Programiranje i programsko inženjerstvo

Predavanja

2014. / 2015.

6. Podatkovna struktura polje

Primjer

- Pročitati s tipkovnice 10 cijelih brojeva te ispisati njihov prosjek (aritmetičku sredinu).
- Rješenje je jednostavno i glasi:

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

Primjer

- Pročitati s tipkovnice 10 cijelih brojeva i uz svaki od učitanih brojeva ispisati jednu od sljedećih poruka:

  `broj je >= od prosjeka`

  `broj je < od prosjeka`

- Prosjek se može izračunati tek kad se učitaju svi brojevi. Zato **sve** učitane brojeve treba "pamtiti" dok se ne izračuna prosjek, a tada treba ispisivati učitane 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;
/* citanje i izracunavanje prosjeka */
scanf("%d", &broj1); suma = suma + broj1;
scanf("%d", &broj2); suma = suma + broj2;
... itd ...
prosjek = suma / 10.f;
/* ispis */
printf("%d: %s\n", broj1, broj1 < prosjek ? MANJI:VECI);
printf("%d: %s\n", broj2, broj2 < prosjek ? MANJI:VECI);
... itd ...
```

Što ako se zadatak još 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: $broj_0$, $broj_1$, $broj_2$..., $broj_n$.
- Kada bismo na raspolaganju imali mogućnost korištenja niza, prethodni zadatak bi se mogao riješiti na sljedeći način:

```
definiraj varijable n, suma, prosjek
definiraj niz broj
učitaj (n)
za i = 1 do n
    učitaj(broj_i)
    suma := suma + broj_i;
prosjek := suma / n;
za i = 1 do n
    ispiši (broj_i uz odgovarajuću poruku)
```

Tada više ne bi bilo 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[2] → 8.85e-12

y[0], y[1], ... su L-values

Definicija varijabli tipa polje

- Pri definiciji varijable tipa polje potrebno je odrediti:
  - ime varijable: definira se na isti način kao za jednostavne tipove podataka
  - tip podatka za članove polja (**`int, float, ...`**). Svi članovi jednog polja su istog tipa
  - broj članova polja. Navodi se u uglatim zagradama, kao cjelobrojni **konstantni** izraz (koji sadrži konstante i/ili simboličke konstante)

```
#define MAX 100
#define MAX_T 80
int x[20];                  polje od 20 cijelih brojeva (članova)
char znakovi[5*80];         polje od 400 znakova
float niz[MAX];             polje od 100 realnih brojeva
char tekst[MAX_T+1]         polje od 81 znaka
```

Pristupanje članovima polja

- Članovima polja pristupa se 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]`**

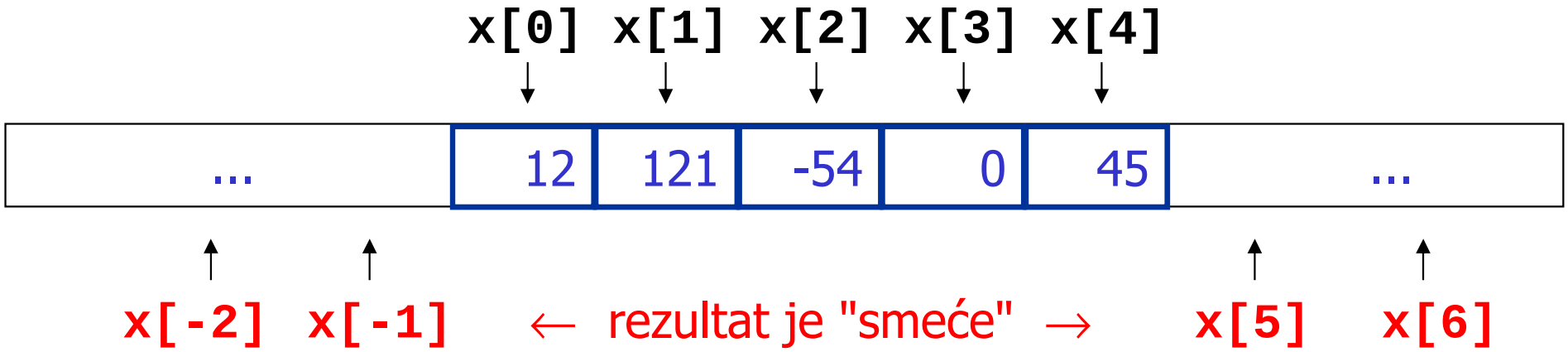
- Indeks člana (elementa) je broj između 0 i broja elemenata minus jedan, uključivo, tj.

  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 $0 \leq i \leq$ BrojElemenata – 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.f;
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

- Učitati pozitivan cijeli broj *n* ne veći od 1000. Nije potrebno provjeravati ispravnost unesenog broja. Zatim učitati *n* cijelih brojeva, izračunati njihov prosjek te ispisati učitane brojeve, svaki u svom retku, a uz svaki broj ispisati jednu od poruka

  broj je >= od prosjeka

  broj je < od prosjeka

```
Upisite pozitivan cijeli broj: 4
Upisite 1. broj: 3
Upisite 2. broj: 0
Upisite 3. broj: 19
Upisite 4. broj: -3
3: broj je < od prosjeka
0: broj je < od prosjeka
19: broj je >= od prosjeka
-3: broj je < od prosjeka
```

Rješenje

