4
- Četverodimenzionalno polje:  
`double c[5][2][3][4];`  
 5 susjednih trodimenzionalnih polja veličine 2x3x4
- ...

6

## Primjer računanja arit. sredine nenultih članova polja

- Učitati broj  $n \leq 100$  i  $n$  vrijednosti članova cjelobrojnog polja  $k$ . Izračunati aritmetičku sredinu članova polja različitih od nule. Ispisati vrijednosti članova polja i aritmetičku sredinu.

```

učitaj n
učitaj n članova cjelobrojnog polja k
postavi sumu i brojač nenultih članova na nulu
izračunaj sumu i brojač nenultih članova
ako je brojač različit od nule
  | izračunaj sredinu
inače
  | postavi sredinu na nulu
ispiši članove polja
ispiši sredinu
kraj
  
```

## Primjer računanja arit. sredine nenultih članova polja Detaljniji pseudokod

```

učitaj ( n )
učitaj n članova cjelobrojnog polja k
suma=0
brojac=0
{izračunaj sumu i brojač nenultih članova }
za i := 0 do n-1 ( s korakom 1)
  | ako je k [i] ≠ 0 onda
  |   | suma := suma + k [i]
  |   | brojac :=brojac + 1
  | ako je brojac ≠ 0 tada
  |   | sredina = suma / brojac
inače
  |   | sredina = 0
ispiši članove polja
ispiši (sredina)
kraj
  
```

8

## Primjer računanja arit. sredine nenultih članova polja Rješenje u C-u (početak programa)

```
AritmetickaSredinaPolja
/* Program za računanje aritmetičke sredine */
#include <stdio.h>

#define DIM 100

int main(){
    int k[DIM];
    int n, brojac, i, suma;
    float sredina;
    do {
        printf("Unesite broj članova polja : ");
        scanf("%d",&n);
    } while (n < 1 || n >= 100);
    for (i=0; i<n; i=i+1){
        scanf("%d", &k[i]);
    }
```

9

## Primjer računanja arit. sredine nenultih članova polja Rješenje u C-u ( nastavak programa)

```
suma = 0;
brojac = 0;

/* izračunaj sumu i brojač nenultih članova */
for (i=0; i<n; i=i+1){
    if (k[i] != 0){
        suma = suma + k[i];
        brojac = brojac + 1;
    }
}
```

10

## Primjer računanja arit. sredine nenultih članova polja Rješenje u C- (kraj programa)

```
if (brojac != 0) {
    sredina=(float)suma/
        (float)brojac;
} else {
    sredina= 0;
}

for (i=0; i<n; i=i+1){
    printf("K(%d) = %d\n", i, k[i]);
}
printf("Aritmeticka sredina %d brojeva = %f",
    brojac, sredina);

return 0;
}
```

11

## Dodjeljivanje početnih vrijednosti članovima polja: primjer gdje je broj članova polja zadan kao cjelobrojna konstanta

```
int znam[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
float x[6] = {0, 0.25, 0, -0.5, 0, 0};
char boja[3] = {'C', 'P', 'Z'};

znam[0] = 1    x[0] = 0    boja[0] = 'C'
znam[1] = 2    x[1] = 0.25  boja[1] = 'P'
znam[2] = 3    x[2] = 0    boja[2] = 'Z'
znam[3] = 4    x[3] = -0.5
znam[4] = 5    x[4] = 0
znam[5] = 6    x[5] = 0
znam[6] = 7
znam[7] = 8
znam[8] = 9
znam[9] = 10
```

12

Dodjeljivanje početnih vrijednosti članovima polja: primjer gdje je zadano manje članova polja od deklarirane dimenzije

```
int znam[10] = {1, 2, 3};
float x[6] = {-0.3, 0, 0.25};
znam[0] = 1      x[0] = -0.3
znam[1] = 2      x[1] = 0
znam[2] = 3      x[2] = 0.25
znam[3] = 0      x[3] = 0
znam[4] = 0      x[4] = 0
znam[5] = 0      x[5] = 0
znam[6] = 0
znam[7] = 0
znam[8] = 0
znam[9] = 0
```

