

# Algoritmi i strukture podataka

## 2. međuispit

22. svibnja 2009.

Napomena za sve zadatke:

- Nije dopušteno korištenje naredbe **goto** te statičkih i globalnih varijabli.

### Zadatak 1. (5 bodova)

Neka je zadan tip podatka **Stog** za koji su definirane funkcije za inicijalizaciju stoga, dodavanje elementa na stog te brisanje elementa sa stoga. Elementi stoga su podaci **tipa Student** koji sadrže prezime studenta (40 znakova), ime studenta (30 znakova) te broj položenih ispita. Prototipovi navedenih funkcija su:

```
void init_stog(Stog *stog);
int dodaj(Student element, Stog *stog);
int skini(Student *element, Stog *stog);
```

Funkcije `dodaj` i `skini` vraćaju 1 ako je operacija dodavanja ili skidanja uspjela, a 0 inače.

a) (4 boda) Napišite **rekurzivnu** funkciju koja će sa stoga maknuti sve studente koji nisu položili niti jedan ispit. Funkcija vraća broj studenata maknutih sa stoga. Prototip funkcije treba biti:

```
int MakniStudente (Stog *stog);
```

**Napomena:**

**Nerekurzivno rješenje neće se priznavati.**

b) (1 bod) Napišite **definicije** odgovarajućih struktura (tipovi podataka **Student**, **Stog** i **atom**) za stog realiziran vezanom listom.

### Zadatak 2. (5 bodova)

Zadan je niz brojeva: 6, 4, 10, 5, 8, 2, 1, 9, 3, 7.

Ilustrirajte uzlazno sortiranje zadanog niza brojeva (ispišite niz nakon svake zamjene dvaju elemenata):

- (3 boda) Algoritmom **quicksort**. Stožer odaberite metodom aproksimacije medijana temeljem početnog, krajnjeg i srednjeg člana polja.
- (2 boda) Algoritmom **selection sort**.

### Zadatak 3. (5 bodova)

Neka je zadan tip podatka **Red** za koji su definirane funkcije za inicijalizaciju reda, dodavanje elementa u red te za brisanje elementa iz reda. Prototipovi navedenih funkcija su:

```
void init_red(Red *red);
int dodaj(int element, Red *red);
int skini(int *element, Red *red);
```

Funkcije `dodaj` i `skini` vraćaju 1 ako je operacija dodavanja ili skidanja uspjela, a 0 inače.

Napisati funkciju čiji je prototip

```
void izbaci(Red *red);
```

koja će iz reda izbaciti sve one elemente koji su manji ili jednaki od svog sljedbenika. Zadnji element se ne izbacuje. Primjerice, ako je red sadržavao sljedeće elemente: 6, 7, 4, 9, 2, 5, 3, 3, 2, 5, 4, 2 nakon poziva funkcije red će imati elemente 7, 9, 5, 3, 5, 4, 2.

**Napomena: možete pretpostaviti da ulazni red neće biti prazan, ali npr. može imati samo jedan element.**

### Zadatak 4. (5 bodova)

Neka je zadan tip podatka **Stog** za koji su definirane funkcije za inicijalizaciju stoga, dodavanje elementa na stog te za brisanje elementa sa stoga. Prototipovi navedenih funkcija su:

```
void init_stog(Stog *stog);
int dodaj (int element, Stog *stog);
int skini (int *element, Stog *stog);
```

Neka je zadan tip podatka **Red** za koji su definirane funkcije za inicijalizaciju reda, dodavanje elementa u red te za brisanje elementa iz reda. Prototipovi navedenih funkcija su:

```
void init_red(Red *red);
int dodaj_u_red (int element, Red *red);
int skini_iz_reda (int *element, Red *red);
```

Funkcije dodavanja i skidanja vraćaju 1 ako je operacija dodavanja odnosno skidanja uspjela, a 0 inače. Neka je zadana funkcija

```
int sadrzi(Stog *stog, int element);
```

koja vraća 1 ako stog sadrži navedeni element, a 0 inače. Nakon poziva funkcije stog ostaje nepromijenjen.