```
#include <stdio.h>
#define VECI "broj je >= od prosjeka"
#define MANJI "broj je < od prosjeka"
int main(void) {
   int n, i, broj[1000], suma=0; float prosjek;
   printf("Upisite pozitivan cijeli broj: ");
   scanf("%d", &n);
   for (i = 0; i < n; ++i) {
      printf("Upisite %d. broj: ", i+1);
      scanf("%d", &broj[i]);
      suma = suma + broj[i];
   }
   prosjek = (float)suma / n;
   for (i = 0; i < n; ++i) {
      printf("%d: %s\n",
             broj[i],
             broj[i] < prosjek ? MANJI : VECI);
   }
   return 0;
}
```

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] = {0.0f, 0.25f, 0.0f, -0.5f, 0.0f, 0.0f};
char boja[3] = {'C', 'P', 'Z'};
```

```
znam[0] = 1;    x[0] = 0.0f;      boja[0] = 'C';
znam[1] = 2;    x[1] = 0.25f;     boja[1] = 'P';
znam[2] = 3;    x[2] = 0.0f;      boja[2] = 'Z';
znam[3] = 4;    x[3] = -0.5f;
znam[4] = 5;    x[4] = 0.0f;
znam[5] = 6;    x[5] = 0.0f;
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] = {-0.3f, 0.0f, 0.25f};
```

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

```
x[0] = -0.3f;
x[1] = 0.0f;
x[2] = 0.25f;
x[3] = 0.0f;
x[4] = 0.0f;
x[5] = 0.0f;
```

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

- Sljedeća definicija i inicijalizacija polja ne inicijalizira sve članove polja na vrijednost 1:

```
int broj[5] = {1};
```

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[] = {0.0f, 0.25f, 0.0f, -0.5f};
```

isto kao: `float x[4] = {0.0f, 0.25f, 0.0f, -0.5f};`

```
znam[0] = 1;
znam[1] = 2;
znam[2] = 3;
znam[3] = 4;
znam[4] = 5;
znam[5] = 6;
```

```
x[0] = 0.0f;
x[1] = 0.25f;
x[2] = 0.0f;
x[3] = -0.5f;
```

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? U C-u ne postoji tip podatka *string*.
Koristi se jednodimenzijsko 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 nastavit će se ispisivati dok se ne naiđe 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 ako 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, programer mora osigurati prostor za barem jedan znak više!

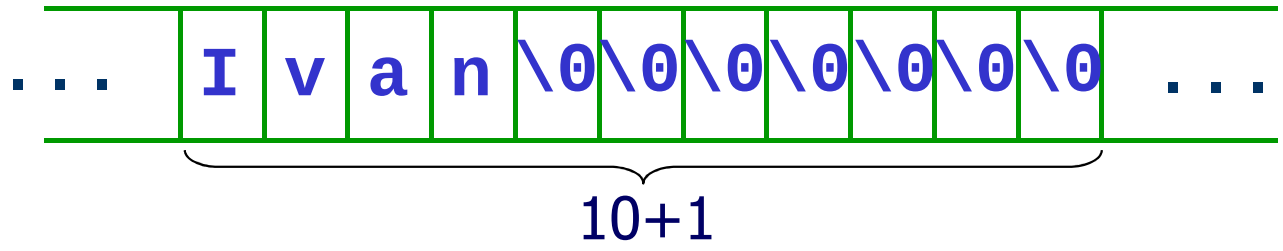
ili

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

potrebnu veličinu polja odredit će prevodilac, znak **\0** će biti dodan

Uobičajeno se varijable definiraju uz pomoć simboličkih konstanti

```
#define MAX_IME 10
char ime[MAX_IME + 1] = "Ivan";
```



Primjer

- Učitati ime studenta koje sigurno nije dulje od 20 znakova. Izračunati sumu ASCII vrijednosti svih znakova iz imena studenta te ispisati ime studenta i dobivenu sumu.

```
Upisite ime studenta: Ivana↵
Ivana 495
```

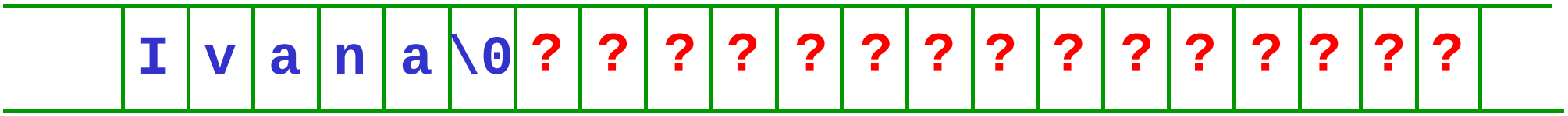
- Za učitavanje niza znakova prikladno je koristiti funkciju **`gets`**. Funkcija će u zadano polje pročitati sve znakove do znaka *novi red* (**`'\n'`** ili Enter) i dodati znak za označavanje kraja niza (**`'\0'`**).

Rješenje

```
#include <stdio.h>
#define MAX_IME 20
int main(void) {
   char ime[MAX_IME + 1];
   int i = 0, suma = 0;
   printf("Upisite ime studenta: ");
   gets(ime);
```

def. polja, niz nije dulji od 20 znakova

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




```
   while (ime[i] != '\0') {
      suma = suma + ime[i];
      i = i + 1;
   }
   printf("%s %d", ime, suma);
   return 0;
}
```

Može i ovako:

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

Ili ovako:

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

Primjer

- Učitati prirodan broj *n*, $1 \leq n \leq 100$. Učitavanje broja *n* ponavljati dok ne bude unesen ispravan broj. Zatim učitati *n* vrijednosti članova cjelobrojnog polja *k*. Izračunati aritmetičku sredinu članova polja različitih od nule. Ispisati vrijednosti i aritmetičku sredinu članova polja.