13

Dodjeljivanje početnih vrijednosti članovima polja: primjer gdje je broj članova polja zadan brojem navedenih konstanti

```
int znam[] = {1, 2, 3, 4, 5, 6};
float x[] = {0, 0.25, 0, -0.5};
znam[0] = 1      x[0] = 0
znam[1] = 2      x[1] = 0.25
znam[2] = 3      x[2] = 0
znam[3] = 4      x[3] = -0.5
znam[4] = 5
znam[5] = 6
```

14

Dodjeljivanje početnih vrijednosti članovima polja: primjeri sa znakovnim nizovima

```
char boja[3] = {'C', 'P', 'Z'};
char bojice[] = {'C', 'P', 'Z'};
```

```
boja[0] = 'C'   bojice[0] = 'C'
boja[1] = 'P'   bojice[1] = 'P'
boja[2] = 'Z'   bojice[2] = 'Z'
```

Ako se navodi konkretna dužina znakovnog polja preporuča se da je za jedan veća od broja znakova:

```
char boja[4] = {'C', 'P', 'Z'};
```

15

Dodjeljivanje početnih vrijednosti članovima polja: primjer zadavanja početnih vrijednosti dvodimenzionalnom polju

```
int y[3][4] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};
y[0][0]= 1   y[0][1]= 2   y[0][2]= 3   y[0][3]= 4
y[1][0]= 5   y[1][1]= 6   y[1][2]= 7   y[1][3]= 8
y[2][0]= 9   y[2][1]= 10  y[2][2]= 11  y[2][3]= 12
```

Da bi program bio čitljiviji bolje je početne vrijednosti zadati na sljedeći način :

```
int y[3][4] = {
    {1, 2, 3, 4},
    {5, 6, 7, 8},
    {9, 10, 11, 12}
};
```

16

### Dodjeljivanje početnih vrijednosti članovima polja: primjer gdje je navedeno premalo vrijednosti

```
int y[3][4] = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9}
};
```

```
y[0][0]= 1   y[0][1]= 2   y[0][2]= 3   y[0][3]= 0
y[1][0]= 4   y[1][1]= 5   y[1][2]= 6   y[1][3]= 0
y[2][0]= 7   y[2][1]= 8   y[2][2]= 9   y[2][3]= 0
```

17

### Dodjeljivanje početnih vrijednosti članovima polja: primjer gdje je navedeno premalo vrijednosti

```
int y[3][4] = { 1, 2, 3, 4, 5, 6, 7, 8, 9};
```

```
y[0][0]= 1   y[0][1]= 2   y[0][2]= 3   y[0][3]= 4
y[1][0]= 5   y[1][1]= 6   y[1][2]= 7   y[1][3]= 8
y[2][0]= 9   y[2][1]= 0   y[2][2]= 0   y[2][3]= 0
```

#### Primjer kada je navedeno previše vrijednosti:

```
int y[3][4] = {
    {1, 2, 3, 4, 5},
    {6, 7, 8, 9, 10},
    {11, 12, 13, 14, 15}
};
```

18

### Pristupanje članovima polja

#### Ispravno pristupanje elementima polja:

x[0] – prvi element polja  
niz[i] – (i+1). element polja,  $0 \leq i \leq \text{BrojElemenata} - 1$   
niz[MAX - 1] – posljednji element polja

#### Neispravno pristupanje elementima polja:

```
int polje[10] = {0};
int x = polje[10];
```

```
float a = 1.;
int x = polje[a];
```

```
int a = 1, b = 0, c = 1;
int x = polje[(a && b) - c];
```

