

Algoritmi i strukture podataka

1. međuispit

9. travnja 2009.

Napomena za sve zadatke:

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

Zadatak 1. (4 boda)

Jedan zapis datoteke organizirane po načelu raspršenog adresiranja definiran je strukturom:

```
typedef struct{
    int sifra;
    char naziv[50+1];
    double cijena;
} zapis;
```

Zapis je prazan ako je na mjestu šifre vrijednost nula. Parametri za raspršeno adresiranje nalaze se u datoteci *parametri.h* i oni su:

- BLOK..... veličina bloka na disku
- MAXZAP broj zapisa
- C..... broj zapisa u jednom pretincu
- M..... broj pretinaca

Ključ zapisa je šifra artikla, a transformacija ključa u adresu obavlja se zadanom funkcijom:

```
int adresa(int sifra);
```

Napišite funkciju koja će u pretincu x pronaći i vratiti prvi zapis koji je po svojoj šifri trebao završiti u pretincu y (ali je zbog preljeva završio u pretincu x). Funkcija preko svog imena vraća 1 ili 0 ovisno o tome postoji li takav zapis u pretincu x ili ne. Prototip funkcije je:

```
int pronadji(FILE *f, zapis *z, int x, int y);
```

Zadatak 2. (4 boda)

Napišite **rekurzivnu** funkciju za izbacivanje znamenki iz zadanog niza. Prototip funkcije treba biti:

```
int IzbaciZnamenke (char *niz);
```

Funkcija vraća broj izbačenih znamenki.

Primjer: za ulazni niz "Godine 2008. je" funkcija mijenja niz u "Godine . je" i vraća 4 jer su izbačene 4 znamenke.

Napomena:

Nerekurzivno rješenje neće se priznavati.

Zadatak 3. (3 boda)

Odredite apriornu složenost (u ovisnosti od m , n i k) sljedećih programskih odsječaka i detaljno obrazložite svoje odgovore.

a)

```
for (j=0; j<m; j++){
    int i = 64;
    n = k;
    while (n > 0){
        n/=i;
        if (i > 2) i /= 2;
    }
}
```

b)

```
int i = 64;
while (i > 1){
    n /= i;
    i /= 2;
}
```

Zadatak 4. (4 boda)

Napišite funkciju koja će stvoriti **skalarnu matricu reda n** (matrica od $n \times n$ elemenata kojoj su svi elementi osim glavne dijagonale 0, a elementi na glavnoj dijagonali međusobno su jednaki). Funkcija kao rezultat vraća pokazivač na stvorenu matricu. U glavnom programu učitajte dimenziju n i vrijednost elemenata na glavnoj dijagonali $e1$, pozovite funkciju, ispišite matricu i nakon toga oslobodite zauzetu memoriju.

Nacrtajte sistemski stog u trenutku neposredno prije završetka izvođenja funkcije (prije poziva naredbe `return`). **Naznačite nazive varijabli i njihove veličine.**

Prototip funkcije je:

```
short *skalMat(int n, short e1);
```

Napomene: Pretpostavite da je prije poziva funkcije `skalMat` stog prazan. Adrese varijabli odaberite proizvoljno.

Zadatak 1:

```
int pronadji(FILE *f, zapis *z, int x, int y){
    zapis pretinac[C]; int i;

    fseek (f, x*BLOK, SEEK_SET);
    fread (pretinac, sizeof (pretinac), 1, f);
    for (i = 0; i < C; i++) {
        if (pretinac[i].sifra != 0) {
            if (adresa(pretinac[i].sifra) == y){
                *z = pretinac[i];
                return 1;
            }
        }
    }
    return 0;
}
```

Zadatak 2:

- a) $O(m \log(k))$
- b) $O(1)$

Zadatak 3:

```
int IzbaciZnamenke (char* niz){
    if (*niz=='\0') return 0;
    else {
        if (*niz >='0' && *niz<='9'){
            strcpy(niz, niz+1);
            return 1 + IzbaciZnamenke(niz);
        }
        return IzbaciZnamenke(niz+1);
    }
}
```

Zadatak 4:

```
#include <stdio.h>
#include <stdlib.h>

short *skalMat(int n, short el){
    int i, j;
    short *polje;
    polje = (short*) malloc(n * n * sizeof(short));
    for (i=0; i<n; i++)
        for (j=0; j<n; j++)
            if (i==j) polje[i*n+j] = el;
            else polje[i*n+j] = 0;
    return polje;
}

int main(){
    int i, j, n;
    short *mat, el;

    printf("n ->"); scanf("%d",&n);
    printf("el ->"); scanf("%hd",&el);
}
```

```
mat = skalMat(n, el);
for (i=0; i<n*n; i++){
    printf("%hd ", mat[i]);
    if (i % n == n-1) printf("\n");
}

free(mat);
return 0;
}
```

STOG:

```
polje 4
j 4
i 4
pov. adr. 4
el 2
n 4
```