

# 1. međuispit iz predmeta Algoritmi i strukture podataka

21. ožujka 2011.

Odgovore na prva tri pitanja napišite na svojim papirima i predajte ih u košuljici.

Odgovore na ostala pitanja napišite na za to predviđenom mjestu uz zadatke.

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

## Zadatak 1. (4 boda)

Napišite glavni program koji dinamički alocira matricu **mat** i puni je pseudoslučajnim cijelim brojevima te funkciju **izbaci\_redak** koja briše zadani redak matrice.

U glavnom programu učitajte broj redaka **brred**, broj stupaca matrice **brstup** te donju i gornju granicu intervala iz kojeg se generiraju pseudoslučajni cijeli brojevi matrice. Nakon učitavanja navedenih parametara zauzmite potrebnu memoriju te matricu napunite pseudoslučajnim brojevima iz zadanog intervala.

Zatim od korisnika zatražite unos indeksa retka matrice koji treba obrisati (**iret**). Redak obrišite u funkciji **izbaci\_redak** tako da se oslobodi memorija koju on zauzima te da ostatak matrice ostane očuvan. Prototip funkcije je:

```
int *izbaci_redak(int *mat, int brred, int brstup, int iret);
```

Na kraju u glavnom programu ispišite matricu koja nastaje izbacivanjem zadanog retka.

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

int *izbaci_redak(int * m, int brred, int brstup, int iret){
    int i, j;
    for (i=iret+1; i<brred-1; i++)
        for (j=0; j<brstup; j++)
            *(m +(i-1)* brstup +j)= *(m +(i)* brstup +j);
    m=(int *)realloc (m,( brred -1)* brstup *sizeof(int));
    return m;
}

int main(){
    int *mat, i, j, iret;
    int red, stup;
    int DG, GG;

    printf("Unesite broj redaka i broj stupaca te donju i gornju granicu intervala za
slucajne brojeve: ");
    scanf("%d %d %d %d", &red, &stup, &DG, &GG);

    mat=(int *) malloc(red*stup*sizeof(int));

    srand((unsigned)time(NULL));
    for (i=0; i<brred; i++)
        for (j=0; j<brstup; j++)
            m[i*maxstup+j]=rand()%(GG-DG +1) + DG;

    do{
        printf("Unesite indeks retka koji zelite izbaciti: ");
        scanf("%d", &iret);
    }while(iret<0 || iret>red);

    mat=izbaci_redak(mat, red, stup, iret);

    for (i=0; i<red-1; i++){
        for (j=0; j<stup; j++)
            printf("%5d", *(mat+i*stup+j));
```

```

        printf("\n");
    }
    free(mat);
    return 0;
}

```

## Zadatak 2. (4 boda)

Zadana je tablica raspršenog adresiranja s 8472 zapisa pohranjena u datoteci "podaci.dat" koja sadrži zapise s podacima o šifri studenta, prosjeku i broju položenih ispita (sami odaberite odgovarajuće tipove podataka!).

Preljevi su realizirani ciklički, upisom u prvi sljedeći slobodni pretinac. Pretinci su usklađeni s veličinom bloka od 512 okteta. Hash-tablica predimenzionirana je za 30%. Neka je ključ šifra studenta. Transformacija ključa u adresu obavlja se zadanom funkcijom:

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

Napišite pretprocesorske direktive **#define** kojima se određuju parametri raspršenog adresiranja.

Napišite i funkciju **brojPreljeva** koja za zadani redni broj pretinca vraća ukupan broj popunjenih zapisa u zadanom pretincu i broj preljeva upisanih u zadani pretinac.

U glavnom programu koristeći funkciju **brojPreljeva** za svaki pretinac ispišite redni broj pretinca, ukupan broj upisanih zapisa u tom pretincu i broj preljeva u tom pretincu.