Napisati funkciju **izbaci\_duplikate** koja će izbaciti sva naknadna pojavljivanja pojedinog elementa iz ulaznog reda. Primjerice, ako je red sadržavao elemente 4, 5, 5, 2, 4, 5, 6, 1, 6 nakon poziva funkcije red će sadržavati 4, 5, 2, 6, 1. (Uputa: koristiti funkciju **sadrzi**.) Prototip tražene funkcije je

```
void izbaci_duplikate(Red *red);
```

## Rješenja

### 1. zadatak

```
a) int fun(Stog * stog){  
  
    int brojac=0;  
    student element;  
    if (skini(&element, stog)){  
        brojac=fun(stog);  
        if (element.brojIspita>0) {  
            dodaj(element, stog);  
        }  
        else  
            brojac++;  
  
        return brojac;  
    }  
    else  
        return 0;  
}
```

b)

```
struct student{  
    char prezime[40];  
    char ime[30];  
    short brojIspita;  
};  
  
typedef struct student Student;  
struct at {  
    Student element;  
    struct at *sljed;  
};  
typedef struct at atom;  
  
typedef struct{  
    atom *vrh;  
} Stog;
```

## 2. zadatak

### a) quicksort

**1** - kandidat za stožera

**1** - stožer

1 - elementi koji se zamjenjuju

<b>6</b>	4	10	5	<b>8</b>	2	1	9	3	<b>7</b>
<b>6</b>	4	10	5	<b>7</b>	2	1	9	3	<b>8</b>
6	4	<u>10</u>	5	3	2	<u>1</u>	9	<b>7</b>	8
6	4	1	5	3	2	10	9	<b>7</b>	8
<b>6</b>	4	<b>1</b>	5	3	<b>2</b>	7	9	10	8
<b>1</b>	4	<b>2</b>	5	3	<b>6</b>	7	9	10	8
1	4	3	5	<b>2</b>	6	7	9	10	8
1	2	<b>3</b>	<b>5</b>	4	<b>6</b>	7	9	10	8
1	2	3	4	<b>5</b>	6	7	9	10	8
1	2	3	4	5	6	7	<b>9</b>	<b>10</b>	<b>8</b>
1	2	3	4	5	6	7	<b>8</b>	<b>9</b>	<b>10</b>

### b) selection sort

**1** - nesortirani dio niza

1 - najmanji element

<b>6</b>	4	10	5	8	2	<u>1</u>	9	3	7
1	<b>4</b>	10	5	8	<u>2</u>	6	9	3	7
1	2	<b>10</b>	5	8	4	6	9	<u>3</u>	7
1	2	3	<b>5</b>	8	<u>4</u>	6	9	10	7
1	2	3	4	<b>8</b>	<u>5</u>	6	9	10	7
1	2	3	4	5	<b>8</b>	<u>6</u>	9	10	7
1	2	3	4	5	6	<b>8</b>	9	10	<u>7</u>
1	2	3	4	5	6	7	<b>9</b>	10	<u>8</u>
1	2	3	4	5	6	7	8	<b>10</b>	<u>9</u>
1	2	3	4	5	6	7	8	9	<b>10</b>

### 3. zadatak

```
void izbaci(Red *red){
    int preth, tren;
    Red pom;

    init_red(&pom);

    skini(&preth, red); /*po pretpostavci zadatka postoji barem jedan*/
    while(skini(&tren, red)){
        if (preth > tren){
            dodaj(preth, &pom);
        }
        preth = tren;
    }
    dodaj(preth, &pom); /*zadnji element u redu*/

    while(skini(&tren, &pom))
        dodaj(tren, red);
}
```

### 4. zadatak

```
void izbaci_duplikate(Red *red){
    Stog stog; Red pom; int el;
    init_red(&pom); init_stog(&stog);
    while(skini_iz_reda(&el, red)){
        if (!sadrzi(&stog, el)){
            dodaj(el, &stog);
            dodaj_u_red(el, &pom);
        }
    }
    /*ispraznimo stog*/
    while(skini(&el, &stog));

    while(skini_iz_reda(&el, &pom))
        dodaj_u_red(el, red);
}
```