

## Tema 8

# Rekurzivne funkcije i Stek

# Rekurzivne funkcije

- Rekurzija: funkcija poziva samu sebe
  - Svaka rekurzivna funkcija mora da ima uslov za izlaz iz rekurzije!
  - Pozitivno:
    - razumljivije
    - ponekad i jedino moguće (Akermanova funkcija)
  - Mana:
    - opterećuje stek
    - brzina
- $$A(m, n) = \begin{cases} n + 1 & \text{if } m = 0 \\ A(m - 1, 1) & \text{if } m > 0 \text{ and } n = 0 \\ A(m - 1, A(m, n - 1)) & \text{if } m > 0 \text{ and } n > 0. \end{cases}$$

# Rekurzivne funkcije

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

```
int fakt(int n)
{
    if (n < 2)
        return 1;

    return n * fakt(n-1);
}
```

## Zadatak 1

Napisati rekurzivnu funkciju za izračunavanje n-tog člana Fibonačijevog niza:

$$f_1 = 1, f_2 = 1, f_i = f_{i-1} + f_{i-2}, i = 3, 4, \dots$$

Prirodan broj  $n$  se unosi sa tastature.

## Zadatak 2

Napisati rekurzivnu funkciju kojom se izračunava suma cifara dekadnog broja  $n$ .

## Zadatak 3

Napisati program u kome se korišćenjem rekurzivne funkcije izračunava najveći zajednički delilac prirodnih brojeva  $x$  i  $y$ . Najveći zajednički delilac se rekurzivno definiše sledećom formulom:

$$\text{nzd}(x, y) = \begin{cases} y, & x = 0 \\ \text{nzd}(y \% x, x), & x \neq 0 \end{cases}$$

# Stek

- LIFO struktura (last in – first out, poslednji element koji je dodat u stek prvi se briše)
  - Ovu strukturu možemo napraviti pomoću liste (primer u nastavku) ili statičkog niza
  - Operacije push (dodavanje na vrh steka), pop (brisanje sa vrha), top(čitanje sa vrha)

# Stek – Primer 1

- Kreirati stek celobrojnih vrednosti
  - (U programu ćemo imati pokazivač 'vrh' koji pokazuje na vrh steka, a vrh steka će nam biti prvi element u spregnutoj listi)



# Primer 1 – definicija tipova

```
1
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  typedef int TIP;
6  typedef struct cvor_st
7  {
8      TIP inf;
9      struct cvor_st *sledeci;
10 } SCVOR;
11
```

## Primer 1 – push funkcija

```
31 void push(TIP inf, SCVOR **vrh)
32 {
33     SCVOR *novi;
34     novi = (SCVOR *)malloc(sizeof(SCVOR));
35     novi->inf = inf;
36     novi->sledeci = *vrh;
37     *vrh = novi;
38 }
```

# Primer 1 – pop funkcija

```
40 TIP pop(SCVOR **vrh)
41 {
42     SCVOR *tmp;
43     TIP pod;
44
45     if (*vrh == NULL)
46         return -1;
47     /* kupimo informaciju sa vrha steka */
48     pod = (*vrh)->inf;
49     /* zapamtimo element sa vrha steka da bismo
50        ga obrisali nakon prevezivanja */
51     tmp = *vrh;
52     /* prevezemo (preskocimo) element sa vrha */
53     *vrh = tmp->sledeci;
54     /* obrisemo element sa vrha */
55     free(tmp);
56     return pod;
57 }
```

## Primer 1 – top funkcija


```
59  TIP top(SCVOR *vrh)
60  {
61      if (vrh == NULL)
62          return -1;
63      return vrh->inf;
64  }
```

# Primer 1 – brisanje steka (koristeći rekurziju)

```
21 void obrisi_stek(SCVOR *vrh)
22 {
23     if (vrh != NULL)
24     {
25         printf("brisemo element: %i\n", vrh->inf);
26         obrisi_stek(vrh->sledeci);
27         free(vrh);
28     }
29 }
```

## Primer 1 – ispis steka (koristeći rekurziju)

```
11
12 void ispisi_stek(SCVOR *vrh)
13 {
14     if (vrh != NULL)
15     {
16         printf("element: %i\n", vrh->inf);
17         ispisi_stek(vrh->sledeci);
18     }
19 }
20
```



## Zadatak 1

- Napisati funkciju, koja izračunava vrednost izraza, čiji su članovi jednocifreni brojevi, zdatog u ulaznom tekstualnom fajlu u obrnutoj poljskoj notaciji. Na primer, vrednost izraza  $395+/472-*+$  je jednaka vrednosti:  
 $3/(9+5)+4*(7-2)$
- Funkciju realizovati korišćenjem steka