JMBAG **IME I PREZIME**

Algoritmi i strukture podataka – Međuispit

21. travnja 2015.

Nije dopušteno korištenje globalnih i statičkih varijabli te naredbe goto. Neučinkovita rješenja mogu donijeti manje bodova. Nerekurzivne funkcije se ne priznaju kao rješenja u zadacima u kojima se traži rekurzivna funkcija (zadaci 3 i 4).
Ispit donosi maksimalno 25 bodova, a praga za prolazak nema. Ovaj primjerak ispita morate predati s upisanim imenom i prezimenom te JMBAG-om.

Zadatak 1. (5 bodova)

Svaki zapis datoteke organizirane po načelu raspršenog adresiranja sadrži podatke o jednom studentu i njegovoj ocjeni na predmetu *Algoritmi i strukture podataka*. Zapis je zadan strukturom:

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

Šifra nula (`0`) označava prazan zapis. Veličina bloka na disku je 4096 B. Očekuje se najviše 1000 zapisa, a kapacitet tablice treba biti 30% veći. Ključ zapisa je **`sifra,`** a prilikom upisa primjenjuje se metoda cikličkog preljeva.

Napišite funkciju koja će sve neprazne zapise iz datoteke kopirati u jednodimenzionalno polje i u pozivajući program vratiti pokazivač na prvi element polja te ukupan broj studenata za koje su iskopirani zapisi. Prototip funkcije je:

```
student *hashUPolje(char *nazivDatoteke, int *brStudenata);
```

Parametar `nazivDatoteke` sadrži naziv datoteke organizirane po načelu raspršenog adresiranja iz koje je potrebno pročitati podatke o studentima.

Zadatak 2. (5 bodova)

Napisati funkciju *split* prototipa:

```
int split(char *recenica, char delimiter, char **rijeci)
```

koja putem argumenata prima znakovni niz (`recenica`) i znak (`delimiter`) prema kojemu treba razdvojiti zadani niz u podnizove, a dobivene podnizove vraća putem trećeg argumenta. Kao povratnu vrijednost funkcija treba vratiti broj pronađenih podnizova.

Argument `rijeci` je polje pokazivača definirano u pozivajučoj proceduri (*main*-u), a može se pretpostaviti da je polje dovoljno veliko za pohranu pokazivača na sve riječi. Unutar funkcije `split` potrebno je alocirati za svaku riječ točno potrebnu memoriju. Npr. sljedeći jednostavan program poziva funkciju `split` i ispisuje 3 retka na standardni izlaz:

```
int main() {
	char *rijeci[MAX_BROJ];
	int i, n;
	n = split( "Ovo je recenica", ' ', rijeci );
	for( i = 0; i < n; i++ ) {
		printf( "%s\n", rijeci[i] );
		free(rijeci[i]);
	}
	return 0;

}
```

Očekivani ispis ovog programa je riječ „`Ovo`“ u prvom retku, zatim „`je`“ u drugom i „`recenica`“ u trećem.

Zadatak 3. (5 bodova)

Napisati program koji pomoću rekurzivne funkcije nalazi, a u glavnom programu ispisuje najmanji zajednički višekratnik i najveći zajednički djelitelj dva prirodna broja. Brojevi se učitavaju putem standardnog ulaza u glavnom programu.

Zadatak 4. (5 bodova)

a) Napisati rekurzivnu funkciju **`okreni_oduzmi`** koja ispisuje prvih n članova cjelobrojnog polja, gdje se svaki član polja ispisuje umanjen za element na mjestu s indeksom 0. Ispis treba biti u obrnutom redoslijedu, tj. prvo treba ispisati element s indeskom n-1. Npr. za polje
`2 6 8 11 -3 100`
uz ispravne argumente (tj. za `n=5`) ispis prvih 5 članova obrnutim redoslijedom umanjenih za iznos prvog (člana s indeksom 0) je:
`-5 9 6 4 0`

b) Nadopuniti funkciju main s ispravnim pozivom funkcije `okreni_oduzmi`:

```
int main() {

	int A[] = { 2, 6, 8, 11, -3, 100 };

	_____________________________________

	return 0;

}
```

Zadatak 5. (5 bodova)

U polje cijelih brojeva pohranjen je sljedeći niz brojeva:

`3, 1, 2, 5, 8, 6, 4, 10, 9, 7`

a) **(2 boda)** Ilustrirati uzlazno sortiranje algoritmom **Bubblesort**.

b) **(3 boda)** Ilustrirati uzlazno sortiranje algoritmom **Quicksort**. Stožer za quicksort odabrati metodom aproksimacije medijana temeljem prvog, središnjeg i zadnjeg člana. Polja s manje ili točno `cutoff = 3` elementa sortirati Insertion sortom.

Za oba algoritma navesti korake u sortiranju kod kojih je došlo do zamjene pojedinih članova polja te označiti zamjene članova polja.