19

### Primjer izračunavanja aritmetičke sredine (1)

- Napisati program koji će tražiti unos niza brojeva, nakon čega će izračunati aritmetičku sredinu tog niza, te najprije ispisati brojeve manje od aritmetičke sredine, a zatim one koji su veći od aritmetičke sredine.

```
#include <stdio.h>
#define DIMENZIJA 10
int main() {
    int i;
    float suma = 0., arit_sred = 0., niz[DIMENZIJA]={0};

    for (i = 0; i < DIMENZIJA; i++) {
        printf("Unesi broj: ");
        scanf("%f",&niz[i]);
        suma += niz[i];
    }
}
```

20

## Primjer izračunavanja aritmetičke sredine (2)

```
arit_sred = suma / DIMENZIJA;
printf("Aritmeticka sredina niza je %6.2f.\n", arit_sred);

for (i = 0; i < DIMENZIJA; i++) {
    if (niz[i] < arit_sred) {
        printf("%6.2f je manji od arit. sredine.\n",
            niz[i]);
    }
}
for (i = 0; i < DIMENZIJA; i++) {
    if (niz[i] > arit_sred) {
        printf("%6.2f je veci od arit. sredine.\n",
            niz[i]);
    }
}

// Što ako je broj točno jednak arit_sred?
return 0;
}
```

21

## Primjer dijeljenja učitanih članova polja najvećim članom niza (1)

- Napisati program u koji će se unijeti niz brojeva. Nakon unosa treba sve članove niza podijeliti s najvećim članom niza, i iskazati ih relativno u odnosu na najveći član.

```
#include <stdio.h>
#define DIMENZIJA 10
int main() {
    int i;
    float max, polje[DIMENZIJA];
```

22

## Primjer dijeljenja učitanih članova polja najvećim članom niza (2)

```
for (i = 0; i < DIMENZIJA; i++) {
    printf("polje[%d] = ", i);
    scanf("%f", &polje[i]);
    if (i == 0) {
        max = polje[i];
    }
    if (max < polje[i]) {
        max = polje[i];
    }
}
printf("Najveci element polja je %f.\n\n", max);
for (i = 0; i < DIMENZIJA; i++) {
    polje[i] /= max;
    printf("polje[%d] = %f\n", i, polje[i]);
}
return 0;
}
```

23

## Primjer: Utvrđivanje frekvencije pojavljivanja brojeva prilikom učitavanja (1)

- Napisati program koji će učitavati prirodne brojeve u intervalu [10, 99] i brojati koliko puta je učitani broj. Učitavanje prekinuti kad se unese broj izvan zadanog intervala. Nakon učitavanja program treba ispisati koliko je puta učitani svaki broj iz zadanog intervala koji je učitani barem jednom.

```
#include <stdio.h>
#define DG 10 /* donja granica intervala */
#define GG 99 /* gornja granica intervala */
int main() {
    int broj, i;
    int brojac[GG - DG + 1] = { 0 };
```

24

## Primjer: Utvrđivanje frekvencije pojavljivanja brojeva prilikom učitavanja (2)

```
do {
    printf("\nUnesite broj u intervalu [%d, %d]: ",
           DG, GG);
    scanf("%d", &broj);
    if (broj >= DG && broj <= GG) {
        brojac[broj - DG]++;
    }
} while (broj >= DG && broj <= GG);

for (i = DG; i <= GG; i++) {
    if (brojac[i - DG] > 0) {
        printf("\nBroj %d se pojavio %d puta"
               , i , brojac[i - DG]);
    }
}
return 0;
}
```

25

## Naredba goto

Opći oblik naredbe:

```
goto oznaka_naredbe_1;
. . .
oznaka_naredbe_1:
    programski odsječak
```

26

## Naredba goto Primjer

- Napisati programski odsječak koji će učitavati pozitivne brojeve dok su manji ili jednaki 100. Ukoliko se unese negativni broj poduzeti odgovarajuće korake.

```
/* programska petlja za čitanje pozitivnih brojeva */
scanf("%d", &x);
while (x <= 100) {
    if (x < 0) goto pogreska;
    ...
    scanf("%d", &x);
}
/* odsječak koji rješava problem pogreške */
pogreska:
    printf("POGREŠKA - NEGATIVAN BROJ");
    ...
```

27

## Naredba goto i strukturirano programiranje Primjer neprihvatljive uporabe goto

```
for (i = 0; i < 10; i++) {
    programski odsječak;
}
↓
i = 0;
while (i < 10) {
    programski odsječak;
    i++;
}
↓
i = 0;
opet:
    if (i >= 10) goto dalje;
    programski odsječak;
    i++;
    goto opet;
dalje:
...

