Uvod u programiranje

- predavanja -

listopad 2020.

Agregatni tipovi podataka

- 2. dio -

Polja

Višedimenzijska polja

Višedimenzijsko polje

- Jednodimenzijsko polje (vektor)

```
int a[4];
```

a

a[0]	a[1]	a[2]	a[3]

- Polje može imati više dimenzija
  - dvije dimenzije (matrica, tablica)
    - npr. matrica od 3 retka i 5 stupaca

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

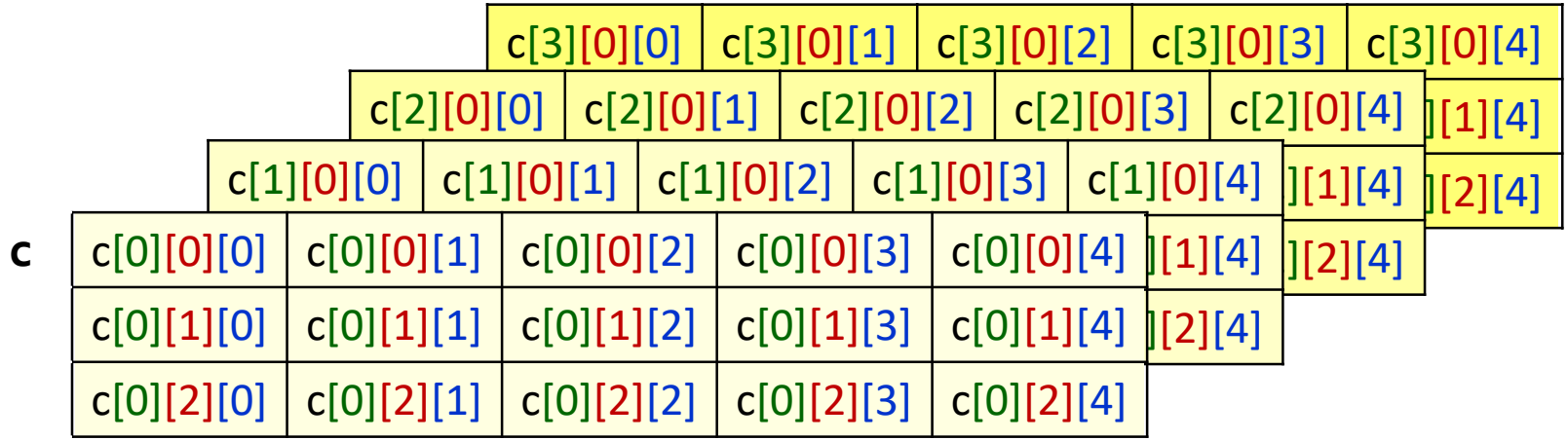
b

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]

Višedimenzijsko polje

- Polje može imati više dimenzija
  - tri dimenzije

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



  - broj dimenzija nije ograničen, npr.

```
int d[4][4][4][4][4][4][4][4][4][4][4][4][4][4];
```

  - oprez: ovo polje ima 4^{14} članova. Memorija?

Definicija polja uz inicijalizaciju

- Članovi polja mogu se inicijalizirati u trenutku definicije polja
  - nije primjenjivo za VLA polja!
  - bez inicijalizacije, članovi polja sadrže nedefinirane vrijednosti

```
int polje[3][4];
```

polje

?	?	?	?
?	?	?	?
?	?	?	?

  - početne vrijednosti se mogu redom navesti u tzv. inicijalizatoru

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

polje

1	2	3	4
5	6	7	8
9	10	11	12

Definicija polja uz inicijalizaciju

- Raspored vrijednosti po redcima bolje će se vidjeti ako se koristi sljedeći oblik inicijalizatora

