
Programiranje

podatkovna struktura **polje**

Zadatak: Pročitati s tipkovnice 10 cijelih brojeva te ispisati njihov prosjek (aritmetičku sredinu)

- rješenje je jednostavno i glasi:

```
...
int i, broj, suma = 0;
for (i = 1; i <= 10; i++) {
    scanf("%d", &broj);
    suma = suma + broj;
}
printf("Prosjek je: %f\n", suma / 10.);
...
```

Teži zadatak: pročitati s tipkovnice 10 cijelih brojeva i uz svaki od učitanih brojeva ispisati: **broj je \geq od prosjeka** ili **broj je $<$ od prosjeka**

- Dok se ne učitaju svi brojevi, ne može se izračunati prosjek. Kada se prosjek uspije izračunati, više nije poznato koji su sve brojevi bili učitani!
- Očito, **sve** učitane brojeve treba "pamtiti" dok se ne izračuna prosjek, a tada treba ispisivati brojeve i uz svaki broj ispisati odgovarajuću poruku.
- Ocijenite moguće rješenje:
 - definirati varijable **broj1, broj2, ..., broj10,**
 - u njih učitati vrijednosti,
 - izračunati prosjek i nakon toga za svaku vrijednost varijabli **broj1** do **broj10** ispisati odgovarajući tekst?

Pokušaj rješavanja na opisani način:

```
#define VECI "broj je >= od prosjeka"
#define MANJI "broj je < od prosjeka"
...
int suma = 0, broj1, broj2, ..., broj10;
float prosjek;
scanf("%d", &broj1); suma = suma + broj1;
scanf("%d", &broj2); suma = suma + broj2;
... itd ...
prosjek = suma / 10.;
/* ispis */
printf("%d %s\n", broj1, broj1 < prosjek ? MANJI:VECI);
printf("%d %s\n", broj2, broj2 < prosjek ? MANJI:VECI);
... itd ...
```

Što ako se zadatak malo oteža: umjesto 10, potrebno je učitati n brojeva pri čemu vrijedi $1 < n \leq 50$. Pokušajte napisati takav program! Takvo rješenje očito nije dobro.

Matematički niz

- Niz brojeva koji dijele isto ime, ali se razlikuju po indeksu: broj₀, broj₁, broj₂... Ne bi li bilo lijepo kad bi se prethodni zadatak mogao riješiti na sljedeći način:

```
definiraj pomoćne varijable
```

```
definiraj niz broj
```

```
za i = 1 do 10
```

```
    |   učitaj(broji)
```

```
    |   suma := suma + broji;
```

```
prosjek := suma / 10.;
```

```
za i = 1 do 10
```

```
    |   ispiši (broji s odgovarajućom napomenom)
```

Sada više nije važno učitava li se 10, 100 ili 1000 brojeva

Podatkovna struktura polje

- Do sada su korišteni jednostavni tipovi podataka: int, float, char, ... (izuzetak su bili nizovi znakova). U varijablama jednostavnog tipa moguće je pohraniti samo po jedan podatak:

x

| |
|---------|
| 3.14159 |
|---------|

x → 3.14159 x je L-value

- Polje je podatkovna struktura ili složeni tip podatka (*data aggregate*) koji obuhvaća više članova:

y

| | | | | |
|---------|--------|----------|----------|-------|
| 3.14159 | 2.7182 | 8.85e-12 | 6.67e-12 | 8.314 |
|---------|--------|----------|----------|-------|

- Svakom pojedinom članu polja **y** može se pristupiti pomoću indeksa:

y[0] → 3.14159 y[0], y[1], ... su L-values
y[2] → 8.85e-12

Definicija varijabli tipa polje

- Potrebno je definirati
 - ime varijable
 - definira se na isti način kao za jednostavne tipove podataka
 - tip podatka za članove polja (`int`, `float`, ...)
 - svi članovi jednog polja moraju biti istog tipa
 - broj članova polja
 - u uglatim zagradama, cjelobrojni konstantni izraz, koji sadrži konstante i/ili simboličke konstante

```
#define MAX 100
```

```
#define MAX_T 80
```

```
int x[20];
```

```
char znakovi[5*80];
```

```
float niz[MAX];
```

```
char tekst[MAX_T+1]
```

polje od 20 cijelih brojeva (članova)

polje od 400 znakova

polje od 100 realnih brojeva

polje od 81 znaka

Pristupanje članovima polja

- Članovima polja se pristupa koristeći indeks, koji mora biti nenegativni cijeli broj (konstanta, varijabla, cjelobrojni izraz).

`x[0]` `x[n]` `x[MAX]` `x[n+1]` `x[k/m+5]`

- Indeks člana (elementa) je broj između 0 i broja elemenata minus jedan, uključivo, tj.

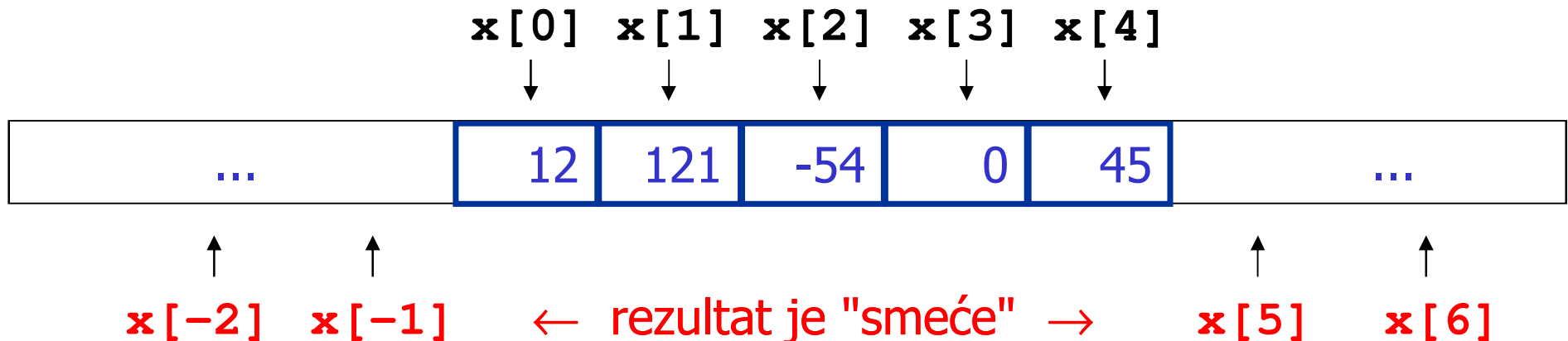
`indeks` \in `[0, BrojElemenataPolja - 1]`.

