STRUKTURE PODATAKA I ALGORITMI 1
VEŽBE 4

Nikola Bačanin
Nikola Rnjak

PRIMER 1

- Napisati rekurzivnu i iterativnu funkciju koja stepenuje realan broj na celobrojni izlozilac.

```
#include <stdio.h>
float power(float f, int k) {
   if (k == 0)  return 1;
   else  return f*power(f, k-1);
}
```

```
#include <stdio.h>
float power(float f, int k) {
   return k==0 ? 1 : f*power(f, k-1);
}
```

```
float power_iterative(float x, int k) {
   int i;
   float s = 1;
   for (i = 0; i<k; i++) s*=x;
   return s;
}

main() {
   printf("%f", power(2.5, 8));
}
```

PRIMER 2

- Napisati rekurzivnu i iterativnu funkciju koja racuna faktorijel celog broja.

```
#include <stdio.h>
long factorial(int n) {
   if (n == 1)  return 1;
   else
      return n*factorial(n-1);
}
```

```
#include <stdio.h>
long factorial(int n) {
   return n == 1 ? 1 : n*factorial(n-1);
}
```

```
long factorial_iterative(int n) {
   long f = 1;
   int i;
   for (i = 1; i<=n; i++)
      f *= i;
   return f;
}

main() {
   printf("5! = %d\n", factorial(5));
}
```

PRIMER 3

- Napisati rekurzivnu i iterativnu funkciju koja racuna Fibonacijeve brojeve: f(0) = 1, f(1) = 1, f(n) = f(n-1) + f(n-2)

```
#include <stdio.h>
#include <stdlib.h>
int Fib(int n) {
   printf("Racunam Fib(%d)\n",n);
   return (n == 0 || n == 1) ? 1 : Fib(n-1) + Fib(n-2);
}
int Fib_array(int n) {
   int *fib;
   int i, result;
   if ((fib = malloc(n*sizeof(int))) == NULL)   return -1;
   fib[0] = 1;
   fib[1] = 1;
   for (i = 2; i<=n; i++)   fib[i] = fib[i-2] + fib[i-1];
   result = fib[n];
   free(fib);
   return result;
}
```

Primer 3

```
int Fib_iterative(int n) {
    int pp = 1, p = 1;
    int i;
    for (i = 0; i <= n-2; i++)  {
        int tmp = pp;
        pp = p;
        p = p + tmp;
    }
    return p;
}

main() {
    printf("Fib(5) = %d\n", Fib(5));
    printf("Fib(5) = %d\n", Fib_array(5));
    printf("Fib(5) = %d\n", Fib_iterative(5));

}
```

PRIMER 4

- Napisati rekurzivnu i iterativnu funkciju sabira n brojeva.

```
#include <stdio.h>
int array_sum(int a[], int n) {
   return n == 0 ? 0 : array_sum(a, n-1)+a[n-1];
}
```

```
#include <stdio.h>
int array_sum_iterative(int a[], int n) {
   int sum = 0;
   int i;
   for (i = 0; i<n;  i++) sum += a[i];
   return sum;
}
```

```
main() {
   int a[] = {1, 2, 3, 4, 5, 6, 7};
   printf("sum = %d\n",  array_sum(a, sizeof(a)/sizeof(int)));
   printf("sum = %d\n",  array_sum_iterative(a,
sizeof(a)/sizeof(int)));
}
```

Primer 5

```
int max_rek2(int a[], int n) {
    if (n == 0) return a[n];
    else {
        int max_pocetka = max_rek2(a, n-1);
        return a[n] > max_pocetka ? a[n] : max_pocetka;
    }
}

main() {
    int a[] = {2, 8, 3, 7, 9, 6, 4, 5, 1, 2};
    int n = sizeof(a)/sizeof(int);
    printf("%d\n", max_rek1(a, n, 0));
}
```