

Uvod u programiranje - predavanja -

studeni 2025.

7. Agregatni tipovi podataka

1D polja, nastavak - sort

Vodič „06 Prije sedmog predavanja”

- Očekuje se da ste savladali (na razini na kojoj je obrađeno u vodiču):
 - Višedimenzionalna polja

- Pitanja?



Primjer

- Programski zadatak
 - Učitati veličinu polja n (ne treba kontrolirati ispravnost) i učitati n članova cjelobrojnog polja. Ispisati poziciju (indeks) i vrijednost najmanjeg člana polja. Ako više članova polja ima istu najmanju vrijednost, ispisati poziciju prvog člana s najmanjom vrijednošću.

```
Upisite velicinu polja > 5↵
Upisite clanove polja > 7 5 4 6 4↵
↵
Vrijednost 4 na poziciji 2↵
```

Rješenje

```
#include <stdio.h>

int main(void) {
    int n, i;    // velicina polja, kontrolna varijabla petlje
    printf("Upisite velicinu polja > ");
    scanf("%d", &n);

    int polje[n];

    printf("Upisite clanove polja > ");
    for (i = 0; i < n; i = i + 1) {
        scanf("%d", &polje[i]);
    }

    int min_ind = 0;    // pretpostavka: indeks minimalnog clana
    for (i = 1; i < n; i = i + 1) {
        if (polje[i] < polje[min_ind]) min_ind = i;    // promijeni pretpostavku
    }
    printf("\nVrijednost %d na poziciji %d\n", polje[min_ind], min_ind);
    return 0;
}
```

Primjer

- Programski zadatak (SORTIRANJE)
 - Učitati veličinu polja n (ne treba kontrolirati ispravnost) i učitati n članova cjelobrojnog polja. **Poredati (sortirati)** članove polja od manjih prema većim. Ispisati sadržaj sortiranog polja.

```
Upisite velicinu polja > 5↵
Upisite clanove polja > 9 4 7 2 5↵
↵
2 4 5 7 9↵
```

Rješenje

- Sortiranje članova polja

- Ovdje će se koristiti jedan od najjednostavnijih i najmanje efikasnih algoritama za sortiranje: *selection sort*

- za svaki i , $0 \leq i \leq n-2$

- pronajdi indeks ind_min najmanjeg člana $polje[j]$, $i < j \leq n-1$

- ako je $polje[ind_min] < polje[i]$, zamijeni $polje[i]$ i $polje[ind_min]$

i = 0	polje	<table><tr><td>9</td><td>4</td><td>7</td><td>2</td><td>6</td><td>5</td></tr></table>	9	4	7	2	6	5	ind_min = 3, polje[ind_min] < polje[i] → zamijeni
9	4	7	2	6	5				
i = 1	polje	<table><tr><td>2</td><td>4</td><td>7</td><td>9</td><td>6</td><td>5</td></tr></table>	2	4	7	9	6	5	ind_min = 5, polje[ind_min] ≥ polje[i]
2	4	7	9	6	5				
i = 2	polje	<table><tr><td>2</td><td>4</td><td>7</td><td>9</td><td>6</td><td>5</td></tr></table>	2	4	7	9	6	5	ind_min = 5, polje[ind_min] < polje[i] → zamijeni
2	4	7	9	6	5				
i = 3	polje	<table><tr><td>2</td><td>4</td><td>5</td><td>9</td><td>6</td><td>7</td></tr></table>	2	4	5	9	6	7	ind_min = 4, polje[ind_min] < polje[i] → zamijeni
2	4	5	9	6	7				
i = 4	polje	<table><tr><td>2</td><td>4</td><td>5</td><td>6</td><td>9</td><td>7</td></tr></table>	2	4	5	6	9	7	ind_min = 5, polje[ind_min] < polje[i] → zamijeni
2	4	5	6	9	7				
kraj	polje	<table><tr><td>2</td><td>4</td><td>5</td><td>6</td><td>7</td><td>9</td></tr></table>	2	4	5	6	7	9	
2	4	5	6	7	9				

