

# Algoritmi i strukture podataka

## 1. međuispit

27.03.2007.

1. (4 boda) Napisati funkciju čiji je prototip:

```
int *duplikat(int *polje, int br_clanova);
```

koja će stvoriti novo polje koje je duplikat ulaznog polja. Funkcija kao rezultat vraća pokazivač na navedeno polje.

Osim toga, napisati i glavni program u kojem će se definirati polje od 50 elemenata, a zatim napuniti slučajno odabranim cijelim brojevima u intervalu [1, 20], pozvati funkcija, ispisati tako dobiveno polje te osloboditi dinamički stvorena memorija.

2. (4 boda) Jedan zapis datoteke organizirane po načelu raspršenog adresiranja definiran je sljedećom strukturom:

```
typedef struct{
    int sifra;
    char naziv[50+1];
    int kolicina;
    float 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 : očekivani maksimalni broj zapisa
- C : broj zapisa u jednom pretincu
- M : broj pretinaca

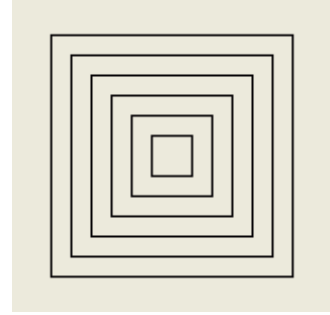
Ključ zapisa je šifra artikla, a transformacija ključa u adresu obavlja se zadanom funkcijom:

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

Napisati funkciju koja će pronaći zapis koji je najviše „udaljen“ od predviđenog pretinca. Ako pojedini zapis nije spremljen kao preljev, njegova je udaljenost 0; inače se udaljenost definira kao broj dodatnih pretinaca koje je potrebno pročitati da bi se zapis pronašao. (*Primjerice, ako je M=15, adresa nekog zapisa 13, a zapis se nalazi u pretincu broj 4, udaljenost je 6*). Funkcija vraća 0 ako nijedan zapis nije zapisan kao preljev; inače vraća najveću udaljenost. Ako postoji više takvih zapisa, vratiti bilo koji. Funkcija treba imati prototip:

```
int max_udaljenost(FILE *f, zapis *z);
```

3. (4 boda) Napisati **rekurzivnu** funkciju koja za zadane  $x$  i  $y$  koordinate lijeve gornje točke kvadrata, duljinu stranice i brojKvadrata crta "koncentrične kvadrate" kao na slici. Točka  $(0, 0)$  koordinatnog sustava se nalazi u gornjem lijevom kutu.



Svaki kvadrat ima središte kao i prethodni, a stranicu duljine 20 piksela veću od prethodnog.

Za crtanje koristite funkciju `DrawRectangle` koja iscrtava pravokutnik s gornjim lijevim kutom u  $(x, y)$  i duljinom stranica `sirina` i `visina`, koja ima prototip:

```
DrawRectangle(int x, int y, int sirina, int visina)
```

Funkciju `DrawRectangle` **ne treba implementirati!**

Na slici je zadan `brojKvadrata = 6`.

Funkcija koju treba napisati ima prototip:

```
void KoncentricniKvadrati(int x, int y, int stranica, int brojKvadrata);
```

4. (3 boda) Odredite apriornu složenost sljedećih programskih odsječaka i detaljno obrazložite svoje odgovore.

```
a) int Funkcija1(int *mat, int n, int maxstu){
    int i, j, suma = 0;
    for (i = 0; i < n; i++){
        for (j = 0; j < n; j++){
            printf("%d\n", mat[i*maxstu + j]);
            if (i + j == 2) suma += mat[i*maxstu + j];
        }
    }
    return suma;
}

b) int Funkcija2(int *mat, int n, int maxstu){
    int i, j;
    for (i = 0; i < n; i++){
        for (j = 0; j < n; j++){
            printf("%d\n", mat[i*maxstu + j]);
            if (i + j == 2) return mat[i*maxstu + j];
        }
    }
}

c) int Funkcija3(int *mat, int n, int maxstu){
    int i, j;
    for (i = 0; i < n; i++){
        for (j = 0; j < n; j++){
            printf("%d\n", mat[i*maxstu + j]);
            if (i == 2) return mat[i*maxstu + j];
        }
    }
}
```

## RJEŠENJA:

### 1.

```
int *duplikat(int *polje, int br_clanova){
    int i, *novo;
    novo = (int*) malloc(br_clanova * sizeof(int));
    for(i=0; i < br_clanova ; i++)
        novo[i] = polje[i];
    return novo;
}
int main(){
    int polje[50], *novo, i, br_clanova = 50;
    srand((unsigned) time(NULL));

    for(i=0; i<br_clanova ; i++){
        polje[i] = 1+rand()%20;
    }

    novo = duplikat(polje, br_clanova);
    for(i=0; i<br_clanova ; i++){
        printf("%d\n", novo[i]);
    }
    free(novo);
    return 0;
}
```

### 2.

```
int max_udaljenost(FILE *f, zapis *z){
    zapis pretinac[C];
    int i, j;
    int udaljenost, max = 0;

    for (i = 0; i < M; i++) {
        fseek (f, i*BLOK, SEEK_SET);
        fread (pretinac, sizeof (pretinac), 1, f);
        for (j = 0; j < C; j++) {
            if (pretinac[j].sifra != 0) {
                /* Ako zapis nije prazan */
                if (adresa(pretinac[j].sifra) != i){
                    udaljenost = i - adresa(pretinac[j].sifra);
                    if (udaljenost < 0)
                        udaljenost += M;
                    if (udaljenost > max){
                        *z = pretinac[j];
                        max = udaljenost;
                    }
                }
            }
        }
    }
    return max;
}
```

### 3.

```
void KoncentricniKvadrati(int x, int y, int stranica, int brojKvadrata)
{
    if (brojKvadrata >= 1)
    {
        DrawRectangle(x, y, stranica, stranica);
    }
}
```

```
        brojKvadrata = brojKvadrata - 1;  
        KoncentricniKvadrati(x-10, y-10, stranica+20, brojKvadrata);  
    }  
}
```

#### 4.

- a) Bez obzira koji je rezultat naredbe  $(i + j == 2)$  program će proći kroz cijelu matricu koja ima  $n^2$  elemenata, pa je zato složenost  $O(n^2)$ .
- b) Uvjet  $(i + j == 2)$  prvi put je ispunjen točno kada je  $i = 0$  i  $j = 2$ . Do tada smo samo 2 puta prošli naredbu `if (i + j == 2) suma += mat[i*maxstu + j];`, a treći put izlazimo iz petlji.  
Složenost je 6 brojimo li naredbu `printf(...)` i `if(...)`, dakle konstantna, odnosno  $O(1)$ .
- c) Uvjet  $(i == 2)$  prvi put je ispunjen točno kada je  $i = 2$  i  $j = 0$ . Do tada smo samo  $2*n$  puta prošli naredbu `if (i + j == 2)`, a onda izlazimo iz petlji.  
Složenost je  $2*n*2$  brojimo li naredbu `printf(...)` i `if(...)`, odnosno  $O(n)$ .