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

polje

1	2	3	4
5	6	7	8
9	10	11	12

- navede li se premalo vrijednosti, ostali članovi se postavljaju na nulu

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

polje

1	2	0	0
5	6	7	0
0	0	0	0

- designiranim inicijalizatorom se članovi polja mogu ciljano postaviti

```
int polje[3][4] = {{1, 2},
                   {[2] = 7},
                   [2][1] = 11};
```

polje

1	2	0	0
0	0	7	0
0	11	0	0

Definicija polja uz inicijalizaciju

- automatsko određivanje veličine polja na temelju inicijalizatora moguće je samo za prvu dimenziju (sve ostale moraju biti navedene)

```
int polje[][4] = {{1, 2, 3, 4},
                  {5, 6, 7},
                  {9, 10}};
```

polje

1	2	3	4
5	6	7	0
9	10	0	0

- inače, prevodilac dojavljuje pogrešku

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

Prevodilac dojavljuje pogrešku

- u inicijalizatoru ne smije biti navedeno previše vrijednosti

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

Prevodilac dojavljuje pogrešku

Primjer

- Programski zadatak
  - Po retcima učitati vrijednosti članova dvodimenzijskog realnog polja od 4 retka i 5 stupaca (kolokvijalno: dimenzija 4 x 5). Sadržaj polja ispisati u obliku tablice

```
Upisite clanove polja >↵
-43.1 15 122.21 0.15 11↵
19.7 0.9761 54 33.7888 1↵
0 0 4.45 4.4 -45↵
28.1 28 6.721 -1 2↵
↵
  -43.10   15.00  122.21    0.15   11.00↵
   19.70    0.98   54.00   33.79    1.00↵
    0.00    0.00    4.45    4.40  -45.00↵
   28.10   28.00    6.72   -1.00    2.00↵
```

Rješenje (1. dio)

```
#include <stdio.h>
#define BR_RED  4                // broj redaka
#define BR_STUP 5                // broj stupaca

int main(void) {
   int redak, stupac;
   float polje[BR_RED][BR_STUP];

   /* ucitavanje vrijednosti clanova polja */
   printf("Upisite clanove polja >\n");
   for (redak = 0; redak < BR_RED; redak = redak + 1) {
      for (stupac = 0; stupac < BR_STUP; stupac = stupac + 1) {
         scanf("%f", &polje[redak][stupac]);
      }
   }
   /* ispis praznog retka nakon ucitavanja */
   printf("\n");
```

Rješenje (2. dio)

```
   /* ispis polja u obliku tablice */
   for (redak = 0; redak < BR_RED; redak = redak + 1) {
      for (stupac = 0; stupac < BR_STUP; stupac = stupac + 1) {
         printf("%8.2f", polje[redak][stupac]);
      }
      /* skok u novi red nakon ispisa jednog retka tablice */
      printf("\n");
   }
   return 0;
}
```

Primjer

- Programski zadatak
  - Učitati vrijednosti za broj redaka mr (ne smije biti veći od 10) i broj stupaca ms (ne smije biti veći od 20). Ponavljati učitavanje broja redaka i ponavljati učitavanje broja stupaca dok ne budu ispravni
  - Učitati vrijednosti članova dvodimenzijskog cjelobrojnog polja od mr redaka i ms stupaca.
  - U svakom retku polja pronaći najveći član i ispisati njegovu poziciju i vrijednost

Primjer

- primjer izvršavanja programa

```
Upisite broj redaka > -2↵
Upisite broj redaka > 11↵
Upisite broj redaka > 3↵
Upisite broj stupaca > 40↵
Upisite broj stupaca > 4↵
Upisite 3 x 4 cijelih brojeva >↵
1 2 4 -8↵
-8 -7 -5 -2↵
9 4 9 1↵
Najveci clanovi po retcima:↵
polje(0, 2) = 4↵
polje(1, 3) = -2↵
polje(2, 0) = 9↵
```

Rješenje (1. dio)

```
#include <stdio.h>
#define MAKS_RED  10              // najveci dopusteni broj redaka
#define MAKS_STUP 20              // najveci dopusteni broj stupaca

int main(void) {
   int i, j;            // kontrolne varijable za petlje
   int brred, brstup;   // stvarni broj redaka i stupaca

   /* Ucitavanje brred dok ne bude ispravan */
   do {
      printf("Upisite broj redaka > ");
      scanf("%d", &brred);
   } while (brred < 1 || brred > MAKS_RED);

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

Rješenje (2. dio)

