JMBAG **IME I PREZIME**

Algoritmi i strukture podataka – Međuispit

23. travnja 2014.

Nije dopušteno korištenje globalnih i statičkih varijabli te naredbe ***goto****. Neefikasna rješenja mogu donijeti manje bodova. Nerekurzivne funkcije se ne priznaju kao rješenja u zadacima u kojima se traži rekurzivna funkcija. Ispit nosi maksimalno 25 bodova, a praga za prolaz međuispita nema. Ovaj primjerak ispita morate predati s upisanim imenom i prezimenom i JMBAG-om.*

Zadatak 1. (7 bodova)

Svaki zapis datoteke `podaci.dat` organizirane po načelu raspršenog adresiranja sadrži podatke o jednom studentu i njegovoj ocjeni na kolegiju Algoritmi i strukture podataka: *šifru studenta (znakovni niz duljine 10 znakova), ime studenta (duljine do 20 znakova), prezime studenta (duljine do 30 znakova)* te *ocjenu (`int`)*.

Šifra nula (`"0"`) označava prazan zapis. Veličina bloka na disku je 4096 B. Očekuje se najviše 10000 zapisa, a kapacitet tablice treba biti 15% veći. Prilikom upisa primjenjuje se metoda cikličkog preljeva. Ključ zapisa je **`sifra`**, a pretvorba ključa u adresu se obavlja već pripremljenom funkcijom prototipa:

```
int adresa (char *sifra);
```

Napišite program koji će odrediti prosječnu ocjenu svih studenata čiji su zapisi završili u preljevu. Pretpostavite da u datoteci nije bilo brisanja (i iskoristite tu pretpostavku u radu s datotekom). Na zaslon je potrebno ispisati podatke za sve studente za koje zapisi nisu završili u preljevu, a kojima je ocjena veća ili jednaka prosječnoj ocjeni studenata čiji su zapisi završili u preljevu.

Primjer: Neka je prosječna ocjena `3,64`. Student `Ivan Novak` ima ocjenu `4` i šifru `0036123456`, a njegov zapis ne nalazi se u preljevu. Funkcija na zaslon mora ispitati njegove podatke u sljedećem formatu (prva linija označava znakovna mjesta pri ispisu):

```
12345678901234567890123456789012345678901234567890123456789012345678901234567890
0036123456 Novak                          Ivan                4
```

Zadatak 2. (5 bodova)

Kvazi-binomni koeficijenti K(n,m) su definirani sa

$$K(n, m) = K(n-1, m-1) + m \cdot K(n-1, m+1) ;$$
$$K(n, n) = K(n, 0) = 1 .$$

Ako je `n < 0` ili `m > n`, onda je `K(n, m) = 0`.

Napišite rekurzivnu funkciju `kvazi_binomni` koja će za bilo koji `n` i `m` izračunati vrijednost kvazi-binomnog koeficijenta `K(n,m)`.

Zadatak 3. (5 bodova)

Svaki prirodni broj $N \geq 2$ može se napisati u obliku rastava na proste faktore:

$$N=p_1^{n1} \cdot p_2^{n2} \cdots p_k^{nk},$$

gdje su p_i, $i=1,\ldots,k$, različiti prosti brojevi (prosti faktori), a eksponenti $n_1,\ldots,n_k$ su prirodni brojevi.

Napišite rekurzivnu funkciju `najveci_eksponent` koja će za zadani broj `N` odrediti najveći eksponent s kojim se neki prosti broj pojavljuje u rastavu broja `N` na proste faktore (najveći od brojeva n_i). Prototip funkcije je:

```
int najveci_eksponent (int n, int m);
```

pri čemu je `m` trenutačni djelitelj (p_i). Početni poziv funkcije za zadani `n` je oblika `najveci_eksponent(n,2)`.

Primjeri:

Za `N = 6`, funkcija treba vratiti `1` jer $6 = 2^1 \cdot 3^1$.
Za `N = 8`, funkcija treba vratiti `3` jer $8 = 2^3$.
Za `N = 36`, funkcija treba vratiti `2` jer $36 = 2^2 \cdot 3^2$.

Zadatak 4. (4 boda)

Napišite funkciju prototipa

```
int trazi_visestruke (int *polje, int brojElemenata)
```

koja u zadanom polju pozitivnih cijelih brojeva (vrijednost svakog pojedinog broja je manja od 101) provjerava ima li brojeva koji se ponavljaju. Funkcija vraća logičku vrijednost `istina (1)` ako polje ima ponavljajuće brojeve, odnosno logičku vrijednost `laž (0)` u suprotnom. **Dozvoljena složenost tražene funkcije je `O(n)`, gdje je `n = brojElemenata`.**

Napomena: Rješenje složenosti veće od O(n) nositi će značajno manje bodova.

