

Uvod u programiranje

- predavanja -

listopad 2020.

Agregatni tipovi podataka

- 1. dio -

Polja - motivacija

- Programski zadatak
 - s tipkovnice učitati 10 cijelih brojeva, ispisati njihovu aritmetičku sredinu i brojeve koji su veći od aritmetičke sredine
 - primjer izvršavanja programa

```
5 15 1 2 3 -4 25 6 8 7↵  
sredina = 6.800000↵  
15↵  
25↵  
8↵  
7↵
```

- očito, svih 10 vrijednosti će trebati pohraniti u varijable, izračunati prosjek, a zatim ispisati vrijednosti varijabli koje zadovoljavaju uvjet

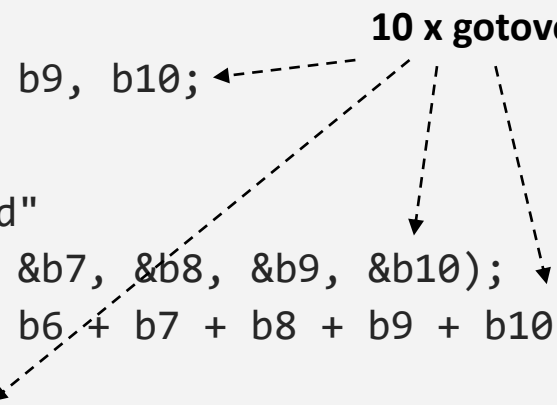
Rješenje (loše)

```
#include <stdio.h>

int main(void) {
    int b1, b2, b3, b4, b5, b6, b7, b8, b9, b10;
    float sredina;

    scanf("%d %d %d %d %d %d %d %d %d %d"
          , &b1, &b2, &b3, &b4, &b5, &b6, &b7, &b8, &b9, &b10);
    sredina = (b1 + b2 + b3 + b4 + b5 + b6 + b7 + b8 + b9 + b10) / 10.f;
    printf("sredina = %f\n", sredina);
    if (b1 > sredina) printf("%d\n", b1);
    if (b2 > sredina) printf("%d\n", b2);
    ...
    if (b9 > sredina) printf("%d\n", b9);
    if (b10 > sredina) printf("%d\n", b10);
    return 0;
}
```

10 x gotovo isti posao



Matematički niz

- Niz brojeva koji imaju zajedničko ime i čiji se članovi identificiraju indeksom: $\text{broj}_0, \text{broj}_1, \text{broj}_2, \dots$
- Kada bi se niz mogao koristiti u programu, prethodni zadatak bi se mogao riješiti na sljedeći način:

```
suma := 0
za i = 1 do 10
    | učitaj( $b_i$ )
    | suma := suma +  $b_i$ 
sredina := suma / 10
za i = 1 do 10
    | ako je  $b_i > \text{sredina}$ 
    | | ispiši( $b_i$ )
```

Matematički niz

- Bez teškoća bi se mogli riješiti i mnogo veći problemi, koje bi bilo iznimno teško riješiti bez korištenja niza: npr. učitavanje n brojeva i ispis onih brojeva koji su veći od njihove aritmetičke sredine

```
učitaj(n)
suma := 0
za i = 1 do n
    | učitaj( $b_i$ )
    | suma := suma +  $b_i$ 
sredina := suma / n
za i = 1 do n
    | ako je  $b_i >$  sredina
    | | ispiši( $b_i$ )
```

Agregatni tipovi podataka

Polja
Strukture

Agregatni tipovi podataka

- do sada su korišteni isključivo jednostavni tipovi podataka: `int` i `float`
 - u varijablu jednostavnog tipa moguće je pohraniti samo jedan elementarni podatak. Jedna varijabla \leftrightarrow jedna vrijednost
 - **skalarni** tip podatka, **skalarna** varijabla, **skalarni** podatak
- agregatni podatak (*data aggregate*) ili složeni podatak obuhvaća više skalarnih podataka i/ili više agregatnih podataka, objedinjenih pod istim imenom
 - polje (*array*)
 - struktura ili zapis (*structure, record*)
- prednosti korištenja agregatnih podataka
 - jednostavniji postupci nad skupinama podataka
 - naglašava se logička povezanost podataka

Polja

Jednodimenzijska polja

Polje

- **Jednostavni tipovi podataka**

- jedna varijabla \leftrightarrow jedna vrijednost
- do vrijednosti varijable pristupa se navođenjem imena varijable
- varijabla jednostavnog tipa jest *modifiable lvalue*

```
float x;  
x = 3.14f;  
printf("%f", x);
```

x 3.14

- **Polje** je složeni tip podatka koji obuhvaća više članova istog tipa

- jedna varijabla \leftrightarrow više vrijednosti
- pojedinim vrijednostima (članovima) pristupa se pomoću indeksa
- ime varijable je *non-modifiable lvalue*; ime varijable uz navedeni indeks elementa jest *modifiable lvalue*

```
float y[4];  
y[1] = 2.71f;  
printf("%f", y[1]);
```

y

?	2.71	?	?
---	------	---	---

 y[0] y[1] y[2] y[3]