Rješenje

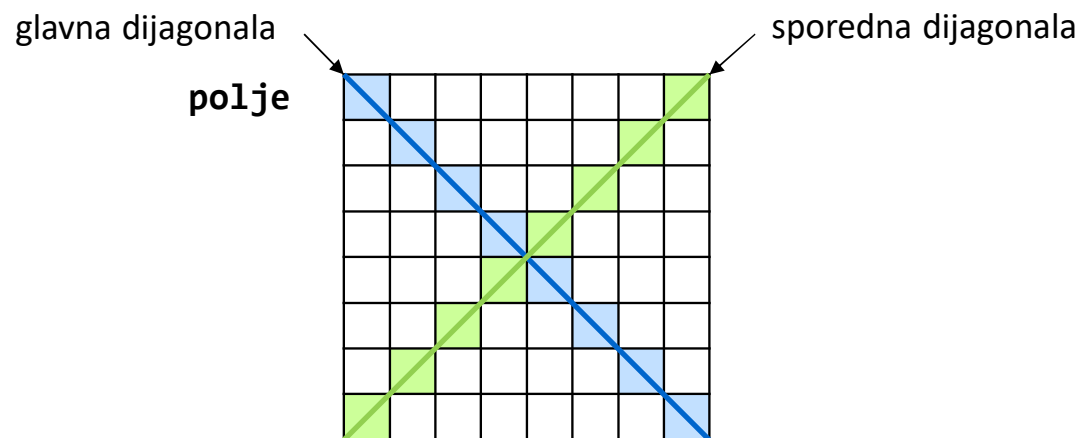
```
/* izostavljen uobicajeni kod za ucitavanje n i n clanova polja */
for (i = 0; i < n - 1; i = i + 1) {
    /* trazi indeks najmanjeg medju polje[i+1] ... polje[n-1] */
    ind_min = i + 1;
    for (j = i + 2; j < n; j = j + 1) {
        if (polje[j] < polje[ind_min]) ind_min = j;
    }
    /* u ind_min se sada nalazi indeks najmanjeg clana */
    if (polje[ind_min] < polje[i]) {
        /* obavi zamjenu clanova polje[i] i polje[ind_min] */
        pomocna = polje[i];
        polje[i] = polje[ind_min];
        polje[ind_min] = pomocna;
    }
}
/* izostavljen uobicajeni kod za ispis */
```

Polja

Višedimenzijska polja

Primjer

- Programski zadatak
 - Učitati vrijednosti (realni tip podataka) kvadratne matrice reda 8
 - kvadratna matrica je tablica koja ima jednak broj redaka i stupaca (red matrice)
 - Ispisati najmanju vrijednost na glavnoj dijagonali i najmanju vrijednost na sporednoj dijagonali
 - glavna i sporedna dijagonala kvadratne matrice



Rješenje

- Glavna dijagonala

- pretpostavka: `polje[0][0]` je najmanji
- provjeriti ostale na glavnoj dijagonali

```
min_gl = polje[0][0];  
for (i = 1; i < 8; i = i + 1)  
    if (polje[i][i] < min_gl)  
        promijeni pretpostavku
```

- Sporedna dijagonala

- pretpostavka: `polje[0][7]` je najmanji
- provjeriti ostale na sporednoj dijagonali

```
min_sp = polje[0][7];  
for (i = 1; i < 8; i = i + 1)  
    if (polje[i][8 - 1 - i] < min_sp)  
        promijeni pretpostavku
```

0, 0							0, 7
	1, 1					1, 6	
		2, 2			2, 5		
			3, 3	3, 4			
			4, 3	4, 4			
		5, 2			5, 5		
	6, 1					6, 6	
7, 0							7, 7