```
Unesite broj clanova polja : 0
Unesite broj clanova polja : -1
Unesite broj clanova polja : 3
Unesite clanove polja : 9 0 3
k(0) = 9
k(1) = 0
k(2) = 3
Aritmeticka sredina = 6.000000
```

Rješenje
- Pseudokod -

```
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 aritmetičku sredinu
inače
  | postavi aritmetičku sredinu na nulu
ispiši članove polja
ispiši sredinu
kraj
```

Rješenje
- 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
```

Rješenje
- C program, 1. dio -

```
#include <stdio.h>
                                        ⇦ #define DIM 100
int main(void) {
  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);

  printf("Unesite clanove polja : ");

  for (i=0; i<n; i=i+1) {               ⇦ for (i=0; i<n; ++i)
    scanf("%d", &k[i]);
  }
```

Rješenje
- C program, 2. dio -

```
suma = 0;
brojac = 0;

/* izracunaj sumu i prebroji nenulte clanove */
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;
  }
}
```

Rješenje
- C program, 3. dio -

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

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

  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 = %f", brojac, sredina);
  return 0;
}
```

Primjer

- Učitati niz od 10 realnih brojeva. Izračunati aritmetičku sredinu članova tog niza te najprije ispisati članove manje od aritmetičke sredine, a zatim one koji su veći od aritmetičke sredine.

```
Unesite 1. broj: 5.
Unesite 2. broj: -7.2
Unesite 3. broj: 9.1
Unesite 4. broj: 4.5
Unesite 5. broj: 8.
Unesite 6. broj: 6.
Unesite 7. broj: 2.
Unesite 8. broj: 15.7
Unesite 9. broj: 3.
Unesite 10. broj: 1.1
```

```
Aritm. sredina niza je 4.720000
-7.200000 je manji od aritm.sred.
4.500000 je manji od aritm.sred.
2.000000 je manji od aritm.sred.
3.000000 je manji od aritm.sred.
1.100000 je manji od aritm.sred.
5.000000 je veci od aritm.sred.
9.100000 je veci od aritm.sred.
8.000000 je veci od aritm.sred.
6.000000 je veci od aritm.sred.
15.700000 je veci od aritm.sred.
```

Rješenje
- 1. dio -

```
#include <stdio.h>
#define DIMENZIJA 10
int main(void) {
  int i;
  float suma = 0.f, ar_sred, niz[DIMENZIJA] = {0};
  for (i = 0; i < DIMENZIJA; ++i) {
    printf("Unesite %d. broj: ", i+1);
    scanf("%f",&niz[i]);
    suma += niz[i];
  }
  ar_sred = suma / DIMENZIJA;
  printf("Aritm. sredina niza je %f\n", ar_sred);
```

Rješenje
- 2. dio -

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

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

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

  return 0;
}
```

Primjer

- Učitati niz od 10 realnih brojeva. Nakon unosa sve članove niza podijeliti s najvećim članom niza i iskazati ih relativno u odnosu na najveći član.

```
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
```

```
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
```

Rješenje
- 1. dio -

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

Rješenje
- 2. dio -

```
  printf("Najveci element polja je %f\n\n", maks);

  for (i = 0; i < DIMENZIJA; ++i) {
    polje[i] /= maks;
    printf("polje[%d] = %f\n", i, polje[i]);
  }
  return 0;
}
```

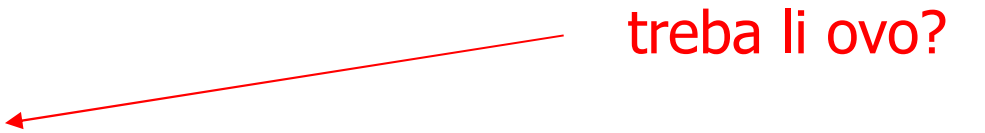
Primjer (varijanta a): Utvrđivanje frekvencije pojavljivanja brojeva prilikom učitavanja (1)

- Učitavati cijele brojeve u intervalu [0, 9]. Učitavanje prekinuti kad se unese broj izvan zadanog intervala. Zatim ispisati koliko je puta učitan svaki broj iz zadanog intervala, pri čemu treba ispisati samo one brojeve koji su učitani barem jednom.

```
Unesite broj u intervalu [0, 9]: 1
Unesite broj u intervalu [0, 9]: 5
Unesite broj u intervalu [0, 9]: 7
Unesite broj u intervalu [0, 9]: 5
Unesite broj u intervalu [0, 9]: 0
Unesite broj u intervalu [0, 9]: 5
Unesite broj u intervalu [0, 9]: 7
Unesite broj u intervalu [0, 9]: 10

Broj 0 se pojavio 1 puta
Broj 1 se pojavio 1 puta
Broj 5 se pojavio 3 puta
Broj 7 se pojavio 2 puta
```

Rješenje
- 1. dio -

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



Rješenje
- 2. dio -

```
  /* 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. Učitavati cijele brojeve u intervalu [10, 99]. Učitavanje prekinuti kad se unese broj izvan zadanog intervala. Zatim ispisati koliko je puta učitan svaki broj iz zadanog intervala, pri čemu treba ispisati samo one brojeve koji su učitani barem jednom.

```
Unesite broj u intervalu [10, 99]: 15
Unesite broj u intervalu [10, 99]: 25
Unesite broj u intervalu [10, 99]: 25
Unesite broj u intervalu [10, 99]: 15
Unesite broj u intervalu [10, 99]: 15
Unesite broj u intervalu [10, 99]: 11
Unesite broj u intervalu [10, 99]: 7

Broj 11 se pojavio 1 puta
Broj 15 se pojavio 3 puta
Broj 25 se pojavio 2 puta
```

Rješenje
- 1. dio -

```
#include <stdio.h>
#define DG 10              /* donja granica intervala */
#define GG 99              /* gornja granica intervala */
int main(void) {
  int broj, i;
  int brojac[GG - DG + 1] = { 0 };
  do {
    printf("Unesite broj u intervalu [%d, %d]: ",
           DG, GG);
    scanf("%d", &broj);
    if (broj >= DG && broj <= GG) {
      brojac[broj - DG]++;
    }
  } while (broj >= DG && broj <= GG);
```

Rješenje
- 2. dio -

```
  /* 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;
}
```

Višedimenzijska polja

- Jednodimenzijsko 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]
```