```

najbolje rješenje

prihvatljivo rješenje

neprihvatljivo rješenje

28

## Naredba `goto` i strukturirano programiranje Primjer prikladan za uporabu `goto`

```
for (...; uvjet1; ...) {
    while (uvjet2) {
        do {
            ...
            if (uvjet) nastaviti s odsječkom X;
            ...
        } while (uvjet3);
    }
}
odsječak X;
```

29

## Naredba `goto` i strukturirano programiranje Primjer korektne uporabe `goto`

```
for (...; uvjet1; ...) {
    while (uvjet2) {
        do {
            ...
            if (uvjet) goto van;
            ...
        } while (uvjet3);
    }
}
van:
odsječak X;
```

30

## Naredba `goto` i strukturirano programiranje Rješenje bez `goto`

```
int gotovo = 0;
for (...; uvjet1 && !gotovo; ...) {
    while (uvjet2 && !gotovo) {
        do {
            ...
            if (uvjet) {
                gotovo = 1;
                break;
            }
            ...
        } while (uvjet3);
    }
}
van:
odsječak X;
```

31

## Podsjetnik na definiciju dvodimenzionalnog polja

- Definicija 2D polja u C-u:  
`int x[3][2]` – matrica 3X2 ( 3 retka i 2 stupca ) čiji su elementi cijeli brojevi  
`char znakovi[2][4]` – polje znakova 2 retka i 4 stupca  
`float niz[MAXRED][MAXSTUP]` – MAXRED i MAXSTUP su konstante; matrica MAXRED X MAXSTUP
- Deklaracija višedimenzionalnog polja  
`int x[3][2][4]` 3D polje cijelih brojeva  
`float x[3][2][4][1]` 4D polje realnih brojeva

32

### Primjer određivanja najvećeg člana polja u svakom retku

- Pročitati vrijednosti za broj redaka  $mr \leq 100$  i broj stupaca  $ms \leq 10$ . Kontrolirati jesu li pročitane vrijednosti unutar dozvoljenog intervala. Pročitati vrijednosti članova dvodimenzionalnog realnog polja od  $mr$  redaka i  $ms$  stupaca. Odrediti u svakom retku najveći član i ispisati njegovu poziciju i vrijednost.

*Pseudokôd:*

```
{ Program za pronalaženje najvećih članova u retcima dvodimenzionalnog polja }
  učitaj mr i ms dok ne budu ispravni
  učitaj i ispiši realno polje a od mr redaka i ms stupaca
  { ispis najvećih članova u retcima }
  ponavljaj za sve retke
    | pronađi najveći član u retku
    | ispiši njegovu poziciju i vrijednost
  kraj
```

33

### Primjer određivanja najvećeg člana polja u svakom retku - detaljniji pseudokôd:

```
{učitavanje mr i ms dok ne budu ispravni}
ponavljaj
  | ispiši("Upisite vrijednost za broj redaka : ")
  | učitaj(mr)
dok je (mr < 1) ∨ (mr > NR)
ponavljaj
  | ispiši("Upisite vrijednost za broj stupaca : ")
  | učitaj(ms)
dok je (ms < 1) ∨ (ms > NS)
```

34

### Primjer određivanja najvećeg člana polja u svakom retku - detaljniji pseudokôd

```
{ ispis najvećih članova u retcima }
ispisi ("Najveci clanovi polja u retcima: ")
{ ponavljaj za sve retke }
za i := 0 do mr-1
  najveci = ai,0
  pozicija = 0
  za j := 1 do ms-1
    | ako je ai,j > najveci
    |   najveci := ai,j
    |   pozicija = j
  { ispis pozicije i vrijednosti najvećeg člana polja }
  ispiši(i+1, pozicija+1, ai, pozicija) ⇔ ispiši(i+1, pozicija+1, najveci)
```

