

# Programiranje 2

## Predavanje 01 - ponavljanje

Matej Mihelčič

Prirodoslovno-matematički fakultet  
Matematički odsjek

04. ožujka 2024.



Što će ispisati sljedeći odsječci programa?

```
1 int a = 20, b = 20, c = 4;  
2 a/=c*5;  
3 b=b/c*5;  
4  
5 printf("a_ b=_%d\n", a-b);
```

```
1 int a = b = 20, c = 4;  
2 a = c*b;  
3 printf("b_=%d\n",b);
```

Što će ispisati sljedeći odsječci programa?

```
1 int b, a = b = 20, c = 4;  
2 a = c * b;  
3  
4 printf("b_□=□%d\n", b);
```

```
1 int i, j, k;  
2 k = 0; i = 3; j = 2;  
3 if(i-i && j++) k = 1;  
4  
5 printf("k_□=□%d\n", k);
```

Kolika je konačna vrijednost varijable j?

Što će ispisati sljedeći odsječak programa?

```
1 int i,j,k;
2 k = 0; i = 3; j = 0;
3 if(i+i && j++) k = 1;
4
5 printf("k=□%d\n", k);
```

Kolika je konačna vrijednost varijable  $j$ ?

## Eksplicitna konverzija (operator cast)

Što će ispisati sljedeći odsječki programa?

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

```
1 double y = 1.8e+20;
2
3 printf("%d\n", (int) y/2);
```

## Konverzija double u int

Što će ispisati sljedeći odsječci programa?

```
1 double x = 5.1;
2
3 printf("%d\n", (int) (1000*x));
```

```
1 double x = 64.1;
2
3 printf("%d\n", (int) (1000*x));
```

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

Što će ispisati sljedeći odsječci programa?

```
1 int a = 1;
2 int *b;
3
4 b = &a;
5 *b = 8;
6 printf("%d□%d\n", a, *b);
```

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      unsigned int m, nzm(unsigned int, unsigned int);
6
7      m = nzm(12, 48);
8      printf("mjera□=□%u\n", m);
9
10     return 0;
11 }
```



```
1 unsigned int nzm(unsigned int a, unsigned int b)
2 {
3
4     while (a != b)
5         if(a > b)
6             a -= b;
7         else
8             b -= a;
9     return a;
10 }
```

## Funkcije - prijenos adrese argumenta

```
1 #include <stdio.h>
2
3 void kvadrat(int *x, int *y)
4 {
5     *y = *x**x; /* = (*x) * (*x). */
6     printf("Unutar funkcije: x=%d, y=%d.\n",
7           *x, *y);
8     return; }
```

```
1 int main(void){
2     int x = 3, y = 5;
3
4     printf("Prije: x=%d, y=%d.\n", x, y);
5     kvadrat(&x, &y);
6     printf("Nakon: x=%d, y=%d.\n", x, y);
7     return 0; }
```

```
1 #include <stdio.h>
2
3 int main(void) {
4     int polje[] = {-6, 2, 1, 7, 3, 8, 0, 5, 9, 4};
5     int indeks;
6     indeks = 0;
7     while (polje[indeks] != 0)
8         ++indeks;
9     printf("Broj el. prije nule: %d\n", indeks);
10
11     return 0;}
```

```
1 int a[10], *pa;
2 pa = a;
3 *(a+1) = 10; /*a[1] = 10;*/
4 pa = pa+2; /*&a[2]*/
5 pa++; /*&a[3]*/
6 *(pa + 3) = 20; /* a[6] = 20;*/
```

## Pokazivači i polja

```
1 #include <stdio.h>
2
3 int main(void) {
4     int polje[] = {-6, 2, 1, 7, 3, 8, 0, 5, 9, 4};
5     int *pokazivac;
6     pokazivac = polje;
7     while ((*pokazivac) != 0)
8         ++pokazivac;
9     printf("Broj el. prije nule: %d\n",
10           pokazivac - polje);
11
12     return 0;}
```

```
1 #include <stdio.h>
2 #define MAX 10
3
4 int main(void) {
5     int polje[MAX];
6     int i, *pokazivac;
7     pokazivac = polje;
8     for (i=0; i < MAX; ++i)
9         polje[i] = i;
10    printf("%d\n", *pokazivac);
11
12    return 0;}
```

**Primjer:** napišite funkcije unos i ispis te glavni program koji upisuje i ispisuje polje s maksimalno 50 elemenata.