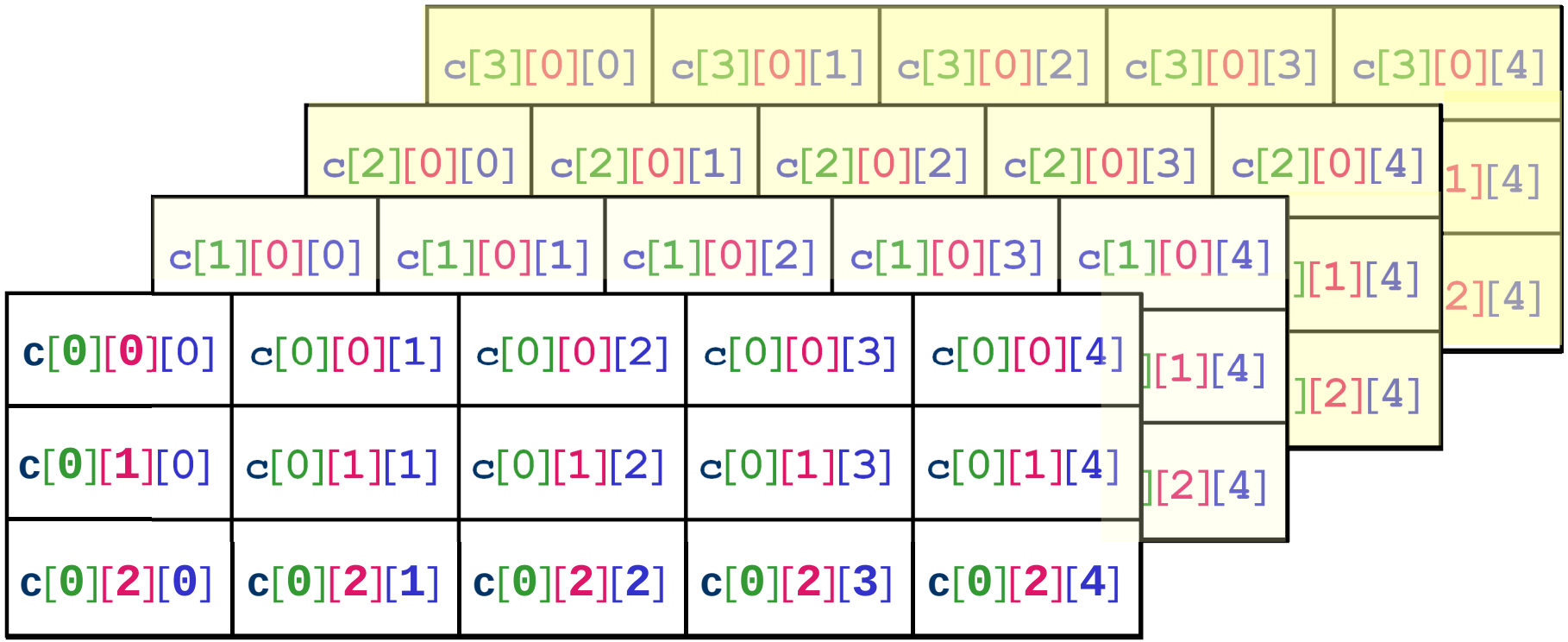
b[0][0]	b[0][1]	b[0][2]	b[0][3]	b[0][4]	1. redak
b[1][0]	b[1][1]	b[1][2]	b[1][3]	b[1][4]	2. redak
b[2][0]	b[2][1]	b[2][2]	b[2][3]	b[2][4]	3. redak

matrica od 3 retka i 5 stupaca

Višedimenzijska polja

- Polje može imati više dimenzija
  - tri dimenzije

```
int c[4][3][5]
```



Višedimenzijska polja

- Broj dimenzija nije ograničen npr.

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

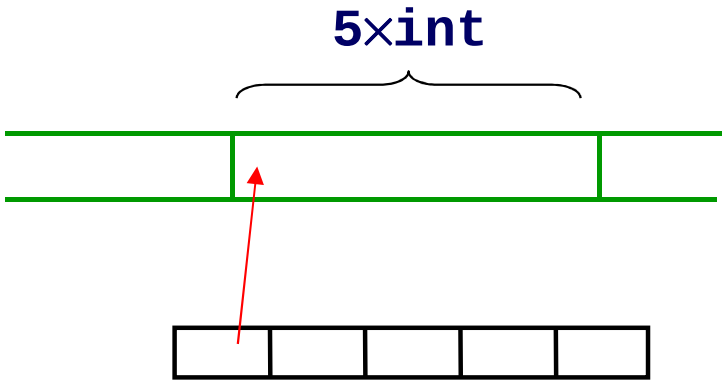
Oprez: veličina ovog polja je $8 * 3^{14} = 38\ 263\ 752$ bajta

Pri definiciji dimenzija polja uobičajeno je korištenje simboličkih konstanti:

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

Smještaj višedimenzijskih polja u memoriji računala

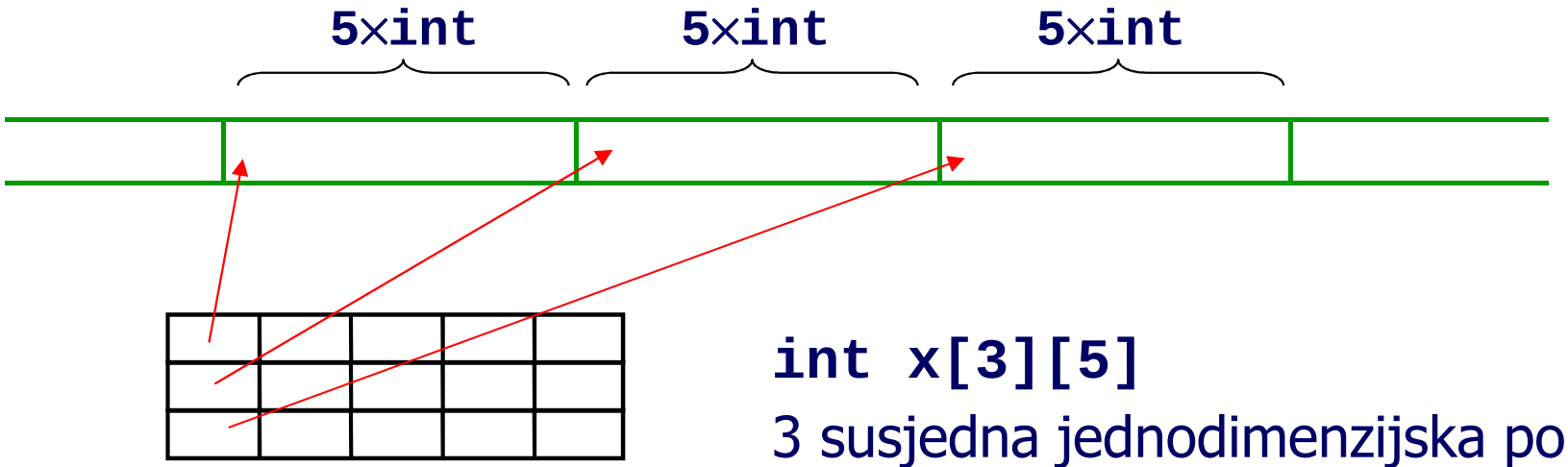
- Jednodimenzijsko polje: član za članom



