

# Algoritmi i strukture podataka

*ak. god. 2010./11.*

## Zadaci za 3. laboratorijsku vježbu

Za vrijeme laboratorijskih vježbi dopušteno je korištenje računala **isključivo** u svrhu izrade labosa. Svaka zloupotreba (surfanje, *chat* i slično) bit će **sankcionirana udaljavanjem s vježbi** i adekvatnim evidentiranjem aktivnosti u nastavi.

Posebno, ne tražite od demonstratora da rješavaju zadatke kakvi se pojavljuju na blicevima: ako vam nešto nije jasno, **isprobajte** zadatke na računalu i sami dođite do rješenja - demonstratori će vam rado pomoći, ali neće rješavati zadatke umjesto vas!

Na laboratorijske vježbe dođite **pripremljeni**. **Proučite dosad ispredavano gradivo i primjere sa slajdova**. Programski kodovi nalaze se u repozitoriju predmeta i **možete ih slobodno koristiti** prilikom rješavanja zadataka za laboratorijsku vježbu.

Vježbe sadrže **tri** zadatka: jedan obavezan i dva dodatna. Vezani su uz binarna stabla.

**Zadaci sadrže i općeniti dio - realizaciju glavnog programa - koji također treba napraviti.**

Dodatne zadatke riješite na vježbama ako vam je obavezni zadatak u potpunosti jasan i uspješno ste ga riješili. U suprotnom, **riješite ih kod kuće** za vježbu nakon što savladate materiju prvog zadatka.

Demonstratori će evidentirati studente koji na predavanje dođu **nepripremljeni** i ne pokažu **osnovno znanje** o problemu koji je u zadatku postavljen.

Laboratorijske vježbe traju 2 školska sata. Počinju na puni sat uz akademsku četvrt (10:15, 12:15, 14:15...).

Laboratorijske vježbe **nisu obavezne**. No, studenti koji su se na njih prijavili putem Ferka, **moraju ih odraditi u rezerviranom terminu**, u suprotnom će izgubiti bodove iz aktivnosti u nastavi.

## Na vježbama riješite barem prvi zadatak!

### 1. zadatak

---

Zadana je struktura:

```
typedef struct cvor {
    int broj;
    struct cvor *lijevo;
    struct cvor *desno;
} cvor;
```

Napišite funkciju "dodaj" koja će dodati cijeli broj u binarno stablo. Njen prototip neka bude:

```
cvor* dodaj(cvor* korijen, int broj);
```

Prvi je parametar pokazivač na korijen stabla, a drugi je broj koji se ubacuje u stablo. Povratna vrijednost funkcije je korijen stabla. **Ako je broj koji se treba ubaciti već u stablu, on se ne smije ponovo ubaciti.**

Napišite funkciju "trazi" koja će provjeriti postoji li cijeli broj u binarnom stablu. Njen prototip neka bude:

```
int trazi(cvor* korijen, int broj);
```

Prvi parametar je pokazivač na korijen stabla, a drugi je broj za koji se provjerava nalazi li se u stablu. Ako se taj broj doista pojavljuje u stablu, funkcija treba vratiti 1, a inače treba vratiti 0.

### 2. zadatak

---

Napišite funkciju "postoji\_zbroj" koja će provjeriti postoji li put od korijena do nekog lista stabla za koji je zbroj elemenata na putu jednak traženom broju. Neka njen prototip bude:

```
int postoji_zbroj(cvor* korijen, int zbroj);
```

Prvi je parametar pokazivač na korijen stabla, a drugi je traženi zbroj. Ako takav zbroj postoji, funkcija treba vratiti 1, a inače treba vratiti 0. Za prazno stablo, ova funkcija mora vratiti 0.

### 3. zadatak

---

Napišite funkciju "obrisi" koja će primiti pokazivač na korijen stabla te će obrisati sve elemente. Neka njen prototip bude:

```
void obrisi (cvor* korijen);
```

Parametar je pokazivač na korijen stabla.

## Općenito

---

U glavnom programu testirajte napisane funkcije. Preporuka je sljedeća: prvo napravite prazno stablo, zatim s tipkovnice unesite broj operacija te potom čitajte operacije s tipkovnice u obliku:

- "dodaj X" - pozvati funkciju **dodaj** s parametrom X,
- "trazi X" - ispisati povratnu vrijednost funkcije **trazi** kada se pozove s parametrom X,
- "zbroj X" - ispisati povratnu vrijednost funkcije **zbroj** kada se pozove s parametrom X.

Ako funkciju **zbroj** niste implementirali, ispišite bilo koji broj.

Na kraju obrišite stablo.

Primjer izvođenja:

|     | ulaz    | izlaz | komentar ispisa  |
|-----|---------|-------|--|
|     | 10      |       | broj operacija, nema ispisa  |
| #1  | dodaj 5 |       | dodaje se 5 u korijen, nema ispisa   |
| #2  | dodaj 6 |       | dodaje se 6 kao desno dijete od 5, nema ispisa   |
| #3  | trazi 3 | 0     | u trenutku kada se prvi put provjerava postoji li element 3 u stablu, odgovor je negativan pa treba ispisati 0 |
| #4  | dodaj 3 |       | dodaje se 3 kao lijevo dijete od 5, nema ispisa  |
| #5  | trazi 3 | 1     | u prethodnom koraku broj 3 je ubačen pa ćemo ga pronaći pri ovom pozivu  |
| #6  | zbroj 8 | 1     | put 5-3 ima traženi zbroj 8 te stoga treba ispisati 1  |
| #7  | dodaj 4 |       | dodaje se 4 kao desno dijete od 3, nema ispisa   |
| #8  | zbroj 8 | 0     | u prethodnom koraku dodali smo 4 pa 3 više nije list; kod sljedećeg traženja zbroja 8, njega više nema         |
| #9  | dodaj 2 |       | dodajemo 2 kao lijevo dijete od 3  |
| #10 | trazi 5 | 1     | 5 je korijen stabla pa je jasno da se on nalazi u stablu   |