```
/* definicija VLA polja dimenzija brred x brstup */
int polje[brred][brstup];

/* Ucitavanje clanova polja */
printf("Upisite %d x %d cijelih brojeva >\n", brred, brstup);

for (i = 0; i < brred; i = i + 1) {
   for (j = 0; j < brstup; j = j + 1) {
      scanf("%d", &polje[i][j]);
   }
}
```

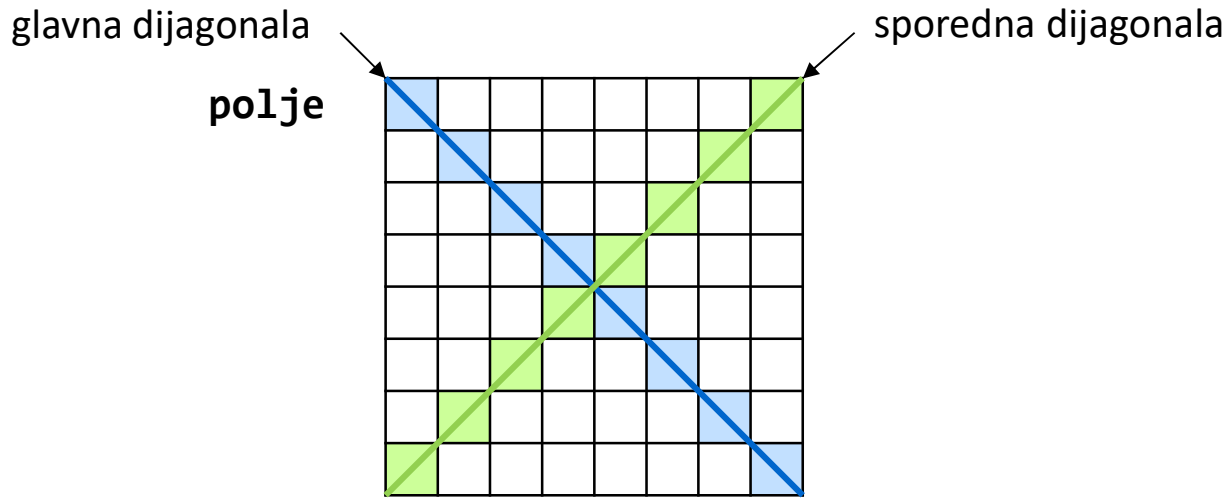
Rješenje (3. dio)

```
   /* pronadji i ispisi najveci clan u svakom retku */
   int stupacNajveceg;

   printf("Najveci clanovi po retcima:\n");
   for (i = 0; i < brred; i = i + 1) {
      /* pronadji indeks najveceg u retku i */
      stupacNajveceg = 0;         // pretpostavka: prvi je najveci
      for (j = 1; j < brstup; j = j + 1) {
         if (polje[i][j] > polje[i][stupacNajveceg]) {
            stupacNajveceg = j;  // promijeni pretpostavku
         }
      }
      printf("polje(%d, %d) = %d\n",
               i, stupacNajveceg, polje[i][stupacNajveceg]);
   }
   return 0;
}
```

Primjer

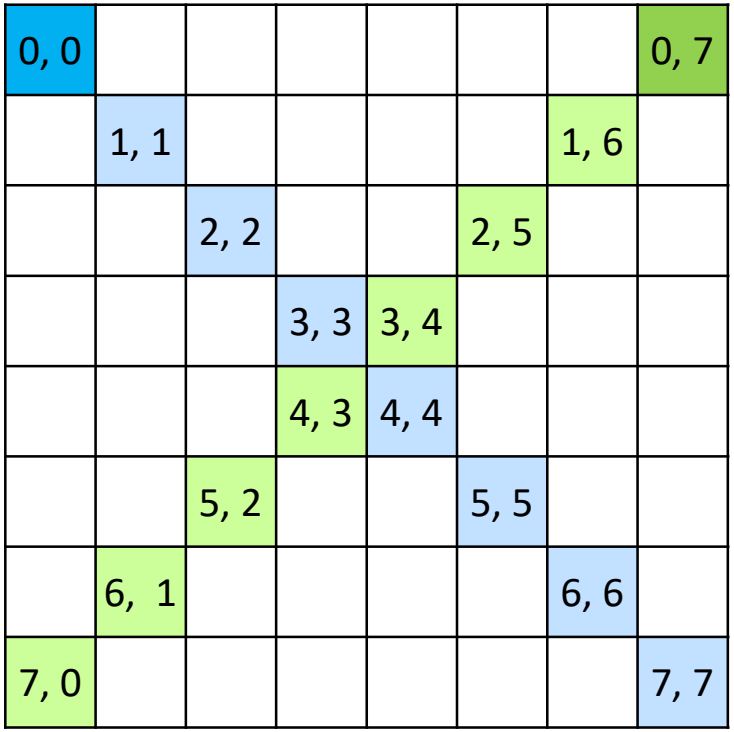
- Programski zadatak
  - Učitati vrijednosti (realni tip podataka) kvadratne matrice reda 8
    - kvadratna matrica je tablica koja ima jednak broj redaka i stupaca (red matrice)
  - Ispisati najmanju vrijednost na glavnoj dijagonali i najmanju vrijednost na sporednoj dijagonali
    - glavna i sporedna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

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
6	7	8	9	10
11	12	13	14	15

m