```

#define BLOK 512L           // Blok na disku
#define N 8472             // Ocekivani broj zapisa:
#define C ((int) (BLOK / sizeof (struct zapis))) // Broj zapisa u pretincu
#define M ((int) (N / C * 1.3)) // Broj pretinaca, kapacitet 30% veci od minimalnog:

struct zapis{
    int sifra;
    float prosjek;
    int brojpolozenih;
};

void brojPreljeva (FILE *fh, int rbrPret, int *ukbroj, int * brPreljev) {
    struct zapis pretinac [C];
    int j;
    *ukbroj=0;
    *brPreljev=0;
    fseek (fh, rbrPret *BLOK, SEEK_SET);
    fread (pretinac, sizeof (pretinac), 1, fh);
    for (j = 0; j < C; j++) {
        if (pretinac[j].sifra != 0) {
            if (rbrPret != adresa(pretinac[j].sifra)) (*brPreljev)++;
            (*ukbroj)++;
        }
    }
}

int main() {
    FILE *fh;
    int brUk=0,brpreljeva=0
    if ((fh = fopen ("podaci.dat", "rb")) == NULL) exit (1);

    for (j = 0; j < M; j++) {
        brojPreljeva(fh, j, &brUk, &brpreljeva);
        printf("%d. pretinac   Ukupno: %d Broj preljeva: %d",j,brUk, brpreljeva);
    }
    close(fh);
}

```

### Zadatak 3. (3 boda)

Napišite rekurzivnu funkciju prototipa:

```
int prebroji_neparne(int broj);
```

koja će odrediti broj neparnih znamenaka zadanog broja.

Napišite glavni program u kojem se s tipkovnice učitava broj, poziva funkcija **prebroji\_neparne** i ispisuje dobiveni rezultat. Odredite složenost napisane funkcije.

```
#include <stdio.h>

int prebroji_neparne(int broj){

    if(broj==0) return 0;

    if ((broj%2) != 0)
        return 1 + prebroji_neparne(broj/10);
    else
        return prebroji_neparne(broj/10);
}

int main(){

    int broj;
    printf("Broj: ");
    scanf("%d", &broj);

    printf("U broju %d, broj neparnih znamenki je %d.", broj,
prebroji_neparne(broj));
    return 0;
}
```

**Zadatak 4. (2 boda)**

Odredite apriornu i asimptotsku složenost sljedećeg programskog odsječka i obrazložite oba odgovora:

```

int suma = 0, i = 0, j, k, m, n;
...
while(i < n){
    for (j = 0; j < m; j++){
        if (i == j){
            for (k = 0; k < n; k++){
                suma+= i * j * k;
            }
        }
        ++i;
    }
}

```

Rješenje: apriorna složenost:  $O(n*m)$

Asimptotska složenost:  $n*m + \min(n,m)*n$

Obrazloženje: Vanjska petlja vrti do  $n$ , a unutarnja do  $m$  i u svakom koraku imamo provjeravanje je li  $i=j$  pa je to sigurno  $n*m$ . K petlja se desi samo minimalno od  $n$  i  $m$  puta a vrti se do  $n$ , znači da se  $k$  petlja obavi  $\min(n,m)*n$

**Zadatak 5. (2 bod)**

Prikažite sadržaj stoga u trenutku neposredno prije prvog izlaska iz funkcije  $f2$  i naznačite veličine svih varijabli na stogu.

<pre> void f2(double *a, int *b, int n) {     int brojac[10]={0};     a[n]=b[n];     brojac[n%10]++; } void f1(int *polje, int n) {     double *polje2;     int i;     polje2=(double*)malloc(n*sizeof(double));     for(i=0;i&lt;n;i++)         f2(polje2,polje,n-i-1); }  int main() {     int polje[100];     ...     f1(polje, 100); } </pre>	<pre> 40  brojac 4   pov.adr. f2 4   polje2      (ili a) 4   polje       (ili b) 4   n-i-1      (ili n) 4   i 4   polje2 4   pov.adr. f1 4   polje 4   100        (ili n) </pre>
---	--