Smještaj polja u memoriji računala

- Članovi polja susjedni po indeksu susjedni su i u memoriji računala

```
int x[5];
```

```
x[0]=12; x[1]=121; x[2]=-54; x[3]=0; x[4]=45;
```



Pristupanje članovima polja: moguće pogreške

Ispravno pristupanje članovima (elementima) polja:

```
#define MAX 100
```

```
int x[MAX]
```

`x[0]` prvi element polja

`x[i]` (i+1). element polja, uz $0 \leq i \leq \text{BrojElementata} - 1$

`x[MAX - 1]` posljednji element polja

Neispravno pristupanje elementima polja:

```
int polje[10];
```

```
int x = polje[10];
```

 11. član, a polje nema 11 članova
→ rezultat je "smeće"

```
float a = 1.;
```

```
int x = polje[a];
```

 indeks mora biti cjelobrojan
→ prevodilac: pogreška

```
int a = 1, b = 0, c = 1;
```

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

 ne postoji član s indeksom -1
→ rezultat je "smeće"

Primjer: pročitati s tipkovnice 10 cijelih brojeva i uz svaki od učitanih brojeva ispisati: broj je \geq od prosjeka ili broj je $<$ od prosjeka

```
#define VECI "broj je  $\geq$  od prosjeka"
#define MANJI "broj je  $<$  od prosjeka"

...
int i, broj[10], suma=0;
float prosjek;
for (i = 0; i < 10; i++) {
    scanf("%d", &broj[i]);
    suma = suma + broj[i];
}
prosjek = suma / 10.;
for (i = 0; i < 10; i++) {
    printf("%d %s\n",
           broj[i],
           broj[i] < prosjek ? MANJI : VECI);
}
...
```

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

```
int broj[20];
```

vrijednosti članova ovog polja su trenutno nedefinirane ("smeće")

```
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] = 0.0;      boja[0] = 'C';
```

```
znam[1] = 2;      x[1] = 0.25;     boja[1] = 'P';
```

```
znam[2] = 3;      x[2] = 0.0;      boja[2] = 'Z';
```

```
znam[3] = 4;      x[3] = -0.5;
```

```
znam[4] = 5;      x[4] = 0.0;
```

```
znam[5] = 6;      x[5] = 0.0;
```

```
znam[6] = 7;
```

```
znam[7] = 8;
```

```
znam[8] = 9;
```

```
znam[9] = 10;
```

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] = 0.0;
    znam[2] = 3;           x[2] = 0.25;
    znam[3] = 0;           x[3] = 0.0;
    znam[4] = 0;           x[4] = 0.0;
    znam[5] = 0;           x[5] = 0.0;
    znam[6] = 0;
    znam[7] = 0;
    znam[8] = 0;
    znam[9] = 0;
```

Članovi za koje nije navedena početna vrijednost postavljaju se na vrijednost 0.

- Kako najlakše inicijalizirati članove polja na 0?

```
int broj[2000] = {0};
```

Dodjeljivanje početnih vrijednosti članovima polja: česte pogreške

- Ako se polju pridjeljuju početne vrijednosti, u vitičastim zagradama se mora nalaziti barem jedna vrijednost. Nije dopušteno:

```
float x[5] = {};
```

→ prevodilac: pogreška

- Broj početnih vrijednosti ne smije biti veći od deklariranog broja članova polja. Nije dopušteno:

```
int broj[5] = {3, 7, 11, 121, 12, 10};
```

→ prevodilac: pogreška

Dodjeljivanje početnih vrijednosti članovima polja: broj članova polja može biti definiran brojem navedenih konstanti

```
int znam[] = {1, 2, 3, 4, 5, 6};
```

```
isto kao: int znam[6] = {1, 2, 3, 4, 5, 6};
```

```
float x[] = {0, 0.25, 0, -0.5};
```

```
isto kao: float x[4] = {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;
```

Dodjeljivanje početnih vrijednosti članovima polja: primjeri s poljima znakova

```
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';
```

Polje znakova kao niz znakova (*string*)

Podsjetnik: konstanta se u C-u piše unutar dvostrukih navodnika:

"Ovo je niz"



Što je s varijablama? Ne postoji tip podatka *string*. Koristi se jednodimenzionalno polje znakova:

```
char ime[4+1];  
ime[0] = 'I';  
ime[1] = 'v';  
ime[2] = 'a';  
ime[3] = 'n';  
ime[4] = '\0';
```



```
printf("%s", ime);
```

ispisat će se:

Ivan

Polje znakova kao niz znakova (*string*)

Čemu služi \0



Pomoću \0 se može zaključiti gdje je kraj niza znakova.