`int x[5]`

5 susjednih cijelih brojeva

- Dvodimenzijsko polje: redak za retkom

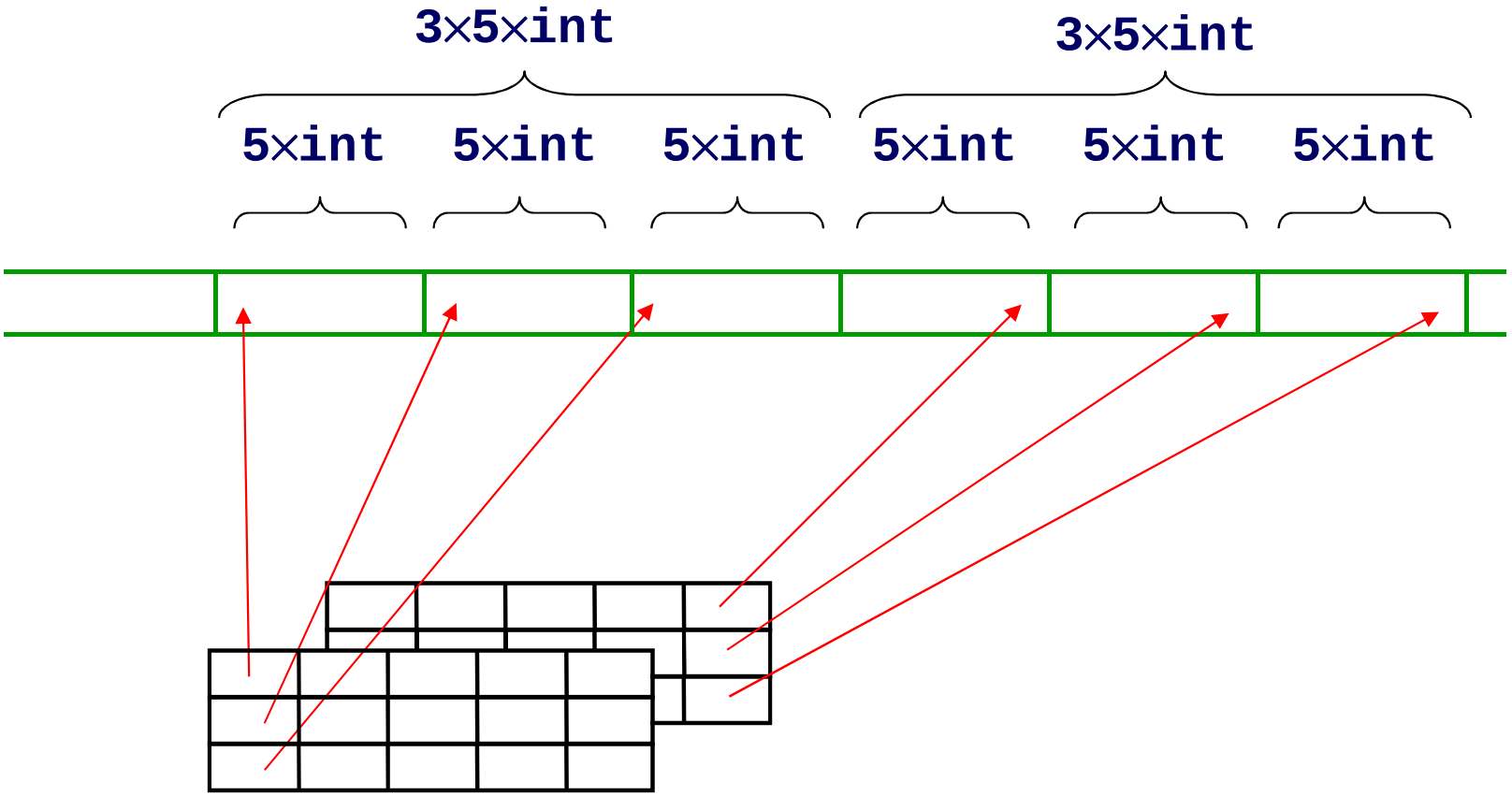


`int x[3][5]`

3 susjedna jednodimenzijska polja veličine 5 × **`int`**

Smještaj višedimenzijskih polja u memoriji računala

- Trodimenzijsko polje: sloj za slojem



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

2 susjedna dvodimenzijska polja veličine 3×5

Dodjeljivanje početnih vrijednosti članovima polja: primjer zadavanja početnih vrijednosti dvodimenzijskom 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 ovako:

```
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 višedimenzijskih polja 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

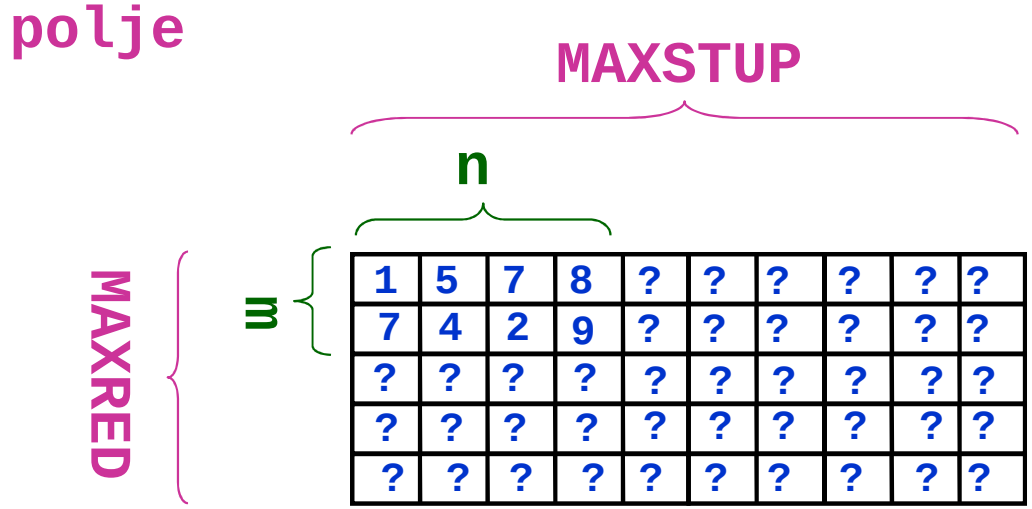
Operator sizeof i polja

- sizeof primijenjen nad imenom polja vraća ukupnu veličinu polja u oktetima (bajtovima)

```
int x[3][2];
sizeof(x)       →  24, tj. 6*sizeof(int)
sizeof(x[1][1]) →   4, tj. sizeof(int)

char c[10][20][80];
sizeof(c)             → 16000
sizeof(c[3][3][3])    →     1
```

Primjer

- Učitavanje i ispis matrice
- Učitati *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. Zatim učitati $m \times n$ članova matrice i ispisati ih u obliku dvodimenzijske tablice.



Primjer

```
Upisite m i n: 6 4
Upisite m i n: 3 11
Upisite m i n: 3 4
Upisite clanove (3 redaka i 4 stupaca):
183 11 -44 35
-2 0 57 64
987 4 0 145

  183    11   -44    35
   -2     0    57    64
  987     4     0   145
```

Rješenje
- 1. dio -

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

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

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

Rješenje
- 2. dio -

```
    printf("Upisite clanove (%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("%5d ", polje[i][j]);
       printf("\n");
    }

    return 0;
}
```

Primjer

- Određivanje najvećeg člana u svakom retku polja
- Učitati vrijednosti za broj redaka **mr** ≤ 100 i broj stupaca **ms** ≤ 10. Ponavljati učitavanje broja redaka i ponavljati učitavanje broja stupaca dok ne budu ispravni. Pročitati vrijednosti članova dvodimenzijskog realnog polja od **mr** redaka i **ms** stupaca. Odrediti u svakom retku najveći član i ispisati njegovu poziciju i vrijednost.

Primjer

```
Upisite vrijednost za broj redaka: 0
Upisite vrijednost za broj redaka: 3
Upisite vrijednost za broj stupaca: 0
Upisite vrijednost za broj stupaca: 11
Upisite vrijednost za broj stupaca: 4
Unesite vrijednosti clanova za 3 redaka i 4 stupaca:
1 2 4 3
-0.99 -1 -5 0
0 10 5.5 5.4
1.000000 2.000000 4.000000 3.000000
-0.990000 -1.000000 -5.000000 0.000000
0.000000 10.000000 5.500000 5.400000
Najveci clanovi polja u retcima:
a(1,3) = 4.000000
a(2,4) = 0.000000
a(3,2) = 10.000000
```

Rješenje
- Pseudokod -

```
{ Program za pronalaženje najvećih članova u recima polja }
učitavaj mr dok ne bude ispravan
učitavaj ms dok ne bude ispravan
učitaj i ispiši realno polje a od mr redaka i ms stupaca

{ traženje i ispis najvećih članova u retcima }
ponavljaj za sve retke
    | pronađi najveći član u retku
    | ispiši njegovu poziciju i vrijednost
kraj
```

Rješenje

- Detaljniji pseudokod - 1. dio -

```
{ učitavanje mr dok ne bude ispravan }
 ponavljaj
      | ispiši("Upisite vrijednost za broj redaka :  " )
      | učitaj (mr)
 dok je (mr < 1) ∨ (mr > NR)

{ učitavanje ms dok ne bude ispravan }
 ponavljaj
      | ispiši("Upisite vrijednost za broj stupaca :  " )
      | učitaj (ms)
 dok je (ms < 1) ∨ (ms > NS)
```

Rješenje

- Detaljniji pseudokod - 2. dio -

```
{ traženje najvećeg člana u svakom retku i ispis }
za i := 0 do mr-1
    najveci := a i, 0
    pozicija := 0
    za  j := 1 do ms-1
        ako je a i, j  > najveci
            najveci := a i, j
            pozicija := j
    { ispis pozicije i vrijednosti najvećeg člana polja }
    ispiši(i+1, pozicija+1, a i, pozicija)      ili ispiši(i+1, pozicija+1, najveci)
```

Rješenje

- C program, upute pretprocesoru, definicije -