Definicija varijable tipa polje

- Pri definiciji varijable potrebno je odrediti
 - ime varijable: određuje se na isti način kao ime varijable za jednostavne tipove podataka
 - tip podatka za članove polja (int, float, ...)
 - veličinu polja izraženu u broju članova polja
 - polje poznatih konstantnih dimenzija
 - dimenzija polja poznata je u trenutku prevođenja
 - u uglatim zagradama navodi se **konstantni cjelobrojni izraz**
 - polje varijabilne veličine (*variable-length array, VLA*)
 - veličina polja se utvrđuje u trenutku izvršavanja naredbe za definiciju polja (ali nakon toga se ne mijenja)
 - u uglatim zagradama navodi se **cjelobrojni izraz** (može sadržavati varijable) čiji rezultat **mora** biti veći od nule

Primjer

```
#define VELICINA_VEKTORA 50
#define BROJ_CLANOVA (5 * 10)
...
// primjeri definicije polja poznatih konstantnih dimenzija
//float vektor[50];
float vektor[VELICINA_VEKTORA]; // bolje nego 50

//int velicine[5 * 10];
int velicine[BROJ_CLANOVA]; // bolje nego 5 * 10

// primjeri definicije polja varijabilnih dimenzija (VLA)
int n;
scanf("%d", &n); // osigurati da n bude > 0 !
float temperature[n];
int tlakovi[2 * n];
```

Pristupanje članovima polja

- Članovima (elementima) polja pristupa se korištenjem indeksa
 - indeks može biti cjelobrojni izraz (konstante, varijable, operatori, funkcije) čiji rezultat mora biti nenegativni cijeli broj iz intervala
 $[0, \text{brojElemenataPolja} - 1]$ (znači: indeks prvog člana je 0)
 - C prevodilac može utvrditi jedino pogrešnu upotrebu tipa podatka indeksa (npr. float umjesto int)
 - poštovanje pravila o dopuštenim granicama indeksa odgovornost je isključivo programera
 - rezultat korištenja neispravnih indeksa za vrijeme izvršavanja programa je nedefiniran: dobije se pogrešna vrijednost (logička pogreška), a pridruživanje vrijednosti uz korištenje neispravnog indeksa može, osim logičke pogreške, dovesti i do pogreške izvršavanja i prekida programa

Primjer

```
int i = 2, m, brojevi[10];
```

```
float x = 5.0f;
```

```
brojevi[0] = 0;
```

```
brojevi[1] = 10;
```

```
...
```

```
brojevi[i * 4] = 80;
```

```
brojevi[i * 4 + 1] = 90;
```

```
brojevi[x] = 1;    prevodilac dojavljuje pogrešku
```

```
...
```

```
m = brojevi[10];   rezultat je nedefiniran (garbage value). Logička pogreška.
```

```
brojevi[-1] = 15;  15 se upisuje na pogrešno mjesto u memoriji, što može  
izazvati logičku pogrešku ili pogrešku tijekom izvršavanja
```

Primjer

- s tipkovnice učitati 10 cijelih brojeva, ispisati njihovu aritmetičku sredinu i brojeve koji su veći od aritmetičke sredine

```
#include <stdio.h>
#define DIMENZIJA 10

int main(void) {
    int brojevi[DIMENZIJA], suma = 0, i;
    float sredina;

    for (i = 0; i < DIMENZIJA; i = i + 1) {
        scanf("%d", &brojevi[i]);
        suma = suma + brojevi[i];
    }
    sredina = 1.f * suma / DIMENZIJA;
    printf("sredina = %f\n", sredina);

    for (i = 0; i < DIMENZIJA; i = i + 1) {
        if (brojevi[i] > sredina) {
            printf("%d\n", brojevi[i]);
        }
    }
    return 0;
}
```

Primjer (kada koristiti VLA)

- s tipkovnice učitati n, zatim n cijelih brojeva, ispisati njihovu aritmetičku sredinu i brojeve koji su veći od aritmetičke sredine

```
...
int n, suma = 0, i;
float sredina;

scanf("%d", &n);
int brojevi[n];
for (i = 0; i < n; i = i + 1) {
    scanf("%d", &brojevi[i]);
    suma = suma + brojevi[i];
}
sredina = 1.f * suma / n;
printf("sredina = %f\n", sredina);

for (i = 0; i < n; i = i + 1) {
    if (brojevi[i] > sredina) {
        printf("%d\n", brojevi[i]);
    }
}
...
```

