

Sadržaj

- Ponavljanje onog dijela C-a koji je napravljen na Prog1:
 - Izrazi.
 - Eksplicitna konverzija tipova.
 - Konverzije iz double u int — greške zaokruživanja.
 - Pokazivači.
 - Funkcije.
 - Polja.
 - Pokazivači i polja.
 - Polje kao argument funkcije.
 - Rekurzivne funkcije.

Izrazi (1)

Primjer. Što će ispisati sljedeći odsječak programa?

```
int a = 10, b = 10, c = 2;
```

```
a /= c * 5;
```

```
b = b / c * 5;
```

```
printf("a - b = %d\n", a - b);
```

Izrazi (1) — rješenje

Primjer. Što će ispisati sljedeći odsječak programa?

```
int a = 10, b = 10, c = 2;
```

```
a /= c * 5;
```

```
b = b / c * 5;
```

```
printf("a - b = %d\n", a - b);
```

a - b = -24

Izrazi (2)

Primjer. Što će ispisati sljedeći odsječak programa?

```
int a = b = 10, c = 2;
```

```
a = c * b;
```

```
printf("b = %d\n", b);
```

Izrazi (2) — rješenje

Primjer. Što će ispisati sljedeći odsječak programa?

```
int a = b = 10, c = 2;
```

```
a = c * b;
```

```
printf("b = %d\n", b);
```

greška pri kompajliranju:

error: identifier "b" is undefined

```
int a = b = 10, c = 2;  
      ^
```

Izrazi (3)

Primjer. Što će ispisati sljedeći odsječak programa?

```
int b, a = b = 10, c = 2;
```

```
a = c * b;
```

```
printf("b = %d\n", b);
```

Izrazi (3) — rješenje

Primjer. Što će ispisati sljedeći odsječak programa?

```
int b, a = b = 10, c = 2;
```

```
a = c * b;
```

```
printf("b = %d\n", b);
```

b = 10

Usput, a = 20.

Izrazi (4)

Primjer. Što će ispisati sljedeći odsječak programa?

```
int i, j, k;  
  
k = 0;  i = 3;  j = 2;  
if (i - i && j++) k = 1;  
  
printf("k = %d\n", k);
```

Pitanje: kolika je konačna vrijednost varijable **j**?

Izrazi (4) — rješenje

Primjer. Što će ispisati sljedeći odsječak programa?

```
int i, j, k;
```

```
k = 0;  i = 3;  j = 2;  
if (i - i && j++) k = 1;
```

```
printf("k = %d\n", k);
```

```
k = 0
```

Odgovor: j = 2.

Izrazi (5)

Primjer. Što će ispisati sljedeći odsječak programa?

```
int i, j, k;  
  
k = 0;  i = 3;  j = 0;  
if (i + i && j++) k = 1;  
  
printf("k = %d\n", k);
```

Pitanje: kolika je konačna vrijednost varijable **j**?

Izrazi (5) — rješenje

Primjer. Što će ispisati sljedeći odsječak programa?

```
int i, j, k;  
  
k = 0;  i = 3;  j = 0;  
if (i + i && j++) k = 1;  
  
printf("k = %d\n", k);
```

k = 0

Odgovor: j = 1.

Eksplícitna konverzija (operator cast)

Primjer. Što će ispisati sljedeći odsječak programa?

```
int z = 5;  double y = 5.8;
```

```
printf("%d\n", (int) y/2);
```

```
printf("%d\n", (int) (double) z/2);
```

```
printf("%f\n", (float) '1');
```

Eksplícitna konverzija (operator cast) — rješ.