```
char ime[4];  
ime[0] = 'I';  
ime[1] = 'v';  
ime[2] = 'a';  
ime[3] = 'n';
```



Što će sada ispisati naredbom `printf("%s", ime)`

Ivan*) %&/!) = () Z) (B#DW=) (@ (\$/") #*' @! / ["&/\$/"/...

... i nastaviti će se ispisivati dok se ne nađe na oktet u kojem je upisana vrijednost `0x00` (tj. `'\0'`)

Pridjeljivanje početnih vrijednosti nizu znakova

Jednako kao kod polja ostalih tipova podataka:

```
char ime[4+1] = {'I', 'v', 'a', 'n', '\0'};
```

ili

```
char ime[4+1] = {'I', 'v', 'a', 'n'};
```

ili

```
char ime[] = {'I', 'v', 'a', 'n', '\0'};
```

Još jednom primijetite da sljedeće polje nije dobro inicijalizirano ukoliko se namjerava koristiti kao niz znakova:

```
char ime[] = {'I', 'v', 'a', 'n'};
```

Bolji način inicijalizacije char polja koje se koristi za pohranu niza znakova

Umjesto

```
char ime[5] = {'I', 'v', 'a', 'n', '\0'};
```

može se (i bolje je!) koristiti

```
char ime[5] = "Ivan";
```

znak \0 će biti dodan, osigurati prostor za barem jedan znak više!

ili

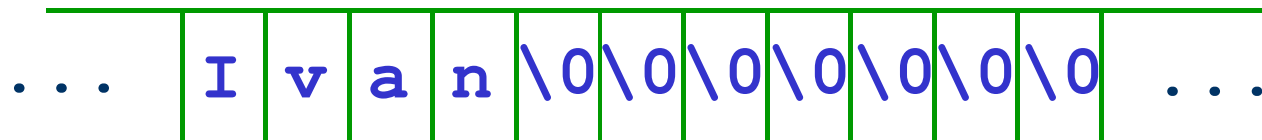
```
char ime[] = "Ivan";
```

znak \0 će biti dodan

Uobičajeno se varijable definiraju uz pomoć simboličkih konstanti

```
#define MAX_IME 10
```

```
char ime[MAX_IME + 1] = "Ivan";
```



Primjer: učitaj ime studenta (sigurno nije dulje od 20 znakova), sumiraj ASCII vrijednosti svih znakova iz imena studenta, ispiši ime studenta i dobivenu sumu

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

```
#define MAX_IME 20
```

```
int main() {
```

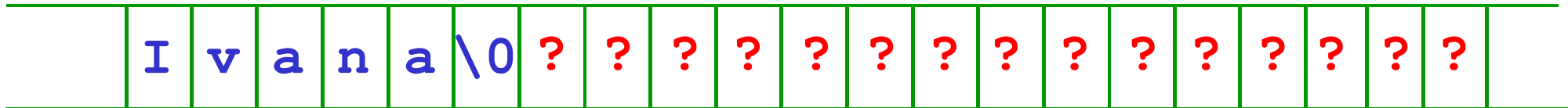
```
    char ime[MAX_IME + 1];
```

```
    int i = 0, suma = 0;
```

```
    gets(ime);
```

def. polja, nije dulje od 20 znakova

pročita niz znakova, spremi ga u ime, doda \0 na kraj



```
    while (ime[i] != '\0')
```

```
        suma = suma + ime[i++];
```

```
    printf("%s %d", ime, suma);
```

```
    return 0;
```

```
}
```

Korisnik upiše npr.

Ivana

ispisat će se:

Ivana 495

Primjer: izračunavanje aritmetičke sredine "nenultih" članova polja

- Uz kontrolu učitati broj $1 \leq n \leq 100$. Učitati 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 arit.sredinu
```

```
inače
```

```
    | postavi arit.sredinu na nulu
```

```
ispiši članove polja
```

```
ispiši sredinu
```

```
kraj
```

Izračunavanje aritmetičke 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
```

Izračunavanje aritmetičke sredine "nenultih" članova polja: rješenje u C-u (početak programa)

📁 AritmetickaSredinaPolja

```
/* Program za racunanje aritmeticke sredine */
```

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

```
↔ #define DIM 100
```

```
int main() {
```

```
    int k[100];
```

```
↔ int k[DIM];
```

```
    int n, brojac, i, suma;
```

```
    float sredina;
```

```
    do {
```

```
        printf("Unesite broj clanova polja : ");
```

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

```
    } while (n < 1 || n > 100);
```

```
    for (i=0; i<n; i=i+1){
```

```
↔ for (i=0; i<n; i++) {
```

```
        scanf("%d", &k[i]);
```

```
    }
```

Izračunavanje aritmetičke sredine "nenultih" članova polja: rješenje u C-u (nastavak programa)

```
suma = 0;
```

```
brojac = 0;
```

```
/* izracunaj sumu i brojac nenultih clanova */
```

```
for (i=0; i<n; i=i+1){           ⇐ for (i=0;i<n;i++) {
```

```
    if (k[i] != 0) {
```

```
        suma = suma + k[i];     ⇐ suma += k[i];
```

```
        brojac = brojac + 1;    ⇐ ++brojac;
```

```
    }
```

```
}
```

Izračunavanje aritmetičke sredine "nenultih" članova polja: rješenje u C-u (kraj programa)

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

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

Primjer izračunavanja aritmetičke sredine (1)

- Napisati program koji će tražiti unos niza od 10 realnih brojeva, nakon čega će izračunati aritmetičku sredinu članova tog niza, te najprije ispisati članove 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., ar_sred, niz[DIMENZIJA] = {0};
    for (i = 0; i < DIMENZIJA; i++) {
        printf("Unesi broj: ");
        scanf("%f", &niz[i]);
        suma += niz[i];
    }
    ar_sred = suma / DIMENZIJA;
    printf("Aritm. sredina niza je %6.2f.\n", ar_sred);
```

treba li ovo?



