



PROGRAMIRANJE
Predavanje 08 – Rekurzivne funkcije


Ishod učenja 3

1

Rekurzija

- Rekurzija je pristup rješavanju problema u kojemu funkcija poziva samu sebe s različitim vrijednostima parametara
- Mora postojati jednostavan uvjet zaustavljanja
- Svaki poziv funkcije rješava manji dio problema
- Svaki poziv funkcije je neovisan o ostalim pozivima iste funkcije
 - Ima vlastiti *stackframe*
 - Dva poziva iste funkcije ne mogu pristupiti *stackframovima* jedan drugog

Strana • 2



2

Osnovni pristup rekurziji

- Implementacije rekurzije mogu biti složene
- Osnovni pristup sadrži sljedeće elemente:
 - Inicijalni poziv funkcije (radimo ga iz funkcije `main()`)
 - Provjera uvjeta zaustavljanja
 - Ako je zadovoljen, kraj (i opcionalno vratimo vrijednost)
 - Naredbe specifične za problem koji rješavamo
 - Rekurzivni poziv funkcije
 - Barem jedan parametar mora imati drukčiju vrijednost

Strana * 3



3

Jednostavan void primjer (crtamo stackframe)

- Prikažimo znakove stringa pomoću rekurzije (bez korištenja petlje for)

```
void display(string name, int i, int n) {
    if (i == n) { ← Provjera uvjeta zaustavljanja
        return;
    }
    cout << name[i] << endl; ← Rješavanje problema
    display(name, i + 1, n); ← Rekurzivni poziv funkcije
}

int main() {
    string name = "Mirko";
    display(name, 0, name.size()); ← Inicijalni poziv funkcije
    return 0;
}
```

Strana * 4



4

Jednostavan ne-void primjer

- Izračunajmo n -tu potenciju broja 10

```
int power_of_10(int n) {
    if (n == 0) {
        return 1;
    }
    return power_of_10(n - 1) * 10;
}

int main() {
    cout << power_of_10(0) << endl;
    cout << power_of_10(1) << endl;
    cout << power_of_10(3) << endl;
    return 0;
}
```

Strana * 5



5

Kuharica

- Kuharica za jednostavne rekurzivne probleme:
 1. Prepoznati rekurzivnu formulu, tj. Kako razbiti originalni problem u manje probleme, sve do uvjeta zaustavljanja
 - U većini naših zadataka ćemo koristiti rekurziju umjesto petlje for
 2. Definirati što rekurzivna funkcije prima i vraća
 3. Implementirati uvjet zaustavljanja
 4. Napisati kod za rješavanje problema i rekurzivni poziv funkcije
 5. Napisati inicijalni poziv funkcije iz funkcije `main()`

Strana * 6



6

Primjeri

1. Napišite program koji rekurzivno računa faktorijele.

$$n! = n * (n - 1)!$$

2. Napišite program koji rekurzivno izračunava n-ti Fibonaccijevi broj.

$$F(0) = 0$$

$$F(1) = 1$$

$$F(n) = F(n - 1) + F(n - 2), \text{ za } n \geq 2$$

3. Napišite program koji od korisnika učitava dva prirodna broja a i b. Kroz rekurzivnu funkciju ispišite brojeve od 1 do a u koracima od b. Pretpostavka: vrijednost a je veća od vrijednosti b.

Strana * 7



7

Primjeri

4. Napišite program koji od korisnika učitava tri prirodna broja a, b i c. Kroz rekurzivnu funkciju ispišite brojeve od a do b u koracima od c. Pretpostavka: vrijednost a je manja od vrijednosti b.
5. Napišite program koji od korisnika učitava dva prirodna broja a i b. Kroz rekurzivnu funkciju ispišite sve proste brojeve između a i b. Kôd za provjeru je li broj prost izdvojite u posebnu funkciju.
6. Napišite program koji od korisnika učitava string i znak i uz pomoć rekurzivne funkcije ispisuje koliko se puta taj znak nalazi u stringu.

Strana * 8



8