Primjer. Što će ispisati sljedeći odsječak programa?

```
int z = 5;  double y = 5.8;  
  
printf("%d\n", (int) y/2);  
printf("%d\n", (int) (double) z/2);  
printf("%f\n", (float) '1');
```

2

2

49.000000

Eksplícitna konverzija (2)

Primjer. Što će ispisati sljedeći odsječak programa?

```
double y = 1.8e+20;
```

```
printf ("%d\n", (int) y/2);
```

Eksplicitna konverzija (2) — rješenje

Primjer. Što će ispisati sljedeći odsječak programa?

```
double y = 1.8e+20;
```

```
printf ("%d\n", (int) y/2);
```

-1073741824

(oprez: ovisi o kompajleru)

rezultat se ne mijenja povećanjem eksponenta od y

Konverzija double u int

Primjer. Što će ispisati sljedeći odsječak programa?

```
double x = 5.1;
```

```
printf("%d\n", (int)(1000*x));
```


Konverzija double u int — rješenje

Primjer. Što će ispisati sljedeći odsječak programa?

```
double x = 5.1;
```

```
printf("%d\n", (int)(1000*x));
```

5100

Oprez: Rezultat ovisi o greškama zaokruživanja u realnoj aritmetici (tip double)

🎯 i “pukim slučajem” je točan!

Na drugom računalu (i prevoditelju), kolega je dobio rezultat 5099

Konverzija double u int (2)

Primjer. Što će ispisati sljedeći odsječak programa?

```
double x = 64.1;
```

```
printf("%d\n", (int)(1000*x));
```

Vjerojatno očekujete 64100.

Konverzija double u int (2) — rješenje

Primjer. Što će ispisati sljedeći odsječak programa?

```
double x = 64.1;
```

```
printf("%d\n", (int)(1000*x));
```

Vjerojatno očekujete 64100.

Međutim, nije! Stvarni rezultat je

64099

Zašto?

Razlog: Greške zaokruživanja, i to dvije!

Konverzija double u int (2) — objašnjenje

Prva nastaje kod prikaza broja $x = 64.1$ u tipu double.

- Naime, broj nije egzaktan prikaz u binarnom sustavu (ima beskonačan prikaz).

Prikaz broja $x = 64.100$ u racunalu:

1. riječ: 0110 0110 0110 0110 0110 0110 0110 0110
 2. riječ: 0100 0000 0101 0000 0000 0110 0110 0110
-

Druga riječ sadrži bitove 63–32, tj. počinje predznakom — 0,

- pa onda ide 11 bitova karakteristike — 100 0000 0101,
- a zatim idu bitovi mantise (bez vodeće jedinice).

Prva riječ sadrži bitove 31–0, tj. kraj mantise (periodičnost “pravog” binarnog zapisa se vidi).

Konverzija double u int (2) — objašnjenje

Druga greška nastaje prilikom množenja $1000 \cdot x$.

- Zaokruženi rezultat u double je malo manji od 64100, pa (int) zaokruži nadalje.

Prikaz broja $1000 \cdot x = 64100.000$ u racunalu:

- 1. rijec: 1111 1111 1111 1111 1111 1111 1111 1111
 - 2. rijec: 0100 0000 1110 1111 0100 1100 0111 1111
-

Zanemarite to što gore piše 64100.000.

Format ispisa je `%.3f`. On, također, zaokružuje! Probajte format s više decimala.

Poopćenje — Konverzija double u int (3)

Primjer. Što će ispisati sljedeći odsječak programa?

```
int c = 1, i;  
  
for (i = 0; i < 20; ++i) {  
    double x = c + 0.1;  
  
    printf("%d\n", (int)(1000*x));  
    c *= 2;  
}
```

Idemo redom. Nađimo **vrijednosti** varijabli **c** i **x**, u **ovisnosti** o “indeksu” **i**,

 koji se **mijenja** u petlji, od 0 do 19.

Konverzija double u int (3) — nastavak

```
int c = 1, i;  
for (i = 0; i < 20; ++i) {  
    double x = c + 0.1;  
    printf("%d\n", (int)(1000*x));  
    c *= 2; }
```

Na početku je $c = 1$, a zatim se c množi s 2, na dnu petlje. Zato u deklaraciji $\text{double } x = c + 0.1$; vrijedi da je $c = 2^i$, tj. vrijednosti varijable x su

$$x = 2^i + 0.1, \quad i = 0, \dots, 19.$$

Iza toga, računamo (i pišemo) vrijednost cjelobrojnog izraza

$$(\text{int})(1000*x) = \lfloor 1000 \cdot x \rfloor.$$

Konverzija double *u* int (3) — nastavak

```
int c = 1, i;  
for (i = 0; i < 20; ++i) {  
    double x = c + 0.1;  
    printf("%d\n", (int)(1000*x));  
    c *= 2; }
```

Dakle, ovaj dio programa **računa** (i piše) vrijednost izraza

$$\lfloor 1000 \cdot (2^i + 0.1) \rfloor, \quad i = 0, \dots, 19.$$

Matematički ekvivalentno je

$$1000 \cdot 2^i + 100, \quad i = 0, \dots, 19,$$

što je (očito) **cijeli** broj — **djeljiv** sa 100.