Primjer izračunavanja aritmetičke sredine (2)

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

/* Sto ako je broj točno jednak ar_sred? */

return 0;
}
```

Primjer dijeljenja učitanih članova polja najvećim članom niza (1)

- Napisati program u kojem će se unijeti 10 realnih brojeva. Nakon unosa, sve članove niza podijeliti s najvećim članom niza, i iskazati ih relativno u odnosu na najveći član. Npr.

Unos podataka:

```
polje[0] = 1.0
polje[1] = 3.0
polje[2] = 5.0
polje[3] = 7.0
polje[4] = 9.0
polje[5] = 8.0
polje[6] = 6.0
polje[7] = 4.0
polje[8] = 2.0
polje[9] = 0.0
```

Ispis:

```
Najveci element polja je 9.000000.
polje[0] = 0.111111
polje[1] = 0.333333
polje[2] = 0.555556
polje[3] = 0.777778
polje[4] = 1.000000
polje[5] = 0.888889
polje[6] = 0.666667
polje[7] = 0.444444
polje[8] = 0.222222
polje[9] = 0.000000
```

Primjer dijeljenja učitanih članova polja najvećim članom niza (2)

```
#include <stdio.h>
#define DIMENZIJA 10
int main() {
    int i;
    float max, polje[DIMENZIJA];
    for (i = 0; i < DIMENZIJA; i++) {
        printf("polje[%d] = ", i);
        scanf("%f", &polje[i]);
        if (i == 0) {
            max = polje[i];
        } ← else
        if (max < polje[i]) {
            max = polje[i];
        }
    }
}
```

Primjer dijeljenja učitanih članova polja najvećim članom niza (3)

```
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;
}
```

Primjer (varijanta a): Utvrđivanje frekvencije pojavljivanja brojeva prilikom učitavanja (1)

- Napisati program koji će učitavati prirodne brojeve u intervalu $[0, 9]$ i brojati koliko puta je učitani koji broj. Učitavanje prekinuti kad se unese broj izvan zadanog intervala. Nakon učitavanja program treba ispisati koliko je puta učitani svaki broj iz zadanog intervala (ispisati samo one brojeve koji su učitani barem jednom). Npr. ako se učitaju brojevi:

1 5 7 5 0 5 7 1 10

treba ispisati:

Broj 0 se pojavio 1 puta

Broj 1 se pojavio 2 puta

Broj 5 se pojavio 3 puta

Broj 7 se pojavio 2 puta

Primjer (varijanta a): Utvrđivanje frekvencije pojavljivanja brojeva prilikom učitavanja (2)

```
#include <stdio.h>
#define DG 0          /* donja granica intervala */
#define GG 9         /* gornja granica intervala */
int main() {
    int broj, i;
    int brojac[10] = { 0 };
    do {
        printf("\nUnesite broj u intervalu [%d, %d]: ",
                DG, GG);
        scanf("%d", &broj);
        if (broj >= DG && broj <= GG) {
            brojac[broj]++;
        }
    } while (broj >= DG && broj <= GG);
```

treba li ovo?



Primjer (varijanta a): Utvrđivanje frekvencije pojavljivanja brojeva prilikom učitavanja (3)

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

Primjer (varijanta b): Utvrđivanje frekvencije pojavljivanja brojeva prilikom učitavanja (1)

- **Varijanta prethodnog primjera: promijenjene su samo granice intervala**
- Napisati program koji će učitavati prirodne brojeve u intervalu [10, 99] i brojati koliko puta je učitani koji broj. Učitavanje prekinuti kad se unese broj izvan zadanog intervala. Nakon učitavanja program treba ispisati koliko je puta učitani svaki broj iz zadanog intervala (ispisati samo one brojeve koji su učitani barem jednom). Npr. ako se učitaju brojevi:

15 25 25 77 25 15 11 7

treba ispisati:

Broj 11 se pojavio 1 puta

Broj 15 se pojavio 2 puta

Broj 25 se pojavio 3 puta

Broj 77 se pojavio 1 puta

Primjer (varijanta b): Utvrđivanje frekvencije pojavljivanja brojeva prilikom učitavanja (2)

```
#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 };
    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);
```

treba li ovo?



Primjer (varijanta b): Utvrđivanje frekvencije pojavljivanja brojeva prilikom učitavanja (3)

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

Višedimenzionalna polja

- Jednodimenzionalno polje (vektor)

```
int a[5]
```

| | | | | |
|------|------|------|------|------|
| a[0] | a[1] | a[2] | a[3] | a[4] |
|------|------|------|------|------|

- Polje može imati više dimenzija
 - dvije dimenzije (matrica, tablica)

```
int b[3][5]
```

| | | | | |
|---------|---------|---------|---------|---------|
| b[0][0] | b[0][1] | b[0][2] | b[0][3] | b[0][4] |
| b[1][0] | b[1][1] | b[1][2] | b[1][3] | b[1][4] |
| b[2][0] | b[2][1] | b[2][2] | b[2][3] | b[2][4] |

1. redak

2. redak

3. redak

matrica od 3 retka i 5 stupaca

Višedimenzionalna polja

- Broj dimenzija nije ograničen npr.

```
double a[3][3][3][3][3][3][3][3][3][3][3][3][3][3];
```

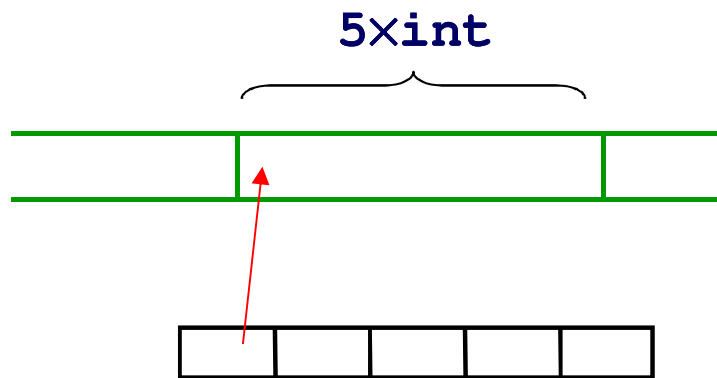
Oprez: veličina ovog polja je $8 * 3^{14} = 38\,263\,752$ bajta

Pri definiciji dimenzija polja uobičajeno je korištenje simboličkih konstanti:

```
#define MAXRED 10
#define MAXSTUP 20
...
int matrica[MAXRED][MAXSTUP];
```

Smještaj višedimenzionalnih polja u memoriji računala

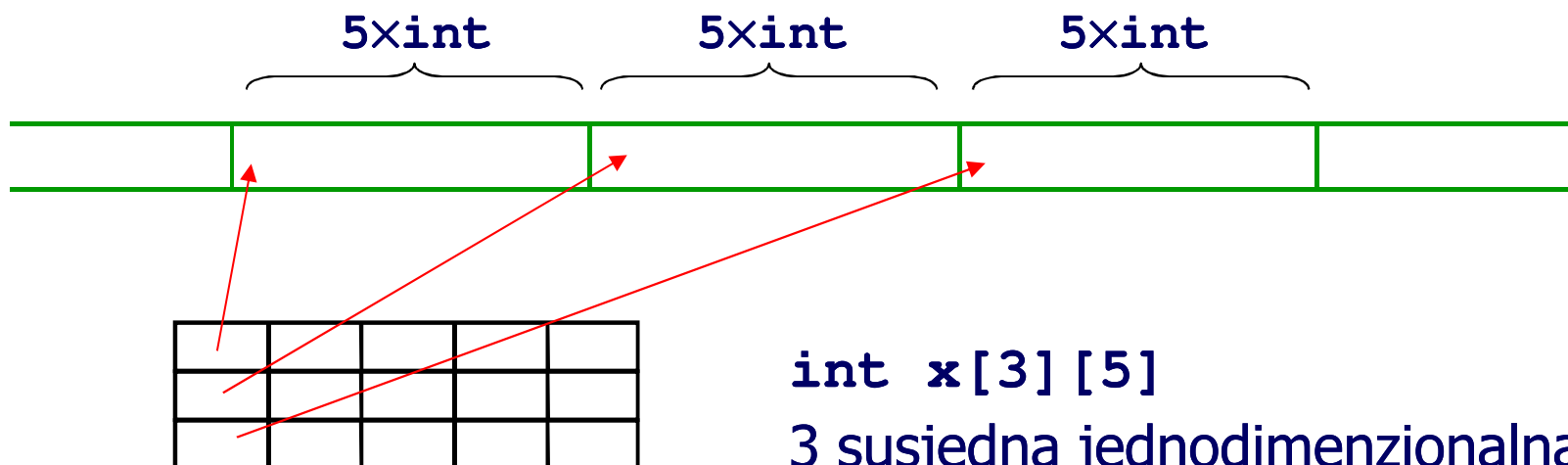
- Jednodimenzionalno polje: član za članom



`int x[5]`

5 susjednih cijelih brojeva

- Dvodimenzionalno polje: redak za retkom

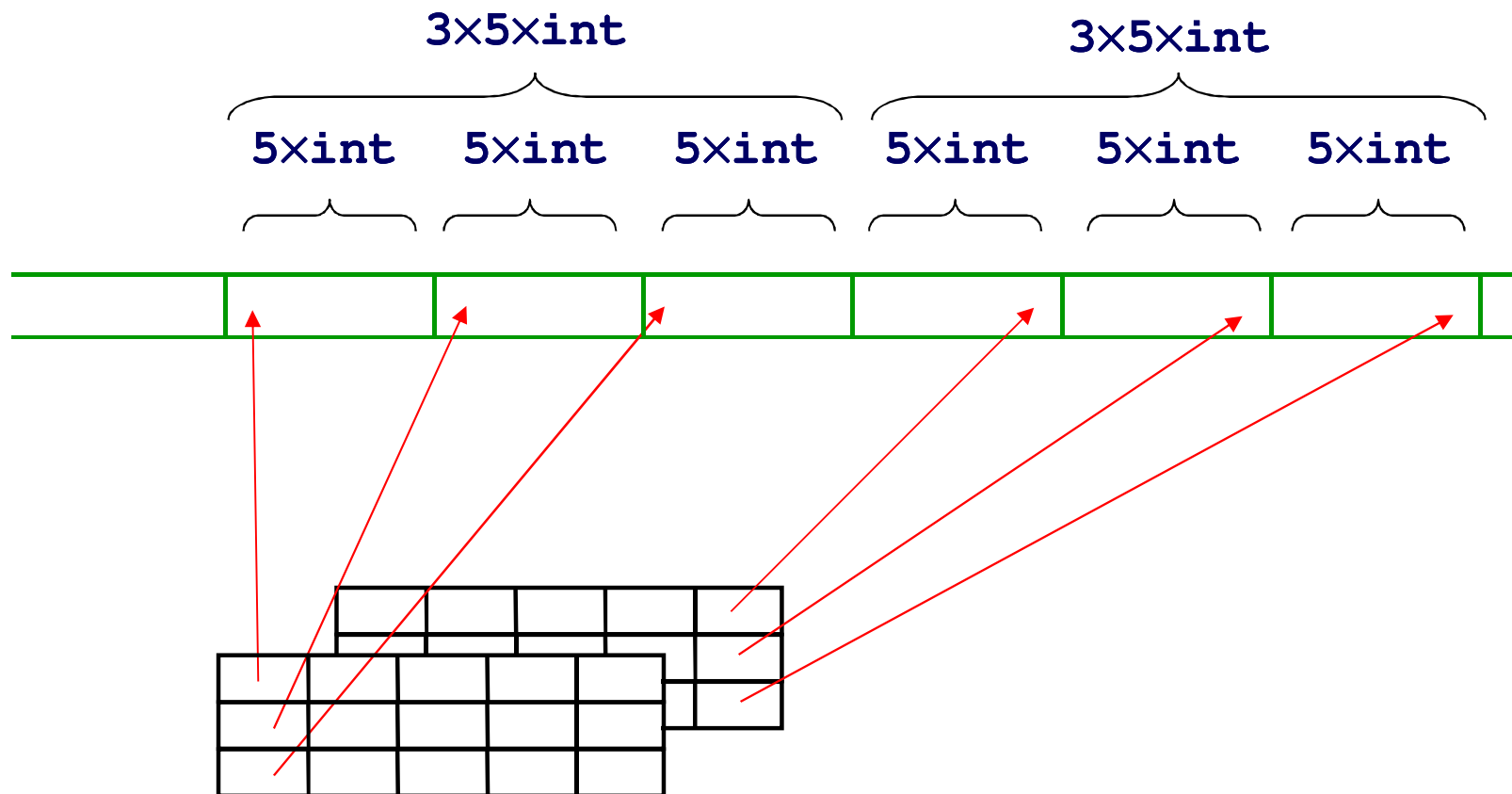


`int x[3][5]`

3 susjedna jednodimenzionalna polja veličine $5 \times \text{int}$

Smještaj višedimenzionalnih polja u memoriji računala

- Trodimenzionalno polje: sloj za slojem



`int x[2][3][5]`

2 susjedna dvodimenzionalna polja veličine 3×5

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
```