Naputak: Koristite pomoćno polje za utvrđivanje učestalosti brojeva.

Zadatak 5. (4 boda)

Zadano je polje brojeva s elementima: **`91, 7, 8, 20, 5, 18, 4, 9`**. Ilustrirajte uzlazno sortiranje zadanog niza brojeva (ispišite polje nakon svake promjene i potcrtajte sve brojeve relevantne za sljedeći korak) algoritmom ***shellsort*** **uz korake k = 4, 2, 1**

Rješenja

1. (7 bodova)

```c
#include <stdio.h>
#include <string.h>

#define N 10000
#define BLOK 4096
#define C BLOK/sizeof (zapis)
#define M (int)(N * 1.15/C)

typedef struct {
  char sifra[10 + 1];
  char ime[20 + 1];
  char prezime[30 + 1];
  int ocjena;
} zapis;

int adresa(int sifra);

int main() {

  FILE *f = NULL;
  zapis pretinac[C];
  int i, j, sumaOcjena = 0, brStudenata = 0;
  float prosjecnaOcjena = 0;

  f = fopen("podaci.dat", "rb");

  /*Dohvat ocjena iz zapisa koji su zavrsili u preljevu*/
  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(strcmp(pretinac[j].sifra,"0") == 0) {
        /*Prazan zapis - prelazimo na sljedeci pretinac*/
        break;
      }
      if(adresa(pretinac[j].sifra) != i) {
        /*Preljev*/
        brStudenata++;
        sumaOcjena += pretinac[j].ocjena;
      }
    }
  }

  /*Racunanje prosjecne ocjene*/
  prosjecnaOcjena = (float)sumaOcjena / brStudenata;

  /*Na zaslon ispisujemo podatke za studente cija je ocjena >= prosjecnoj ocjeni*/
  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(strcmp(pretinac[j].sifra,"0") == 0) {
        /*Prazan zapis - prelazimo na sljedeci pretinac*/
        break;
      }
      if(adresa(pretinac[j].sifra) == i && pretinac[j].ocjena >= prosjecnaOcjena) {
        /*Zapis koji nije preljev*/
        printf("%-11s%-30s%-20s%-9d\n",
          pretinac[j].sifra,
          pretinac[j].prezime,
          pretinac[j].ime,
          pretinac[j].ocjena
        );
      }
    }
  }

  fclose(f);
  return 0;
}
```

2. (5 bodova)

```c
int kvazi_binomni(int n, int m) {
       if( ( n < 0 ) || ( m > n ) ) return 0;
       if( ( m == 0 ) || ( n == m ) ) return 1;
       return kvazi_binomni( n - 1, m - 1 )
              + m * kvazi_binomni( n - 1, m + 1 );
}
```

3. (5 bodova)

```
int najveci_eksponent(int n, int m){
	int dalje, eksponent=0;
	if (n<m) return 0; /* Osnovni slucaj */
	while(n % m == 0){   /* Koliko puta m dijeli n */
		eksponent++;
		n /= m;
	}
	dalje= najveci_eksponent (n,m+1); /* Rekurzivno napredovanje */

	if (dalje>eksponent) return dalje; /* Veci se vraca pozivajucoj proceduri */
	return eksponent;
}
```

4. (4 boda)

```
int trazi_visestruke(int* polje, int brojElemenata)  {
	int *pomPolje = NULL;
	pomPolje = (int *) malloc(
		100 * sizeof(int));
	for (int i = 0; i < 100; i++) {
		pomPolje[i] = 0;
	}
	for (int i = 0; i < brojElemenata; ++i){
		pomPolje[polje[i] - 1]++;
		if (pomPolje[polje[i] - 1] > 1) {
			return 1;
		}
	}
	free(pomPolje);
	return 0;
}
```

5. (4 boda)

Korak = 4							
91	7	8	20	5	18	4	9
5	7	8	20	91	18	4	9
5	7	8	20	91	18	4	9
5	7	4	20	91	18	8	9
5	7	4	9	91	18	8	20

Korak = 2							
5	7	4	9	91	18	8	20
4	7	5	9	91	18	8	20
4	7	5	9	91	18	8	20
4	7	5	9	91	18	8	20
4	7	5	9	91	18	8	20
4	7	5	9	8	18	91	20
4	7	5	9	8	18	91	20

Korak = 1							
4	7	5	9	8	18	91	20
4	7	5	9	8	18	91	20
4	5	7	9	8	18	91	20
4	5	7	9	8	18	91	20
4	5	7	8	9	18	91	20
4	5	7	8	9	18	91	20
4	5	7	8	9	18	91	20
4	5	7	8	9	18	20	91