```
#include <stdio.h>
#define NR 100
#define NS 10

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

Rješenje

- C program, učitavanje broja redaka i stupaca -

```
/* Ucitavanje mr dok ne bude ispravan */
do {
  printf("Upisite vrijednost za broj redaka: ");
  scanf("%d", &mr);
}  while (mr < 1 || mr > NR);

/* Ucitavanje ms dok ne bude ispravan */
do {
  printf("Upisite vrijednost za broj stupaca: ");
  scanf("%d", &ms);
}  while (ms < 1 || ms > NS);
```

Rješenje

- C program, učitavanje i kontrolni ispis polja -

```
/* ucitaj polje a od mr redaka i ms stupaca */
 printf("Unesite vrijednosti clanova 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");
 }
```

Rješenje

- C program, traženje najvećeg člana i ispis -

```
   /* Odredi i ispisi najvece clanove u retcima */
   printf("Najveci clanovi polja u retcima:\n");
   for (i = 0; i < mr; ++i) {
      najveci = a[i][0];
      pozicija = 0;
      for (j = 1; j < ms; ++j) {
        if (a[i][j] > najveci) {
             najveci = a[i][j];
             pozicija = j;
        }
      }
      /* Ispis pozicije i vrijednosti nadjenog clana */
      printf("a(%d,%d) = %f\n", i+1, pozicija+1,
              a[i][pozicija]);
   }
   return 0;
}
```

Primjer

- Pročitati vrijednosti članova dvodimenzijskog realnog polja od 10 redaka i 10 stupaca. Odrediti i ispisati najveću vrijednost na glavnoj i najveću vrijednost 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]?
```

Rješenje
- 1. dio -

```
#include <stdio.h>
#define BR_RED  10
#define BR_STUP 10
int main(void) {
  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]);
    }
  }
```

Rješenje
- 2. dio -

```
  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;
}
```

Rješenje

- skraćena varijanta sa samo jednim prolaskom kroz matricu - 1. dio -

```
#include <stdio.h>
#define BR_RED  10
#define BR_STUP 10
int main(void) {
  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];
      }
```

Rješenje

- skraćena varijanta sa samo jednim prolaskom kroz matricu - 2. dio -

```
      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

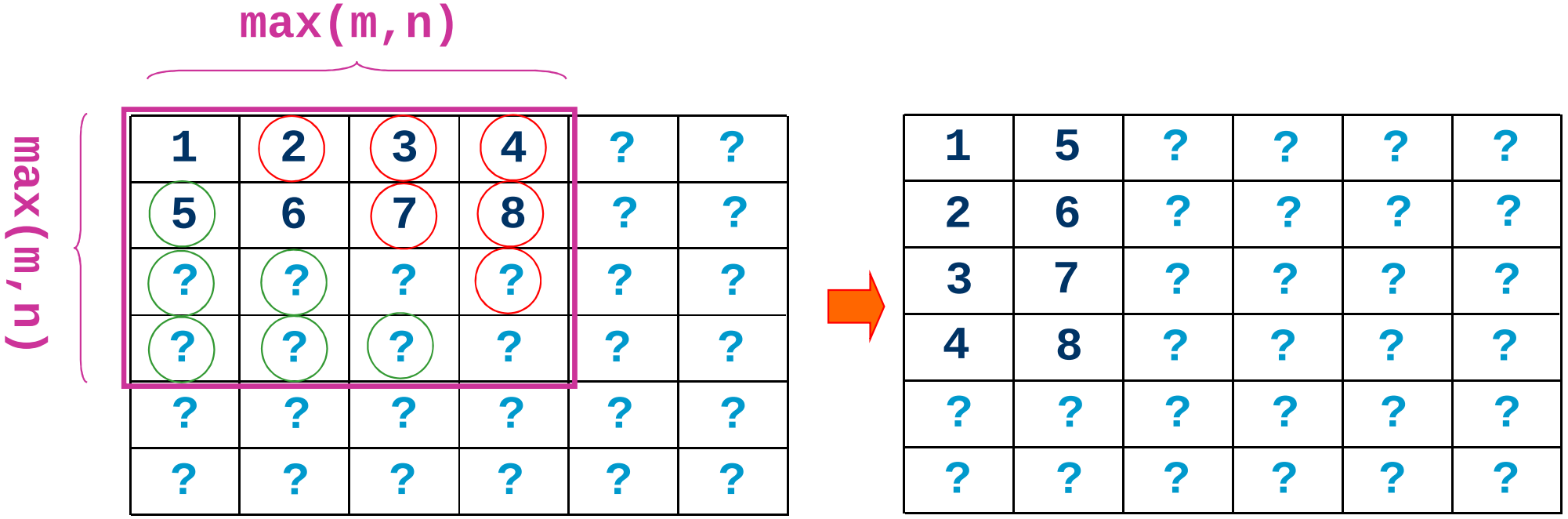
- Transponiranje matrice
- Učitati vrijednosti za broj redaka i broj stupaca cjelobrojne matrice. Matrica ne smije imati više od 50 redaka i 50 stupaca. Učitavanje broja redaka i stupaca ponavljati dok ne budu ispravni. Pročitati vrijednosti elemenata matrice. Ispisati učitanu matricu u obliku tablice. Zatim matricu transponirati i ponovo ispisati.
- 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

```
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
```