35

### Primjer određivanja najvećeg člana polja u svakom retku - upute pretprocesoru, definicije

```
NajveciClanoviPoRetcima
/* Program za pronalazenje najvećih članova
   u retcima dvodimenzionalnog polja */
#include <stdio.h>
#define NR 100
#define NS 10

int main(){
  int mr, ms, i, j, pozicija;
  float najveci, a[NR][NS];
```

36

### Primjer određivanja najvećeg člana polja u svakom retku - učitavanje broja redaka i stupaca

```
/* Učitavanje mr i ms dok ne budu ispravni */
do {
    printf("Upisite vrijednost za broj redaka:");
    scanf("%d", &mr);
} while (mr < 1 || mr > NR);

do {
    printf("Upisite vrijednost za broj stupaca:");
    scanf("%d", &ms);
} while (ms < 1 || ms > NS);
```

37

### Primjer određivanja najvećeg člana polja u svakom retku - učitavanje i kontrolni ispis polja

```
/* učitaj polje a od mr redaka i ms stupaca */
printf("Unesite vrijednosti članova za %d",mr);
printf(" redaka i %d stupaca\n",ms);
for (i = 0; i < mr; i++) {
    for (j = 0; j < ms; j++) {
        scanf("%f", &a[i][j]);
    }
}
/* Ispiši polje a */
for (i = 0; i < mr; i++) {
    for (j = 0; j < ms; j++) {
        printf("%f ", a[i][j]);
    }
    printf ("\n");    /*prijelaz u novi red*/
}
```

38

### Primjer određivanja najvećeg člana polja u svakom retku - traženje najvećeg člana i ispis

```
/*Ispis najvećih članova u retcima */
printf("Najveci članovi polja u retcima:\n");
/* Petlja po svim retcima */
for (i = 0; i < mr; i++) {
    najveći = a[i][0];
    pozicija = 0;
    for (j = 1; j < ms; j++) {
        if (a[i][j] > najveći) {
            najveći = a[i][j];
            pozicija = j;
        }
    }
    /*Ispis pozicije i vrijednosti nadjenog člana*/
    printf("a(%d,%d) = %f\n", i+1, pozicija+1, a[i][pozicija]);
}
return 0;
}
```

39

### Primjer određivanja najvećeg člana polja u svakom retku - primjer izvršenja programa

```
Upisite vrijednost za broj redaka : 3
Upisite vrijednost za broj stupaca: 3
Unesite vrijednosti članova za 3 redaka i 3 stupaca
1 2 3
-0.99 -1 -5
0 10 5.5
1.000000 2.000000 3.000000
-0.990000 -1.000000 -5.000000
0.000000 10.000000 5.500000
Najveci članovi polja u retcima:
a(1,3) = 3.000000
a(2,1) = -0.990000
a(3,2) = 10.000000
```

40

## Primjer traženja najmanjeg elementa na glavnoj i sporednoj dijagonali (1)

- Napisati program koji će učitati realnu matricu dimenzija 10x10 te naći najmanji element na glavnoj i najmanji na sporednoj dijagonali.

```
#include <stdio.h>
#define BR_RED 10
#define BR_STUP 10
int main() {
    int i, j;
    float mat[BR_RED][BR_STUP], min_gl, min_sp;
    printf("Unos elemenata matrice :");
    for (i = 0; i < BR_RED; i++) {
        for (j = 0; j < BR_STUP; j++) {
            printf("\nUnesite element [%d][%d] : ", i, j);
            scanf("%f", &mat[i][j]);
        }
    }
}
```