Rješenje (1. dio)

```
#include <stdio.h>
#define RED_MATRICE 8

int main(void) {
    int i, j;
    float polje[RED_MATRICE][RED_MATRICE], min_gl, min_sp;
    /* izostavljen je uobicajeni kod za ucitavanje clanova polja */
    /* pretpostavka: prvi clan glavne dijagonale je najmanji */
    min_gl = polje[0][0];
    /* provjeri ostale clanove glavne dijagonale */
    for (i = 1; i < RED_MATRICE; i = i + 1) {
        if (polje[i][i] < min_gl) {
            /* promijeni pretpostavku */
            min_gl = polje[i][i];
        }
    }
}
```

Rješenje (2. dio)

```
/* pretpostavka: prvi clan sporedne dijagonale je najmanji */
min_sp = polje[0][RED_MATRICE - 1];
/* provjeri ostale clanove sporedne dijagonale */
for (i = 1; i < RED_MATRICE; i = i + 1) {
    if (polje[i][RED_MATRICE - 1 - i] < min_sp) {
        /* promijeni pretpostavku */
        min_sp = polje[i][RED_MATRICE - 1 - i];
    }
}

/* izostavljen je uobicajeni kod za ispis rezultata */

return 0;
}
```

Primjer

- Programski zadatak
 - Učitati vrijednosti za broj redaka m i broj stupaca n matrice mat (dvodimenzijskog cjelobrojnog polja). Nije potrebno provjeravati jesu li upisane ispravne vrijednosti.
 - Učitati vrijednosti matrice mat . Ispisati ih u obliku tablice
 - Definirati novo polje $matT$ u koje treba pohraniti matricu dobivenu transponiranjem matrice mat
 - transponiranje se obavlja zamjenom redaka sa stupcima: element na poziciji $[i][j]$ matrice mat postaje element na poziciji $[j][i]$ matrice $matT$
 - transponirana matrica će imati n redaka i m stupaca
 - dobivenu matricu $matT$ ispisati u obliku tablice

Primjer

- primjer izvršavanja programa

```
Upisite broj redaka i broj stupaca > 3 5↵
```

```
Upisite 3 x 5 cijelih brojeva >↵
```

```
1 2 3 4 5↵
```

```
6 7 8 9 10↵
```

```
11 12 13 14 15↵
```

```
↵
```

```
1 2 3 4 5↵
```

```
6 7 8 9 10↵
```

```
11 12 13 14 15↵
```

```
↵
```

```
1 6 11↵
```

```
2 7 12↵
```

```
3 8 13↵
```

```
4 9 14↵
```

```
5 10 15↵
```

Rješenje

		n				
mat		1	2	3	4	5
	m	6	7	8	9	10
		11	12	13	14	15

```
for (i = 0; i < m; i = i + 1)
    for (j = 0; j < n; j = j + 1)
        matT[j][i] = mat[i][j];
```

		m		
matT		?	?	?
		?	?	?
	n	?	?	?
		?	?	?
		?	?	?

```
mat[0][0] → matT[0][0]
mat[0][1] → matT[1][0]
mat[0][2] → matT[2][0]
...
mat[2][2] → matT[2][2]
mat[2][3] → matT[3][2]
mat[2][4] → matT[4][2]
```

		m		
matT		1	6	11
		2	7	12
	n	3	8	13
		4	9	14
		5	10	15

Rješenje (1. dio)

```
#include <stdio.h>

int main(void) {
    int m, n, i, j;
    printf("Upisite broj redaka i broj stupaca > ");
    scanf("%d %d", &m, &n);
    int mat[m][n], matT[n][m];

    /* učitaj matricu mat m x n */
    printf("Upisite %d x %d cijelih brojeva >\n", m, n);
    for (i = 0; i < m; i = i + 1) {
        for (j = 0; j < n; j = j + 1) {
            scanf("%d", &mat[i][j]);
        }
    }

    printf("\n");
}
```


Rješenje (2. dio)

```
/* ispiši matricu mat m x n */
for (i = 0; i < m; i = i + 1) {
    for (j = 0; j < n; j = j + 1) {
        printf("%4d", mat[i][j]);
    }
    printf("\n");
}
printf("\n");

/* vrijednosti iz mat kopiraj na odgovarajuće pozicije u matT */
for (i = 0; i < m; i = i + 1) {
    for (j = 0; j < n; j = j + 1) {
        matT[j][i] = mat[i][j];
    }
}
```