Konverzija double u int (3) — rezultati

Rezultati: za $x = 2^i + 0.1$, $i = 0, \dots, 9$, dobivamo

i	(int)(1000*x)
0	1 100
1	2 100
2	4 100
3	8 100
4	16 100
5	32 100
6	64 099
7	128 100
8	256 100
9	512 100

Konverzija double u int (3) — rezultati (nast.)

Rezultati: za $x = 2^i + 0.1$, $i = 10, \dots, 19$, dobivamo

i	(int)(1000*x)
10	1 024 099
11	2 048 100
12	4 096 100
13	8 192 100
14	16 384 099
15	32 768 100
16	65 536 100
17	131 072 100
18	262 144 099
19	524 288 100

Konverzija double u int (3) — objašnjenje

Neki rezultati su **pogrešni**! Na primjer,

☞ Za $i = 6$, umjesto 64 100, dobivamo 64 099.

To je **identično** kao u prošlom primjeru.

Razlog: Greška **zaokruživanja** kod računanja/prikaza broja $x = 2^i + 0.1$ u tipu **double**.

Zadatak. Iz **prikaza** brojeva **zaključite** za koje vrijednosti i dobivamo **točne**, odnosno, **pogrešne** rezultate, uz pretpostavku

- ☞ **pravilnog** zaokruživanja,
- ☞ zaokruživanja **nadolje** (prema 0),
- ☞ zaokruživanja **nagore** (prema ∞).

Pokazivači

Primjer. Što će ispisati sljedeći odsječak programa?

```
int a = 1;
```

```
int *b;
```

```
b = &a;
```

```
*b = 5;
```

```
printf("%d %d\n", a, *b);
```

Pokazivači — rješenje

Primjer. Što će ispisati sljedeći odsječak programa?

```
int a = 1;
```

```
int *b;
```

```
b = &a;
```

```
*b = 5;
```

```
printf("%d %d\n", a, *b);
```

5 5

Funkcije

Primjer. Što će ispisati sljedeći program?

```
#include <stdio.h>

int main(void)
{
    int m, nzm(int, int);

    m = nzm(36, 48);
    printf("mjera = %d\n", m);

    return 0;
}
```

Funkcije (nastavak)

pri čemu je:

```
int nzm(int a, int b)
{
    while (a != b)
        if (a > b)
            a -= b;
        else
            b -= a;
    return a;
}
```

Funkcije (nastavak) — rješenje

pri čemu je:

```
int nzm(int a, int b)
{
    while (a != b)
        if (a > b)
            a -= b;
        else
            b -= a;
    return a;
}
```

mjera = 12

Primjer. Što će ispisati sljedeći program?

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

```
int main(void) {  
    int array[] = {4, 5, -8, 9, 8, 0, -2, 1, 9, 3};  
    int index;  
    index = 0;  
    while (array[index] != 0)  
        ++index;  
    printf("Broj elemenata polja prije nule: %d\n",  
        index);  
    return 0; }
```

Polja — rješenje

Primjer. Što će ispisati sljedeći program?

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

```
int main(void) {  
    int array[] = {4, 5, -8, 9, 8, 0, -2, 1, 9, 3};  
    int index;  
    index = 0;  
    while (array[index] != 0)  
        ++index;  
    printf("Broj elemenata polja prije nule: %d\n",  
        index);  
    return 0; }
```

Broj elemenata polja prije nule: 5

Pokazivači i polja

Zapamtiti: Ime polja je sinonim za

- konstantni pokazivač koji sadrži adresu prvog elementa polja

Polje može biti formalni (i stvarni) argument funkcije. U tom slučaju:

- ne prenosi se cijelo polje po vrijednosti (kopija polja!),
- već funkcija dobiva (po vrijednosti) pokazivač na prvi element polja.