41

## Primjer traženja najmanjeg elementa na glavnoj i sporednoj dijagonali (2)

```
min_gl = mat[0][0]; //minimum je clan mat(0,0) pa krece od 1
for (i = 1; i < BR_RED; i++) {
    if (mat[i][i] < min_gl) {
        min_gl = mat[i][i];
    }
}
min_sp = mat[0][BR_STUP-1];
for (i = 1; i < BR_RED; i++) {
    if (mat[i][BR_STUP-i-1] < min_sp) {
        min_sp = mat[i][BR_STUP-i-1];
    }
}
printf("\nNajmanji element na glavnoj dijagonali je : %f",
        min_gl);
printf("\nNajmanji element na sporednoj dijagonali je : %f",
        min_sp);
return 0;
}
```

42

## Primjer traženja najmanjeg elementa na glavnoj i sporednoj dijagonali – Skrraćena varijanta sa samo jednim prolaskom kroz matricu

```
#include <stdio.h>
#define BR_RED 10
#define BR_STUP 10
int main() {
    int i, j;
    float mat[BR_RED][BR_STUP], min_gl, min_sp;
    printf("\nUnos elemenata matrice :\n");
    for (i = 0; i < BR_RED; i++) {
        for (j = 0; j < BR_STUP; j++) {
            printf("\nUnesite element [%d][%d] : ", i, j);
            scanf("%f", &mat[i][j]);
            if (i == 0 && j == 0) {
                min_gl = mat[i][j];
            }
            if (i == 0 && j == BR_STUP - 1) {
                min_sp = mat[i][j];
            }
        }
    }
}
```

43

## Skrraćena varijanta sa samo jednim prolaskom kroz matricu -nastavak

```
if (i == j) {
    if (mat[i][j] < min_gl) {
        min_gl = mat[i][j];
    }
}
if (i == BR_STUP - 1 - j) {
    if (mat[i][j] < min_sp) {
        min_sp = mat[i][j];
    }
}
}
printf("\nNajmanji element na glavnoj dijagonali je : %f",
        min_gl);
printf("\nNajmanji element na sporednoj dijagonali je : %f",
        min_sp);
return 0;
}
```

44

## Primjer transponiranja matrice (1)

- Napisati program za transponiranje matrice. Potrebno je učitati dimenzije matrice i sve elemente matrice. Transponirana matrica je ona kojoj se zamijene reci i stupci.

```
#include <stdio.h>
#include <conio.h>
#define MAX_RED 50
#define MAX_STUP 50
int main() {
    int i, j, m, n, pom, max;
    int mat[MAX_RED][MAX_STUP];
    /* varijabla dim postavlja se na manju vrijednost od mogućih
    maksimalnih broja redaka i stupaca */
    int dim = (MAX_RED < MAX_STUP)? MAX_RED : MAX_STUP;
    /*ucitavanje velicine matrice */
    do {
        printf("Upisite vrijednost za broj redaka < %d:", dim);
        scanf("%d", &m);
        printf("Upisite vrijednost za broj stupaca < %d:", dim);
        scanf("%d", &n);
    } while (m < 1 || m > dim || n < 1 || n > dim);
```

45

## Primjer transponiranja matrice (2)

```
// ucitavanje elemenata matrice
printf("\nUnos elemenata matrice :\n");
for (i = 0; i < m; i++) {
    for (j = 0; j < n; j++) {
        printf("Unesite element [%d][%d] : ", i, j);
        scanf("%d", &mat[i][j]);
    }
}
// ispis prije transponiranja
printf("\n\nIspis matrice prije transponiranja:\n");
for (i = 0; i < m; i++) {
    for (j = 0; j < n; j++) {
        printf("%3d", mat[i][j]);
    }
    printf("\n");
}
```