Rješenje (3. dio)

```
/* ispisi matricu matT n x m */  
for (i = 0; i < n; i = i + 1) {  
    for (j = 0; j < m; j = j + 1) {  
        printf("%4d", matT[i][j]);  
    }  
    printf("\n");  
}  
return 0;  
}
```

Primjer

- Programski zadatak
 - Učitati red kvadratne matrice n . Nije potrebno provjeravati ispravnost unosa
 - Učitati elemente kvadratne matrice mat reda n . U obliku tablice ispisati učitane vrijednosti
 - Transponirati matricu mat zamjenom članova unutar matrice (bez definiranja nove matrice i bez korištenja pomoćnih polja)
 - U obliku tablice ispisati novi sadržaj matrice

Rješenje

mat

n

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

mat

n

1	6	11	16	21
2	7	12	17	22
3	8	13	18	23
4	9	14	19	24
5	10	15	20	25

```
for (i = 0; i < n - 1; i = i + 1)
    for (j = i + 1; j < n; j = j + 1)
        mat[i][j] ↔ mat[j][i];
```

```
mat[0][1] ↔ mat[1][0]
mat[0][2] ↔ mat[2][0]
mat[0][3] ↔ mat[3][0]
mat[0][4] ↔ mat[4][0]
mat[1][2] ↔ mat[2][1]
...
mat[3][4] ↔ mat[4][3]
```

Rješenje

```
#include <stdio.h>

int main(void) {
    int n, i, j, pomocna;
    printf("Upisite red matrice > ");
    scanf("%d", &n);
    int mat[n][n];

    /* izostavljen je uobicajeni kod za ucitavanje polja */
    /* izostavljen je uobicajeni kod za ispis polja */

    for (i = 0; i < n - 1; i = i + 1) {
        for (j = i + 1; j < n; j = j + 1) {
            pomocna = mat[i][j];
            mat[i][j] = mat[j][i];
            mat[j][i] = pomocna;
        }
    }
    /* izostavljen je uobicajeni kod za ispis novog sadrzaja polja */
    return 0;
}
```

Zadatak



- **Unesite dimenziju kvadratne matrice.**
- Popunite matricu jedinicama tako da oponašate uzorak mete.
- Npr.



Unesi dim kvadratne matrice (< 100): 12

```
1 1 1 1 1 1 1 1 1 1 1 1
1                                     1
1   1 1 1 1 1 1 1 1 1   1
1   1                                     1 1
1   1   1 1 1 1   1 1   1
1   1   1   1   1 1   1 1   1
1   1   1   1   1 1   1 1   1
1   1   1   1 1 1 1   1 1   1
1   1   1   1 1 1 1   1 1   1
1   1   1 1 1 1 1 1 1 1   1
1   1   1 1 1 1 1 1 1 1   1
1 1 1 1 1 1 1 1 1 1 1 1
```

Unesi dim kvadratne matrice (< 100): 13

```
1 1 1 1 1 1 1 1 1 1 1 1 1
1                                     1
1   1 1 1 1 1 1 1 1 1 1   1
1   1   1 1 1 1 1 1 1 1   1
1   1   1   1 1 1 1 1 1   1
1   1   1   1   1 1 1 1   1
1   1   1   1   1 1 1 1   1
1   1   1   1   1 1 1 1   1
1   1   1   1   1 1 1 1   1
1   1   1   1   1 1 1 1   1
1   1   1   1   1 1 1 1   1
1   1   1   1   1 1 1 1   1
1   1   1   1   1 1 1 1   1
1 1 1 1 1 1 1 1 1 1 1 1
```

Agregatni tipovi podataka

Strukture

Struktura

- Struktura ili zapis (*structure, record*) je složeni tip podatka koji obuhvaća više članova koji ne moraju biti istog tipa
 - jedna varijabla ↔ više vrijednosti
 - jednim imenom objedinjuje se više logički povezanih podataka
 - npr. umjesto zasebnih varijabli

```
int mbr;           // maticni broj studenta
float mi;          // broj bodova medjuispit
float zi;          // broj bodova završni ispit
float lab[8];      // broj bodova lab. vježbe
```