Bolje je početne vrijednosti zadati na sljedeći način:

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

Nije dopušteno:

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

→ prevodilac: pogreška. Kod višedimenzionalnih polja obje dimenzije moraju biti zadane

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
```

```
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
```

Dodjeljivanje početnih vrijednosti članovima polja: primjer gdje je navedeno previše vrijednosti

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

→ prevodilac: pogreška

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

→ prevodilac: pogreška

Operator sizeof i polja

- sizeof vraća ukupnu veličinu polja u oktetima (bajtovima)

```
int x[3][2];
```

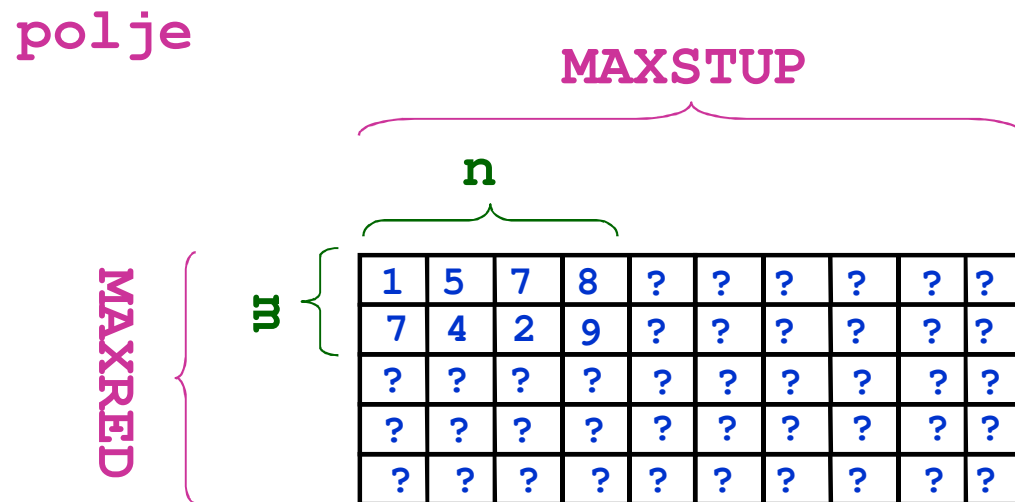
```
sizeof(x) → 24, tj. 6*sizeof(int)
```

```
char c[10][20][80];
```

```
sizeof(c) → 16000
```

Primjer učitavanja i ispisa matrice

- S tipkovnice pročitati m (broj redaka) i n (broj stupaca). Broj redaka ne smije biti veći od 5, a broj stupaca ne smije biti veći od 10. Ponavljati učitavanje m i n dok ne budu ispravni.
- Učitati $m \times n$ članova polja, a zatim ih ispisati u obliku dvodimenzionalne tablice



Primjer učitavanja i ispisa matrice (1)

```
#include <stdio.h>
#define MAXRED 5
#define MAXSTUP 10

int main() {
    int polje[MAXRED][MAXSTUP];
    int m, n, i, j;

    do {
        printf("Upisi m i n:");
        scanf("%d %d", &m, &n);
    } while (m < 1 || m > MAXRED || n < 1 || n > MAXSTUP);
```

Primjer učitavanja i ispisa matrice (2)

```
printf("Upisi članove: %d redaka i %d stupaca\n", m, n);  
for (i = 0; i < m; i++)  
    for (j = 0; j < n; j++)  
        scanf("%d", &polje[i][j]);  
  
printf("\n");  
for (i = 0; i < m; i++) {  
    for (j = 0; j < n; j++)  
        printf("%d ", polje[i][j]);  
    printf("\n");  
}  
return 0;  
}
```

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 recima dvodimenzionalnog polja }
  učitavaj 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
    |   pronadi najveći član u retku
    |   ispiši njegovu poziciju i vrijednost
  kraj
```

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

{učitavanje mr i ms dok ne budu ispravni}

ponavlja

ispiši("Upisite vrijednost za broj redaka : ")

učitaj(mr)

dok je $(mr < 1) \vee (mr > NR)$

ponavlja

ispiši("Upisite vrijednost za broj stupaca : ")

učitaj(ms)

dok je $(ms < 1) \vee (ms > NS)$

Primjer određivanja najvećeg člana polja u svakom retku - detaljniji pseudokôd

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

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 najveći, a[NR][NS];
```

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);
```

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]);
    }
}
/* Ispisi 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 */
}
```

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

```
/* Ispis najvećih članova u retcima */
printf("Najveći 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;
}
```

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 clanovi polja u retcima:

$a(1,3) = 3.000000$

$a(2,1) = -0.990000$

$a(3,2) = 10.000000$

Primjer traženja najmanjeg člana na glavnoj i sporednoj dijagonali (1)

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

pretpostavka: ovaj je najmanji

pretpostavka: ovaj je najmanji

| | | | | | | | | | |
|------|------|------|------|------|------|------|------|------|------|
| 0, 0 | | | | | | | | | 0, 9 |
| | 1, 1 | | | | | | | 1, 8 | |
| | | 2, 2 | | | | | 2, 7 | | |
| | | | 3, 3 | | | 3, 6 | | | |
| | | | | 4, 4 | 4, 5 | | | | |
| | | | | 5, 4 | 5, 5 | | | | |
| | | | 6, 3 | | | 6, 6 | | | |
| | | 7, 2 | | | | | 7, 7 | | |
| | 8, 1 | | | | | | | 8, 8 | |
| 9, 0 | | | | | | | | | 9, 9 |