Unutar funkcije elementi polja mogu se

- dohvatiti i promijeniti, korištenjem indeksa polja.

Razlog: aritmetika pokazivača (v. sljedeću stranicu).

Pokazivači i polja — nastavak

Primjer. Krenimo od deklaracija

```
int a[10], *pa;
```

Tada je: $a = \&a[0]$. Ne samo to, pokazivaču mogu **dodati** i **oduzeti** “indeks” (tzv. “**aritmetika pokazivača**”).

Općenito vrijedi: $a + i = \&a[i]$, gdje je i neki cijeli broj.

```
*(a + 1) = 10;    /* ekv. s a[1] = 10; */  
...  
pa = a;           /* ekv. s pa = &a[0]; */  
pa = pa + 2;      /* &a[2] */  
pa++;             /* &a[3] */  
*(pa + 3) = 20;   /* ekv. s a[6] = 20; */
```

Pokazivači i polja (1)

Primjer. Što će ispisati sljedeći program?

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

```
int main(void) {  
    int array[] = {4, 5, -8, 9, 8, 0, -2, 1, 9, 3};  
    int *array_ptr;  
    array_ptr = array;  
    while ((*array_ptr) != 0)  
        ++array_ptr;  
    printf("Broj elemenata polja prije nule: %d\n",  
          array_ptr - array);  
    return 0; }
```

Pokazivači i polja (1) — rješenje

Primjer. Što će ispisati sljedeći program?

```
#include <stdio.h>

int main(void) {
    int array[] = {4, 5, -8, 9, 8, 0, -2, 1, 9, 3};
    int *array_ptr;
    array_ptr = array;
    while ((*array_ptr) != 0)
        ++array_ptr;
    printf("Broj elemenata polja prije nule: %d\n",
           array_ptr - array);
    return 0; }
```

Broj elemenata polja prije nule: 5

Pokazivači i polja (2)

Primjer. Što će ispisati sljedeći program?

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

int main(void) {
    int a[MAX];
    int i, *p;
    p = a;
    for (i = 0; i < MAX; ++i)
        a[i] = i;
    printf("%d\n", *p);
    return 0; }
```

Pokazivači i polja (2) — rješenje

Primjer. Što će ispisati sljedeći program?

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

int main(void) {
    int a[MAX];
    int i, *p;
    p = a;
    for (i = 0; i < MAX; ++i)
        a[i] = i;
    printf("%d\n", *p);
    return 0; }
```

0

Polje kao argument funkcije

Primjer. Napišite funkcije `unos` i `ispis` te glavni program koji upisuje i ispisuje polje s maksimalno 100 elemenata.

```
#include <stdio.h>
#define MAX 100

void unos(int a[], int n) {
    int i;
    for (i = 0; i < n; ++i)
        scanf("%d", &a[i]); }

void ispis(int *a, int n) {
    int i;
    for (i = 0; i < n; ++i)
        printf("%d\n", *a++); }
```

Polje kao argument funkcije (nastavak)

```
int main(void) {  
    int n, polje[MAX];  
    /* Koliko ce se bajtova rezervirati? */  
  
    scanf("%d", &n);  
  
    unos(polje, n);  
    ispis(polje, n);  
    return 0;  
}
```

Za `polje` se rezervira $100 * 4 = 400$ bajtova.

Primjedba: Pri upisu podataka oni se “upisuju na slijepo” (ne zna se što se upisuje) – loš stil programiranja.

Rekurzivne funkcije

Primjer. Što će ispisati sljedeći program?

```
#include <stdio.h>
int f(int n) {
    if (n == 0)
        return 2;
    else
        return f(--n); }
int main(void) {
    printf("%d\n", f(4));
    return 0; }
```

Rekurzivne funkcije — rješenje

Primjer. Što će ispisati sljedeći program?

```
#include <stdio.h>
int f(int n) {
    if (n == 0)
        return 2;
    else
        return f(--n); }
int main(void) {
    printf("%d\n", f(4));
    return 0; }
```

2

Pitanje: što se ispiše ako napišemo `f(n--)`? Oprez! Zašto?