Rješenja

1. (5 bodova)

```
#define N 1000
#define BLOK 4096
#define C (BLOK / sizeof (student))
#define M (int)(N * 1.3 / C)

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

student* hashUPolje(char *nazivHash, int *brStudenata) {
  FILE *fUlaz;
  student p, pretinac[C], *ret;
  int i, j, init = 0;

  *brStudenata = 0;
  fUlaz = fopen(nazivHash, "rb");

  for (i = 0; i < M; i++) {
    /*Procitaj pretinac*/
    fseek(fUlaz, i * BLOK, SEEK_SET);
    fread(pretinac, sizeof(pretinac), 1, fUlaz);
    /*Kopiraj sve popunjene zapise*/
    for (j = 0; j < C; j++) {
      p = pretinac[j];
      if (p.sifra != 0) {
        (*brStudenata)++;
        if (!init) {
          ret = (student*)malloc(*brStudenata * sizeof(student));
          init = 1;
        } else {
          ret = (student*)realloc(ret, *brStudenata * sizeof(student));
        }
        ret[*brStudenata - 1] = p;
      }
    }
  }
  fclose(fUlaz);
  return ret;
}
```

2. (5 bodova)

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


int split(char *recenica, char delimiter, char **rijeci) {
	int i, brojRijeci, zadnjiIndeks, duljina;
	duljina = strlen(recenica);


	for (i = 0, brojRijeci = 0, zadnjiIndeks = 0; i < duljina; i++){
		if (recenica[i] == delimiter ){
			rijeci[ brojRijeci ] = ( char * ) malloc( i - zadnjiIndeks + 1 );
			strncpy(rijeci[brojRijeci], recenica + zadnjiIndeks, i - zadnjiIndeks);
			rijeci[brojRijeci][i - zadnjiIndeks] = '\0';
			++brojRijeci;
			zadnjiIndeks = i + 1;
		}
	}
	rijeci[ brojRijeci ] = ( char * ) malloc( duljina - zadnjiIndeks + 1 );
	strncpy(rijeci[brojRijeci], recenica + zadnjiIndeks, duljina - zadnjiIndeks);
	rijeci[brojRijeci][duljina - zadnjiIndeks] = '\0';

	return brojRijeci + 1;
}
```

3. (5 bodova)

```c
#include <stdio.h>

int nadji_nzd(int, int);

int main() {
	long x, y, nzd, nzv;

	printf("Unesi dva cijela broja\n");
	scanf("%ld%ld", &x, &y);

	nzd = nadji_nzd(x, y);
	nzv = (x*y) / nzd;

	printf("Najveci zajednicki djelitelj %ld i %ld = %ld\n", x, y, nzd);
	printf("Najmanji zajednicki visekratnik %ld i %ld = %ld\n", x, y, nzv);

	return 0;
}

int nadji_nzd(int a, int b) {
	if (b == 0) {
		return a;
	}
	else {
		return nadji_nzd(b, a % b);
	}
}
```

4. (5 bodova)

```
#include <stdio.h>

void okreni_oduzmi( int *a, int m, int n ) {
        if( m < n ) {
                okreni_oduzmi( a, m + 1, n );
        }
        printf( "%d ", a[ m ] - a[ 0 ] );
}


int main() {
        int A[] = { 2, 6, 8, 11, -3, 100 };
        okreni_oduzmi( A, 0, 4 );
        return 0;
}
```

5. (5 bodova)

BubbleSort (2B)

3 1 2 5 8 6 4 10 9 7	
1 3 2 5 8 6 4 10 9 7	
1 2 3 5 8 6 4 10 9 7	
1 2 3 5 6 8 4 10 9 7	
1 2 3 5 6 4 8 10 9 7	
1 2 3 5 6 4 8 9 10 7	
1 2 3 5 6 4 8 9 7 10	
1 2 3 5 4 6 8 9 7 10	
1 2 3 5 4 6 8 7 9 10	
1 2 3 4 5 6 8 7 9 10	
1 2 3 4 5 6 7 8 9 10	

qsort (3B):

3 1 2 5 8 6 4 10 9 7	početno polje, odabir stožera
3 1 2 5 **9** 6 4 10 **7** 8	Nakon sklanjanja stožera (7)
3 1 2 5 **4** 6 **9** 10 7 8	nakon zamjene A[i=4] s A[j=6]
3 1 2 5 4 6 9 10 7 8	Mimoilaženje na i=6, j=5
3 1 2 5 4 6 **7** 10 **9** 8	Nakon obnove stožera
3 1 2 5 4 6 7 10 9 8	Sortiranje lijeve strane
2 1 **4** 5 **3** 6 7 10 9 8	Nakon sklanjanja stožera 3
2 1 4 5 3 6 7 10 9 8	Mimoilaženje na i=2, j=1
2 1 **3** 5 **4** 6 7 10 9 8	Nakon obnove stožera
2 1 3 5 4 6 7 10 9 8	Sortiranje lijeve srane
1 2 3 5 4 6 7 10 9 8	Sortiranje desne strane
1 2 3 5 4 6 7 10 9 8	Sortiranje ranije desne strane
1 2 3 5 4 6 7 **9** 10 8	Insertion sort – 9 ubačen
1 2 3 5 4 6 7 **8** 9 10	Insertion sort – 8 ubačen