```
mat[0][0]
za i=1; i < 10; i++
  mat[i][i]?
```

```
mat[0][10-1]
za i=1; i < 10; i++
  mat[i][10-1-i]?
```

Primjer traženja najmanjeg člana na glavnoj i sporednoj dijagonali (2)

```
#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]);
        }
    }
}
```

Primjer traženja najmanjeg elementa na glavnoj i sporednoj dijagonali (3)

```
min_gl = mat[0][0];
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;
}
```

Primjer traženja najmanjeg člana na glavnoj i sporednoj dijagonali – skraćena 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];
            }
        }
    }
}
```

Skraćena 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;
}
```

Primjer transponiranja matrice (1)

- Napisati program za transponiranje matrice. Potrebno je učitati dimenzije matrice i sve članove (elemente) matrice. Transponirana matrica je "ona kojoj se zamijene reci i stupci".
- član `mat [i] [j]` originalne matrice postaje član `mat [j] [i]` u transponiranoj matrici

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

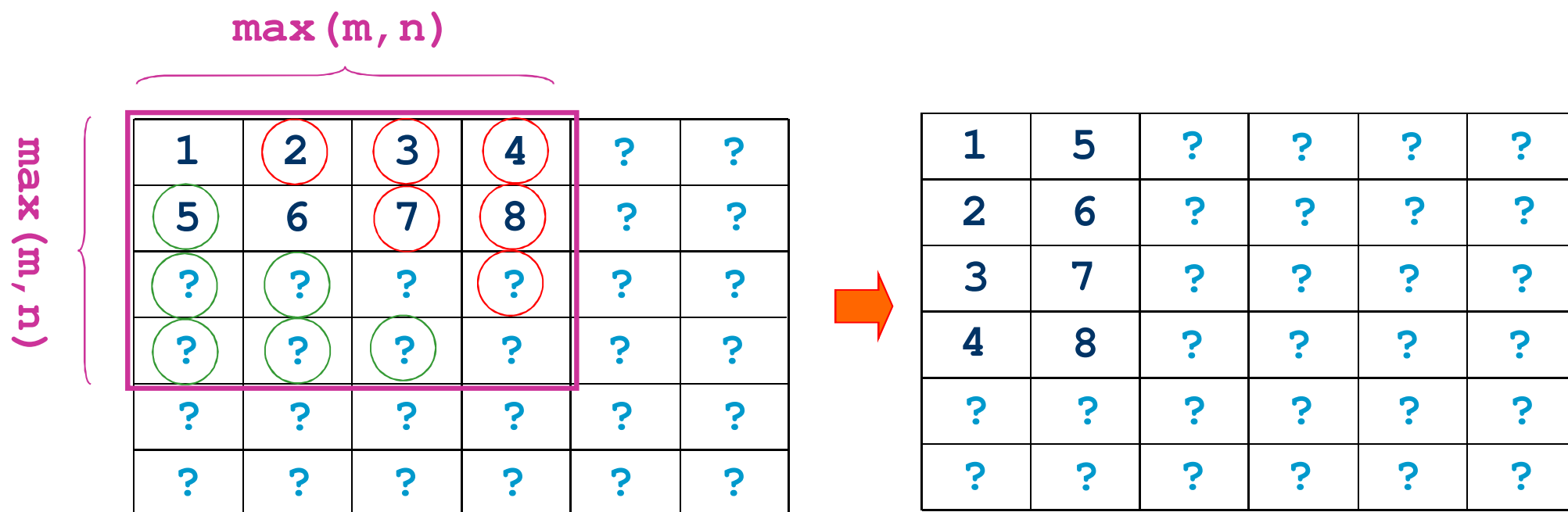
5 6

Ispis matrice nakon transponiranja :

1 3 5

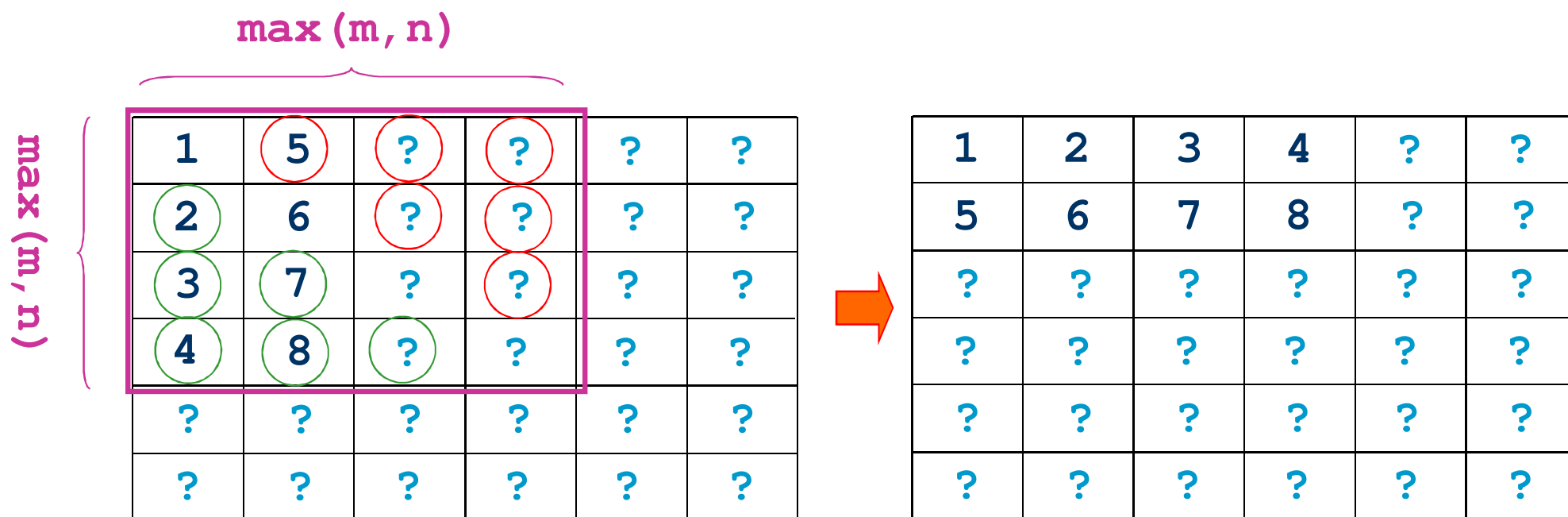
2 4 6

Kad ima više stupaca nego redaka:



- Član **mat[i][j]** originalne matrice zamjenjuje se s članom **mat[j][i]**
- indeks retka: $i = 0; i < \max(m, n) - 1; i++$
- indeks stupca: $j = i + 1; j < \max(m, n); j++$

Kad ima više redaka nego stupaca:



- Član **mat[i][j]** originalne matrice zamjenjuje se s članom **mat[j][i]**
- indeks retka: $i = 0; i < \max(m, n) - 1; i++$
- indeks stupca: $j = i + 1; j < \max(m, n); j++$

Primjer transponiranja matrice (2)