```
1 #include <stdio.h>
2 #define MAX 50
3
4 void unos(int a[], int n){
5     int i;
6     for(i = 0; i < n; ++i)
7         scanf("%d", &a[i]);}
8
9 void ispis(int *a, int n){
10    int i;
11    for (i = 0; i < n; ++i)
12        printf("%d\n", *a++); }
```

## Polje kao argument funkcije

```
1 int main(void) {
2     int n, polje[MAX];
3     /*Koliko ce se rezervirati bajtova?*/
4     scanf("%d", &n);
5
6     unos(polje, n);
7     ispis(polj, n);
8
9     return 0;
10 }
```



# Rekurzivne funkcije

```
1 #include <stdio.h>
2
3 int f(int n){
4     if( n == 0)
5         return 2;
6     else return f(--n);}
7
8 int main(void){
9     printf("%d\n", f(4));
10    return 0;
11 }
```

Što se dogodi ukoliko unutar funkcije imamo poziv  $f(n--)$ ?

# Fibonaccijski brojevi

Fibonaccijski brojevi su definirani rekurzijom:  $F_n = F_{n-1} + F_{n-2}$ ,  
 $n \geq 2$ , uz  $F_0 = 0$ ,  $F_1 = 1$ .

Rekurzivna funkcija po definiciji

```
1 long int fib(int n)
2 {
3     if( n == 0) return 0;
4     if( n == 1) return 1;
5
6     return fib(n-1) + fib(n-2);
7 }
```

## Broj poziva funkcije fib

```
1 long int fib(int n)
2 {
3     ++broj_poziva; /*Globalni brojac*/
4     if( n == 0) return 0;
5     if( n == 1) return 1;
6
7     return fib(n-1) + fib(n-2);
8 }
```

Dokažite da je broj rekurzivnih poziva funkcije `fib` za računanje  $F_n$ , uz  $n \geq 2$  jednaka  $(F_1 + F_2 + \dots + F_n) + F_{n-1}$ .

**Uputa:** razmislite koliko puta se događa poziv  $fib(k)$  za  $k = 1, \dots, n$ , a koliko puta  $fib(0)$ .

## Fibonaccijski brojevi - iterativna verzija

```
1 long int fibonacci(int n){
2     long int f_n, f_p, f_pp;
3     int i;
4
5     if( n == 0) return 0;
6     if (n == 1) return 1;
7
8     f_p = 0; /*Prošli F[0]*/
9     f_n = 1; /*Ovaj F[1]*/
10
11    for (i = 2; i<=n; ++i){
12        f_pp = f_p; /*F[i-2]*/
13        f_p = f_n /*F[i-1]*/;
14        f_n = f_p + f_pp /*F[i]*/; }
15
16    return f_n; }
```

Postoji algoritam za računanje  $F_n$  složenosti  $\mathcal{O}(\log n)$ . Taj algoritam ima smisla koristiti za veliki  $n$ . To zahtjeva korištenje biblioteka za prikaz velikih brojeva.

Najveći prikaziv Fibonaccijev broj:

- `int` (32 – bita) -  $F_{46} = 1836311903$ .
- `unsigned int` (long) -  $F_{47} = 2971215073$ .
- `long int` (64 – bita) -  $F_{92} = 7540113804746346429$ .
- `long unsigned` -  $F_{93} = 12200160415121876738$ .