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[5];
```

polje

?	?	?	?	?
---	---	---	---	---

- početne vrijednosti se mogu redom navesti u tzv. inicijalizatoru

```
int polje[5] = {1, 3, 5, 7, 9};
```

polje

1	3	5	7	9
---	---	---	---	---

- ako se navede premalo vrijednosti, ostali članovi se postavljaju na 0

```
int polje[5] = {1, 3, 5};
```

polje

1	3	5	0	0
---	---	---	---	---

- što se može iskoristiti za inicijalizaciju svih članova na nulu

```
int polje[5] = {0};
```

polje

0	0	0	0	0
---	---	---	---	---

Definicija polja uz inicijalizaciju

- automatsko određivanje veličine polja na temelju inicijalizatora

```
int polje[] = {1, 3, 5, 7, 9};
```

polje

1	3	5	7	9
---	---	---	---	---

- designiranim inicijalizatorom se članovi polja mogu ciljano postaviti

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

polje

1	0	0	0	2
---	---	---	---	---

- u inicijalizatoru se mora nalaziti barem jedna vrijednost

```
int polje[5] = {};
```

Prevodilac dojavljuje pogrešku

- u inicijalizatoru se ne smije napisati previše vrijednosti

```
int polje[5] = {1, 3, 5, 7, 9, 11};
```

Prevodilac dojavljuje pogrešku

Inicijalizacija nije pridruživanje!

- varijabla tipa polje ne smije se nalaziti na lijevoj strani izraza pridruživanja jer polje nije *modifiable lvalue*
 - neispravan način kopiranja sadržaja polja izvor u polje cilj:

```
int izvor[4] = {1, 2, 3, 4};  
int cilj[4];  
cilj = izvor;
```

inicijalizacija polja izvor. O.K.

prevodilac dojavljuje pogrešku!

- ispravan način kopiranja sadržaja polja izvor u polje cilj:

```
int izvor[4] = {1, 2, 3, 4};  
int cilj[4];  
int i;  
for (i = 0; i < 4; i = i + 1) {  
    cilj[i] = izvor[i];  
}
```

inicijalizacija polja izvor. O.K.

Primjer

- Programski zadatak
 - s tipkovnice učitati cijeli broj n koji predstavlja broj članova polja koji će biti učitani. Ponavljati učitavanje vrijednosti za n sve dok broj članova polja ne bude ispravan. Zatim učitati n realnih članova polja i ispisati ih u obrnutom poretaku od onog u kojem su učitani
 - primjer izvršavanja programa

```
Upisite broj clanova polja > 0↵
Upisite broj clanova polja > -2↵
Upisite broj clanova polja > 4↵
Upisite 1. clan > 9.1↵
Upisite 2. clan > 101.55↵
Upisite 3. clan > -476.3333↵
Upisite 4. clan > 5↵
5.0, -476.3, 101.6, 9.1↵
```

Rješenje (1. dio)

```
#include <stdio.h>

int main(void) {
    int n;    // velicina polja
    int i;    // kontrolna varijabla petlje za učitavanje i ispis
    /* ponavljati upisivanje velicine polja dok ne bude ispravna */
    do {
        printf("Upisite broj clanova polja > ");
        scanf("%d", &n);
    } while (n < 1);

    /* definicija VLA polja */
    float polje[n];
}
```

Rješenje (2. dio)

```
/* učitavanje članova polja */
for (i = 0; i < n; i = i + 1) {
    printf("Upisite %d. clan > ", i + 1);
    scanf("%f", &polje[i]);
}

/* ispisivanje članova polja u obrnutom poretaku */
for (i = n - 1; i >= 0; i = i - 1) {
    if (i < n - 1) {        // ispisati zarez prije svakog osim prvog
        printf(", ");
    }
    printf("%.1f", polje[i]);
}
printf("\n");
return 0;
}
```

Primjer

- Programski zadatak

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

```
Upisite broj iz intervala [0, 9] > 1↵
Upisite broj iz intervala [0, 9] > 5↵
Upisite broj iz intervala [0, 9] > 7↵
Upisite broj iz intervala [0, 9] > 5↵
Upisite broj iz intervala [0, 9] > 0↵
Upisite broj iz intervala [0, 9] > 5↵
Upisite broj iz intervala [0, 9] > 7↵
Upisite broj iz intervala [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

