Programiranje 2

Predavanje 01 - ponavljanje

Matej Mihelčić

Prirodoslovno-matematički fakultet
Matematički odsjek

06. ožujka 2025.

Ponavljanje - izrazi

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);
```

Ponavljanje - izrazi

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?

Ponavljanje - izrazi

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 odsječci programa?

```c
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');
```

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

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

Konverzija double u int

Što će ispisati sljedeći odsječci programa?

```c
double x = 5.1;

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

```c
double x = 64.1;

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

```c
int c = 1, i;

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

Pokazivači

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);
```

Funkcije

```c
#include <stdio.h>

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

  m = nzm(12, 48);
  printf("mjera = %u\n", m);

  return 0;
}
```

Funkcije

```
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

```c
#include <stdio.h>

void kvadrat(int *x, int *y)
{
    *y = *x**x; /* = (*x) * (*x). */
    printf("Unutar funkcije: x = %d, y = %d.\n",
            *x, *y);
 return; }
```

```c
int main(void){
     int x = 3, y = 5;

     printf("Prije: x = %d, y = %d.\n", x, y);
     kvadrat(&x, &y);
     printf("Nakon: x = %d, y = %d.\n", x,y);
     return 0; }
```

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 indeks;
6    indeks = 0;
7    while (polje[indeks] != 0)
8        ++indeks;
9    printf("Broj el. prije nule: %d\n", indeks);
10
11   return 0;}
```

Pokazivači i polja

```
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;}
```

Pokazivači i polja

```
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;}
```

Polje kao argument funkcije

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

```c
#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;
}
```

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

Fibonaccijevi brojevi

Fibonaccijevi 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  }
```

Zadatak

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

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

Fibonaccijevi 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; /*Prosli 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;    }
```

Fibonaccijevi brojevi

Postoji algoritam za računanje F_n složenosti $\mathcal{O}(logn)$. 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` - $F_{93} = 12200160415121876738$.