```
#include <stdio.h>
#define MAXDIM 50
int main() {
    int i, j, m, n, pom, max;
    int mat[MAXDIM][MAXDIM];

    /*ucitavanje velicine matrice */
    do {
        printf("Upisite vrijednost za broj redaka < %d:", MAXDIM);
        scanf("%d", &m);
        printf("Upisite vrijednost za broj stupaca < %d:", MAXDIM);
        scanf("%d", &n);
    } while (m < 1 || m > MAXDIM || n < 1 || n > MAXDIM);
```

Primjer transponiranja matrice (3)

```
/* 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]);
    }
}

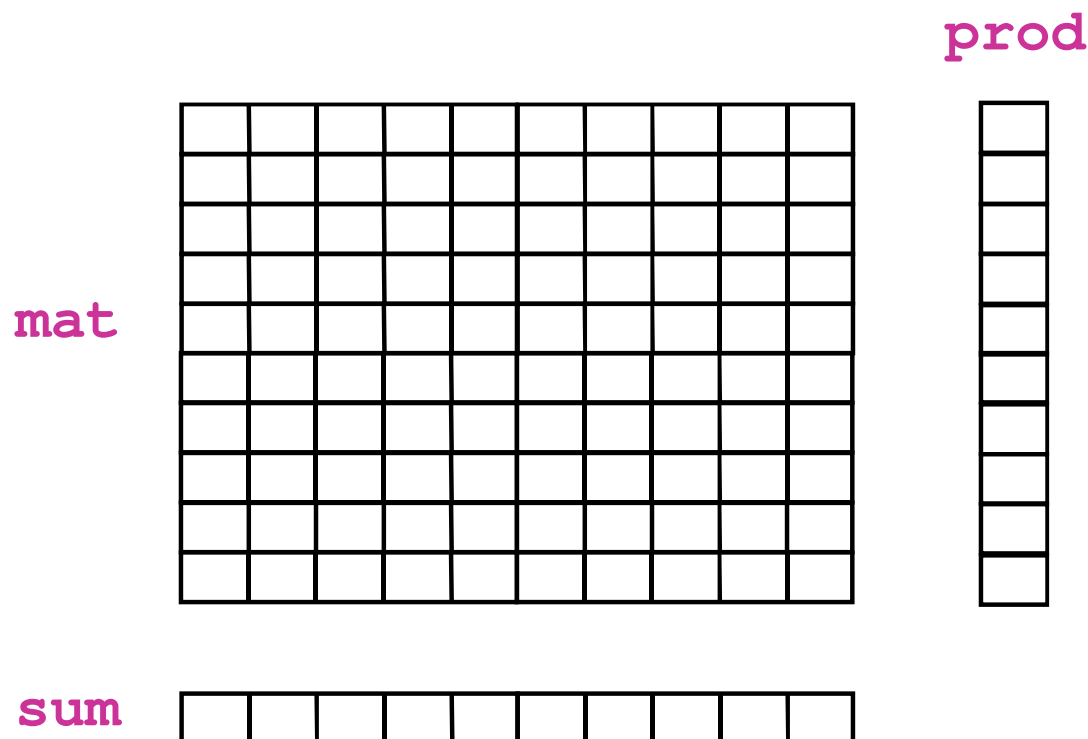
/* ispis prije transponiranja */
printf("\n\nIspis matrice prije transponiranja:\n");
for (i = 0; i < m; i++) {
    for (j = 0; j < n; j++) {
        printf("%3d", mat[i][j]);
    }
    printf("\n");
}
```

Primjer transponiranja matrice (4)

```
max = m > n ? m : n;
/* transponiranje */
for ( i=0; i<max-1; i++ ) {
    for ( j=i+1; j<max; j++ ) { /* petlja ide 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("\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;
}
```

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.



Izračunavanje suma članova stupaca i produkata članova redaka (2)

```
#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];

    /* 1.varijanta unosa 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]);
        }
    }
}
```

Izračunavanje suma članova stupaca i produkata članova redaka (3)

```
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 i racunanja suma i produkata */
```

Izračunavanje suma članova stupaca i produkata članova redaka (4)

```
/* 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;
    }
}
```

Izračunavanje suma članova stupaca i produkata članova redaka (5)

```
printf("\nNajmanja suma je %f , a pripadni"
      " indeks je %d\n",
      sum[min_sum_ind], min_sum_ind);
printf("\nNajveci produkt je %f , a pripadni"
      " indeks je %d\n",
      prod[max_prod_ind], max_prod_ind);
return 0;
}
```

Izračunavanje suma članova stupaca i produkata članova redaka (6)

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

```
/* 2. varijanta unosa 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 i racunanja suma i produkata */
```