- definira se varijabla bodovi koja će imati članove mbr, mi, zi i lab
- pojedinim vrijednostima se pristupa pomoću imena člana

Deklaracija strukture

- Struktura se mora prvo opisati
 - deklaracija strukture, nacrt strukture (*structure tag*)

- ime strukture
 - imena i tipovi članova strukture

```
struct naziv_strukture {  
    tip_elementa_1  ime_elementa_1;  
    tip_elementa_2  ime_elementa_2;  
    ...  
    tip_elementa_n  ime_elementa_n;  
};
```

- Deklaracija strukture se nakon toga može koristiti
 - za definiranje varijabli (može i polja) tipa strukture
 - za deklaraciju drugih struktura koji će sadržavati članove tipa strukture

Primjer

deklaracija strukture **bodovi_s**

```
struct bodovi_s {  
    int mbr;  
    float mi;  
    float zi;  
    float lab[8];  
};
```

prema konvenciji imenu strukture dodaje se nastavak **_s**

definicija varijable **bodovi**

```
struct bodovi_s bodovi;  
...  
bodovi.mbr = 1234;  
bodovi.mi = 23.5f;  
bodovi.lab[0] = 1.5f;  
bodovi.lab[1] = 1.2f;
```

varijabla **bodovi** je tipa strukture **bodovi_s**

operator pristupa članu strukture: **.** (točka)

pristup članovima strukture: **ime_varijable.ime_clana**

bodovi

mbr	1234							
mi	23.5							
zi	?							
lab	1.5	1.2	?	?	?	?	?	?

Deklaracija strukture - varijante

- Varijable je moguće definirati istovremeno uz deklaraciju strukture, a kasnije istu deklaraciju koristiti za definiranje dodatnih varijabli

```
struct datum_s {  
    int dan;  
    int mjesec;  
    int godina;  
} francRev, amerRev;  
...  
francRev.dan = 5; francRev.mjesec = 5; francRev.godina = 1789;  
amerRev.dan = 19; amerRev.mjesec = 4; amerRev.godina = 1775;  
...  
struct datum_s seljBuna;  
seljBuna.dan = 28; seljBuna.mjesec = 1; seljBuna.godina = 1573;
```

Deklaracija strukture - varijante

- Ime strukture se može ispustiti iz deklaracije ako deklaracija neće biti potrebna za definiranje dodatnih varijabli

```
struct {  
    float latituda;  
    float longituda;  
} geogrPozEiffel, geogrPozFER;  
...  
geogrPozEiffel.latituda = 48.85822f;  
geogrPozEiffel.longituda = 2.2945f;  
geogrPozFER.latituda = 45.80107f;  
geogrPozFER.longituda = 15.97083f;  
...
```

Definicija uz inicijalizaciju

- Slično poljima, varijabla tipa strukture se može inicijalizirati u trenutku definicije

```
struct bodovi_s bodovi1 = { 4321, 10.5f, 21.5f,  
                           {0.5f, 1.5f, 1.8f}  
                           };
```

mbr	4321							
mi	10.5							
zi	21.5							
lab	0.5	1.5	1.8	0.0	0.0	0.0	0.0	0.0

```
struct bodovi_s bodovi2 = { 1243, .zi = 12.5f,  
                           {0.5f, [5] = 2.0f}  
                           };
```

mbr	1243							
mi	0.0							
zi	12.5							
lab	0.5	0.0	0.0	0.0	0.0	2.0	0.0	0.0

Složene strukture

- Moguće je definiranje podatkovne strukture proizvoljne složenosti jer član strukture može biti struktura ili polje:

```
struct datum_s {
    int dan;
    int mj;
    int god;
};
struct interval_s {
    struct datum_s dat_od;
    struct datum_s dat_do;
};
struct interval_s zim_rok = {{11, 2, 2019}, {22, 2, 2019}};
printf("Zimski rok: %d.%d.%d. - %d.%d.%d.",
       zim_rok.dat_od.dan, zim_rok.dat_od.mj, zim_rok.dat_od.god,
       zim_rok.dat_do.dan, zim_rok.dat_do.mj, zim_rok.dat_do.god);
```