46

## Primjer transponiranja matrice (3)

```
max=m>n ? m : n;
// transponiranje
for ( i=0; i<max; ++i ) {
    for ( j=i+1; j<max; ++j ) { // petlja mora ici od i+1 !!
        pom = mat[i][j];
        mat[i][j] = mat[j][i];
        mat[j][i] = pom;
    }
}
// ispis nakon transponiranja
// broj redaka je sada broj stupaca ...
printf("\n\nIspis matrice nakon transponiranja:\n");
for (i = 0; i < n; i++) {
    for (j = 0; j < m; j++) {
        printf("%3d", mat[i][j]);
    }
    printf("\n");
}
return 0;
} // main
```

47

## Primjer izvođenja

```
Upisite vrijednost za broj redaka < 50: 3
Upisite vrijednost za broj stupaca < 50: 2
Unos elemenata matrice :
Unesite element [0][0] : 1
Unesite element [0][1] : 2
Unesite element [1][0] : 3
Unesite element [1][1] : 4
Unesite element [2][0] : 5
Unesite element [2][1] : 6
```

```
Ispis matrice prije transponiranja :
1 2
3 4
4 5 6
```

```
Ispis matrice nakon transponiranja :
1 3 5
2 4 6
```

48

## Primjer izračunavanja suma članova stupaca i produkata članova redaka (1)

- Napisati program koji će učitati realnu matricu dimenzija 10x10 te naći sume elemenata svakog stupca i produkte elemenata svakog retka. Ispisati najmanju sumu i pripadni indeks stupca te najveći produkt i pripadni indeks retka. Sume i produkte čuvati u jednodimenzionalnim poljima.

```
#include <stdio.h>
#define BR_RED 10
#define BR_STUP 10
int main() {
    int i, j;
    int min_sum_ind, max_prod_ind;
    float mat[BR_RED][BR_STUP];
    float sum[BR_STUP], prod[BR_RED];
```

49

## Izračunavanje suma članova stupaca i produkata članova redaka (2)

```
/* 1.varijanta unosa elemenata i racunanja suma i produkata */
for (i = 0; i < BR_RED; i++) {
    for (j = 0; j < BR_STUP; j++) {
        printf("\nUnesite element [%d][%d] : ", i, j);
        scanf("%f", &mat[i][j]);
    }
}

for (j = 0; j < BR_STUP; j++) {
    sum[j] = 0;
    for (i = 0; i < BR_RED; i++) {
        sum[j] += mat[i][j];
    }
}

for (i = 0; i < BR_RED; i++) {
    prod[i] = 1;
    for (j = 0; j < BR_STUP; j++) {
        prod[i] *= mat[i][j];
    }
}
/* kraj unosa elemenata i racunanja suma i produkata */
```

50

## Izračunavanje suma članova stupaca i produkata članova redaka (3)

```
/* naci indeks stupca za najmanju sumu */
min_sum_ind = 0;
for (j = 1; j < BR_STUP; j++) {
    if (sum[j] < sum[min_sum_ind]) {
        min_sum_ind = j;
    }
}

/* naci indeks retka za najveći produkt */
max_prod_ind = 0;
for (i = 1; i < BR_RED; i++) {
    if (prod[i] > prod[max_prod_ind]) {
        max_prod_ind = i;
    }
}
printf("\nNajmanja suma je %f , a pripadni indeks je %d\n",
        sum[min_sum_ind], min_sum_ind);
printf("\nNajveći produkt je %f , a pripadni indeks je %d\n",
        prod[max_prod_ind], max_prod_ind);
return 0;
}
```

51

## Izračunavanje suma članova stupaca i produkata članova redaka (4)

- Odsječak skraćene varijanta učitavanja elemenata matrice i računanja suma i produkata:

```
/* 2. varijanta unosa elemenata i racunanja
suma i produkata */
for (i = 0; i < BR_RED; i++) {
    prod[i] = 1;
    for (j = 0; j < BR_STUP; j++) {
        printf("\nUnesite element [%d][%d] : ", i, j);
        scanf("%f", &mat[i][j]);
        prod[i] *= mat[i][j];
        if (i == 0) {
            sum[j] = 0;
        }
        sum[j] += mat[i][j];
    }
}
/* kraj unosa elemenata i racunanja suma i produkata */
```

52