Analiza

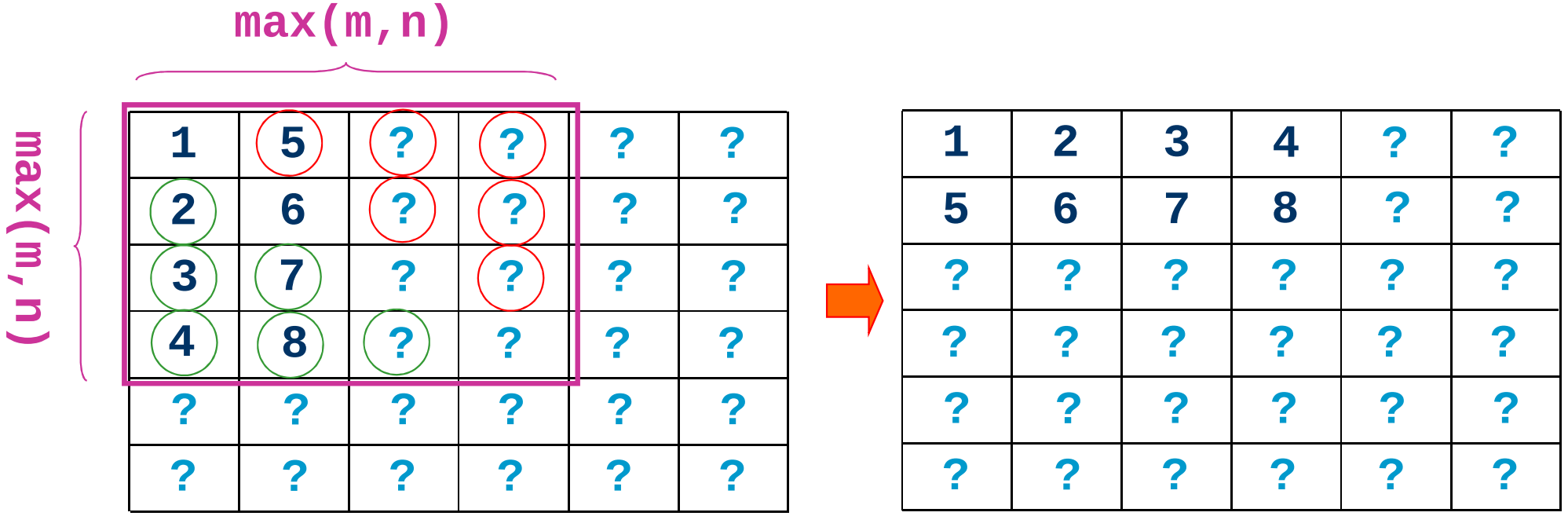
- Slučaj kada ima više stupaca nego redaka



- 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`

Analiza

- Slučaj kada 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`

Rješenje
- 1. dio -

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

  /* ucitavanje dimenzija 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);
```

Rješenje
- 2. dio -

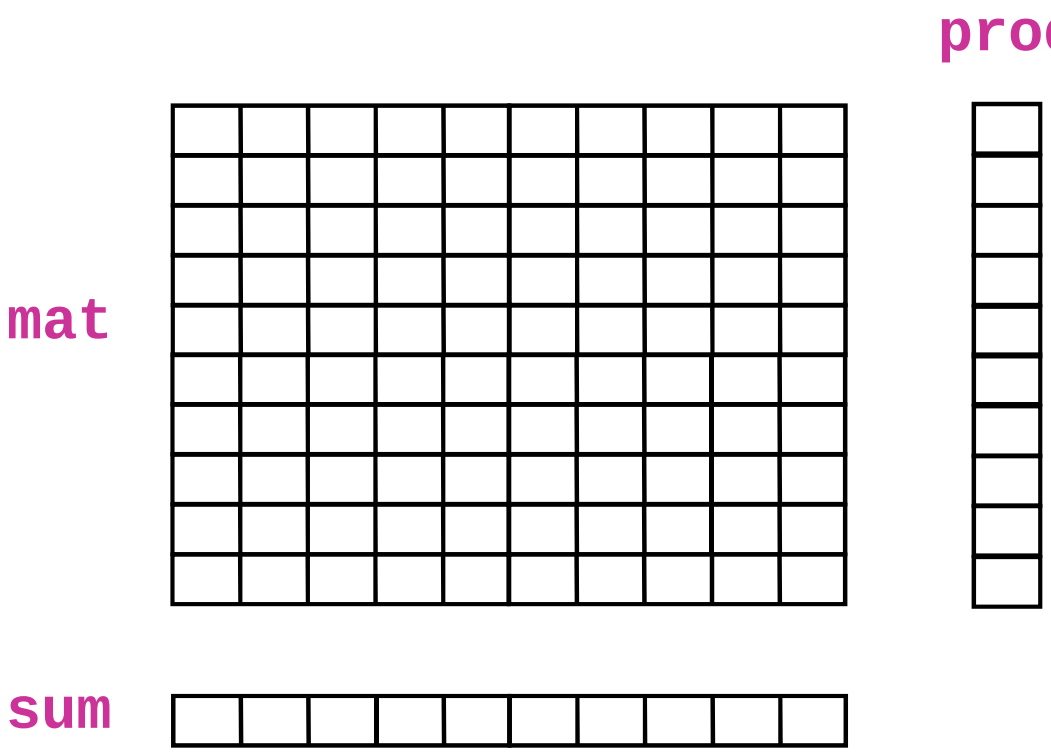
```
/* ucitavanje elemenata matrice */
printf("Unos elemenata matrice :\n");
for (i = 0; i < m; ++i) {
  for (j = 0; j < n; ++j) {
    printf("Unesite element [%d][%d] : ", i, j);
    scanf("%d", &mat[i][j]);
  }
}

/* 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");
}
```

Rješenje
- 3. dio -

```
  maks = m > n ? m : n;
  /* transponiranje */
  for ( i=0; i<maks-1; ++i ) {
    for ( j=i+1; j<maks; ++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

- Učitati elemente realne matrice 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 jednodimenzijskim poljima.



Rješenje
- 1. dio -

```
#include <stdio.h>
#define BR_RED  10
#define BR_STUP 10
int main(void) {
  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]);
    }
  }
```

Rješenje
- 2. dio -

```
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 */
```

Rješenje
- 3. dio -

```
/* 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 najveci produkt */
 max_prod_ind = 0;
 for (i = 1; i < BR_RED; ++i) {
   if (prod[i] > prod[max_prod_ind]) {
     max_prod_ind = i;
   }
 }
```

Rješenje
- 4. dio -

```
  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;
}
```

Rješenje

- skraćena varijanta učitavanja elemenata 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 */
```