Struktura jest modifiable lvalue

- Za razliku od polja, varijabla tipa strukture jest modifiable lvalue
 - čak i onda kada se kao član strukture koristi polje!

```
struct bodovi_s bodovi1 = { 4321, 10.5f, 21.5f,  
                           {0.5f, 1.5f, 1.8f}  
                           };  
  
struct bodovi_s bodovi2;  
bodovi2 = bodovi1;
```

ispravno

- Operator pridruživanja je jedini operator koji se može koristiti za operacije s dva operanda tipa strukture. Npr. nije moguće koristiti relacijske operatore

```
if (bodovi1 == bodovi2) {  
    ...
```

prevodilac dojavljuje pogrešku

Strukture kompatibilnog tipa

- Varijabli tipa strukture može se pridružiti sadržaj druge varijable tipa strukture samo u slučaju kada su njihovi tipovi kompatibilni, a to znači da su varijable definirane na temelju iste deklaracije

```
struct koordinata_s {  
    float latituda;  
    float longituda;  
};
```

```
struct pozicija_s {  
    float latituda;  
    float longituda;  
};
```

```
struct koordinata_s tocka1 = {45.80107f, 15.97083f};
```

```
struct koordinata_s tocka2;
```

```
tocka2 = tocka1;
```

```
struct pozicija_s tocka3;
```

```
tocka3 = tocka1;
```

Varijable definirane na temelju deklaracije `koordinata_s` nisu kompatibilne s varijablama definiranim na temelju deklaracije `pozicija_s`. To što su im članovi jednakih imena i tipova za kompatibilnost nije dovoljno.

ispravno

prevodilac dojavljuje pogrešku

Strukture mogu biti članovi polja

- Iako pojedinačne varijable tipa strukture mogu biti korisne, strukture se najčešće koriste kao elementi složenijih podatkovnih struktura, npr. polja

```
struct datum_s {  
    int dan;  
    int mj;  
    int god;  
};  
struct datum_s praznici_2018[] = {  
    { 1, 1, 2018}, { 6, 1, 2018}  
    , { 1, 4, 2018}, { 2, 4, 2018}  
    , { 1, 5, 2018}, {31, 5, 2018}  
    , {22, 6, 2018}, {25, 6, 2018}  
    , { 5, 8, 2018}, {15, 8, 2018}  
    , { 8, 10, 2018}  
    , { 1, 11, 2018}  
    , {25, 12, 2018}, {26, 12, 2018}  
};
```

Primjer

- Programski zadatak
 - s tipkovnice učitati cijeli broj n koji predstavlja broj studenata na predmetu Uvod u programiranje. Za svakog studenta učitati matični broj (int), broj bodova na međuispitu (float), broj bodova na završnom ispitu (float) i broj bodova za svaku od osam laboratorijskih vježbi (float). Evidentirati sumu bodova na svim provjerama znanja za svakog studenta (ukupni broj bodova).
 - Sortirati studente prema ukupnom broju bodova, u poretku od većih prema manjim. Poredak studenata koji imaju međusobno jednak broj bodova nije važan.
 - Sortirane podatke ispisati u obliku tablice