- Utvrđivanje frekvencije pojavljivanja brojeva
 - za svaki broj potreban je jedan brojač
 - na početku se svi brojači postavljaju na nulu
 - kad god se učitava neki broj, odgovarajući brojač se poveća za 1

brojac	0	0	0	0	0	0	0	0	0
	brojac[0]	brojac[1]	brojac[2]	brojac[3]	brojac[4]	brojac[5]	brojac[6]	brojac[7]	brojac[8]

```
Upisite broj iz intervala [0, 9] > 1↵ → brojac[1] = brojac[1] + 1
Upisite broj iz intervala [0, 9] > 5↵ → brojac[5] = brojac[5] + 1
Upisite broj iz intervala [0, 9] > 7↵ → brojac[7] = brojac[7] + 1
Upisite broj iz intervala [0, 9] > 5↵ → brojac[5] = brojac[5] + 1
Upisite broj iz intervala [0, 9] > 0↵ → brojac[0] = brojac[0] + 1
Upisite broj iz intervala [0, 9] > 5↵ → brojac[5] = brojac[5] + 1
Upisite broj iz intervala [0, 9] > 7↵ → brojac[7] = brojac[7] + 1
```

brojac	1	1	0	0	0	3	0	2	0	0
	brojac[0]	brojac[1]	brojac[2]	brojac[3]	brojac[4]	brojac[5]	brojac[6]	brojac[7]	brojac[8]	brojac[9]

Rješenje (1. dio)

```
#include <stdio.h>
#define D_GR 0          // donja granica intervala
#define G_GR 9          // gornja granica intervala

int main(void) {
    int broj;
    int brojac[G_GR - D_GR + 1] = { 0 };    // inicijalizacija na nulu
    /* učitavanje brojeva i inkrementiranje odgovarajućih brojaca */
    do {
        printf("Upisite broj u intervalu [%d, %d] > ", D_GR, G_GR);
        scanf("%d", &broj);
        if (broj >= D_GR && broj <= G_GR) {
            brojac[broj] = brojac[broj] + 1;
        }
    } while (broj >= D_GR && broj <= G_GR);
```


Rješenje (2. dio)

```
printf("\n");  
/* ispis sadržaja onih brojaca koji su veci od nule */  
int i;  
for (i = D_GR; i <= G_GR; i = i + 1) {  
    if (brojac[i] > 0) {  
        printf("Broj %d se pojavio %d puta\n", i, brojac[i]);  
    }  
}  
return 0;  
}
```

Primjer

- Programski zadatak (varijanta prethodnog zadatka - promijenjene su samo granice intervala)
 - Učitavati cijele brojeve iz intervala [1000005, 1000014]. Učitavanje prekinuti kad se upiše broj izvan zadanog intervala. Zatim ispisati koliko je puta učitani svaki broj iz zadanog intervala, pri čemu treba ispisati samo one brojeve koji su učitani barem jednom.

```
Upisite broj iz intervala [1000005, 1000014] > 1000008↵
Upisite broj iz intervala [1000005, 1000014] > 1000012↵
Upisite broj iz intervala [1000005, 1000014] > 1000012↵
Upisite broj iz intervala [1000005, 1000014] > 1000008↵
Upisite broj iz intervala [1000005, 1000014] > 1000012↵
Upisite broj iz intervala [1000005, 1000014] > 1000014↵
Upisite broj iz intervala [1000005, 1000014] > 1000000↵
↵
Broj 1000008 se pojavio 2 puta↵
Broj 1000012 se pojavio 3 puta↵
Broj 1000014 se pojavio 1 puta↵
```

Rješenje (1. dio)

```
#include <stdio.h>
#define D_GR 1000005           // donja granica intervala
#define G_GR 1000014         // gornja granica intervala

int main(void) {
    int broj;
    int brojac[G_GR - D_GR + 1] = { 0 };    // inicijalizacija na nulu
    /* učitavanje brojeva i inkrementiranje odgovarajućih brojaca */
    do {
        printf("Upisite broj u intervalu [%d, %d] > ", D_GR, G_GR);
        scanf("%d", &broj);
        if (broj >= D_GR && broj <= G_GR) {
            brojac[broj - D_GR] = brojac[broj - D_GR] + 1;
        }
    } while (broj >= D_GR && broj <= G_GR);
}
```

Rješenje (2. dio)

```
printf("\n");  
/* ispis sadržaja onih brojaca koji su veci od nule */  
int i;  
for (i = D_GR; i <= G_GR; i = i + 1) {  
    if (brojac[i - D_GR] > 0) {  
        printf("Broj %d se pojavio %d puta\n", i, brojac[i - D_GR]);  
    }  
}  
return 0;  
}
```

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
 - 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 , $0 \leq i \leq n-2$
 - pronadi indeks ind_min najmanjeg člana $\text{polje}[j]$, $i < j \leq n-1$
 - ako je $\text{polje}[\text{ind_min}] < \text{polje}[i]$, zamijeni $\text{polje}[i]$ i $\text{polje}[\text{ind_min}]$

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 učitavanje n i n članova 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 članova polje[i] i polje[ind_min] */
        pomocna = polje[i];
        polje[i] = polje[ind_min];
        polje[ind_min] = pomocna;
    }
}

/* izostavljen uobicajeni kod za ispis */
```