```
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
3	8	13
4	9	14
5	10	15

n

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];

   /* ucitaj 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)

```
/* ispisi 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 odgovarajuce 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];
   }
}
```

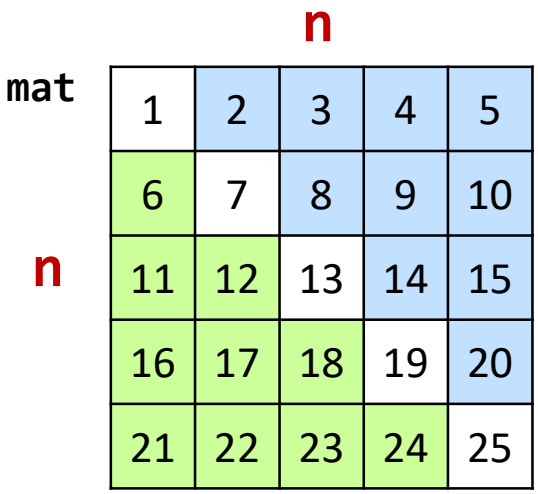
Rješenje (3. dio)

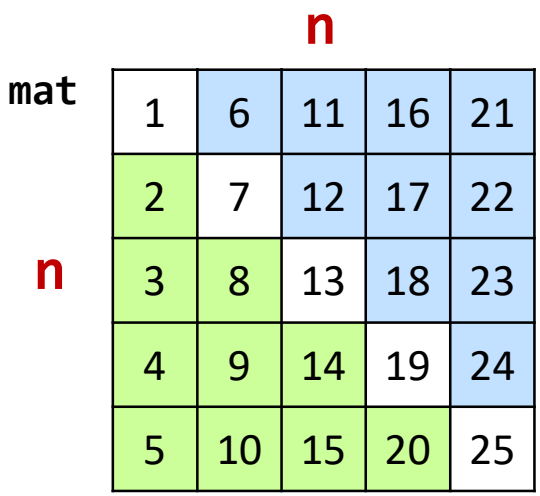
```
   /* ispisi matricu matT n x m */
   for (i = 0; i < n; i = i + 1) {
      for (j = 0; j < m; j = j + 1) {
         printf("%4d", matT[i][j]);
      }
      printf("\n");
   }
   return 0;
}
```

Primjer

- Programski zadatak
  - Učitati red kvadratne matrice n. Nije potrebno provjeravati ispravnost unosa
  - Učitati vrijednosti 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






```
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;
}
```

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 zavrsni ispit
float lab[8];     // broj bodova lab. vjezbe
```

    - 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;                          ispravno
struct pozicija_s tocka3;
tocka3 = tocka1;                          prevodilac dojavljuje pogrešku
```

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.

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)

```
/* ucitavanje 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;

   // ucitavanje bodova za lab. vjezbe
   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("=========================================\n");

   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