Primjer

■ Primjer izvršavanja programa

```
Upisite broj studenata > 740↵
Upisite podatke > 360149290 11.6 5.8 1.9 0.6 1.7 1.4 1.0 1.9 1.5 0.2↵
Upisite podatke > 553721121 15.9 10.2 0.9 0.8 0.8 1.3 1.9 0.4 0.4 1.8↵
Upisite podatke > 277253502 33.3 44.2 1.4 1.8 1.6 0.8 0.9 1.6 0.0 1.4↵
...
Upisite podatke > 380893153 3.4 0.4 1.1 0.2 1.5 0.1 0.4 0.6 1.3 0.7↵
Upisite podatke > 711650074 28.5 9.4 1.8 0.8 1.1 1.2 1.8 0.0 0.0 0.5↵
↵
Rang lista↵
Rbr.  Mat. broj    MI    ZI    LAB  Ukupno↵
=====↵
  1. 277253502  33.3  44.2   9.5   87.0↵
  2. 378063837  33.5  44.6   8.6   86.7↵
  3. 419558299  28.9  44.4  11.9   85.2↵
...
739. 389674171   0.2   1.2   8.6  10.0↵
740. 380893153   3.4   0.4   5.9   9.7↵
```

Rješenje (1. dio)

```
#include <stdio.h>
#define BR_LAB_VJ 8
int main(void) {
    struct bodovi_s {
        int mbr;
        float mi;
        float zi;
        float lab[BR_LAB_VJ];
        float ukupno;
    };
    int n, i, j;
    float ukupno;
    float ukupno_lab;
    printf("Upisite broj studenata > ");
    scanf("%d", &n);
    struct bodovi_s bodovi[n];
```

VLA polje

Rješenje (2. dio)

```
/* učitavanje bodova za n studenata */
for (i = 0; i < n; i = i + 1) {
    printf("Upisite podatke > ");
    scanf("%d %f %f", &bodovi[i].mbr, &bodovi[i].mi, &bodovi[i].zi);
    ukupno = bodovi[i].mi + bodovi[i].zi;

    // učitavanje bodova za lab. vježbe
    for (j = 0; j < BR_LAB_VJ; j = j + 1) {
        scanf("%f", &bodovi[i].lab[j]);
        ukupno = ukupno + bodovi[i].lab[j];
    }
    bodovi[i].ukupno = ukupno;
}
```

Rješenje (3. dio)

```
/* sortiranje prema vrijednosti ukupno */
int ind_max;
struct bodovi_s pomocna;
for (i = 0; i < n - 1; i = i + 1) {
    ind_max = i + 1;
    for (j = i + 2; j < n; j = j + 1) {
        if (bodovi[j].ukupno > bodovi[ind_max].ukupno) ind_max = j;
    }
    if (bodovi[ind_max].ukupno > bodovi[i].ukupno) {
        // zamijeni bodovi[i] i bodovi[ind_max]
        pomocna = bodovi[i];
        bodovi[i] = bodovi[ind_max];
        bodovi[ind_max] = pomocna;
    }
}
```

Rješenje (4. dio)

```
/* ispis rang liste */
printf("\nRang lista\n");
printf("Rbr.  Mat. broj    MI    ZI    LAB  Ukupno\n");
printf("=====");
for (i = 0; i < n; i = i + 1) {
    ukupno_lab = 0.f;
    for (j = 0; j < BR_LAB_VJ; j = j + 1) {
        ukupno_lab = ukupno_lab + bodovi[i].lab[j];
    }
    printf("%4d. %9d %5.1f %5.1f %5.1f %7.1f\n",
        i + 1, bodovi[i].mbr,
        bodovi[i].mi, bodovi[i].zi,
        ukupno_lab, bodovi[i].ukupno);
}
return 0;
}
```

Komentar rješenja

- Zašto rješenje u kojem bi se koristilo pet polja, umjesto polja čiji su članovi strukture, nije ispravno?

```
...  
int mbr[n];  
float mi[n];  
float zi[n];  
float ukupno[n];  
float lab[n][BR_LAB_VJ];  
...
```

- nije prepoznatljiva povezanost podataka u tim poljima
 - npr. nije odmah jasno da se vrijednosti mbr[0], mi[0], zi[0], ukupno[0] i redak polja lab s indeksom 0 odnose na istog studenta
- teže je baratati skupinama podataka
 - npr. napraviti kopiju vrijednosti svih podataka o jednom studentu



Prije sljedećeg predavanja

- Edgar:
 - Tutorial: **07. Prije osmog predavanja**
 - **7. vježbe uz predavanja**