Algoritmi i strukture podataka

4. rujna 2012.

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

1. zadatak (18 bodova)

a) U memoriji postoji sortirana tablica ključeva **tablica**. Napisati funkciju koja će pretraživanjem po blokovima pronaći zadani ključ i vratiti pokazivač na taj ključ. Ako zadani ključ nije pronađen, potrebno je vratiti NULL pokazivač. Funkcija treba izračunati i koristiti optimalnu veličinu bloka.
Prototip funkcije je:

```
int *trazi (int *tablica, int velicinaTablice, int *trazeniKljuc);
```

b) Izvesti izraz za složenost algoritma kad je veličina bloka optimalna.

2. zadatak (10 bodova)

Napisati funkciju koja na zaslon ispisuje zadani niz znakova obrnutim redoslijedom.

Prototip funkcije je: **`void pisi (char *s);`**

Nije dozvoljena standardna programska petlja niti korištenje drugih funkcija osim **`printf`**. Funkcija mora raditi i za prazan niz, u kojem slučaju ne ispisuje ništa.

Primjer: **`pisi („ispit“)`** -> tipsi

3. zadatak (10 bodova)

Napisati funkciju koja će na zaslon ispisati vrijednost iz svih čvorova binarnog stabla koji imaju neparni broj potomaka. Dozvoljen je samo jedan prolaz kroz stablo. Čvor stabla zadan je strukturom:

```
typedef struct cvor {
  int vrijednost;
  struct cvor *lijevo;
  struct cvor *desno;
} cvor;
```

Prototip funkcije je: **`int ispisi (cvor *korijen);`**

4. zadatak (14 bodova)

Napisati funkciju koja će iz jednostruko povezane linearne liste izbrisati sve čvorove koji imaju neparnu vrijednost. Prototip funkcije je **`void brisi (atom **glava)`**. Funkcija ne smije koristiti niti jednu pomoćnu varijablu. **Nije potrebno** oslobađati zauzetu memoriju.

Atom liste zadan je strukturom:

```
typedef struct atom {
  int elem;
  struct atom *sljed;
} atom;
```

5. zadatak (18 bodova)

Napisati funkciju **`void merge (char *prva, char *druga, char *treca)`** koja će od podataka iz sortiranih datoteka **`prva`** i **`druga`** stvoriti sortiranu datoteku **`treca`**. Svaki redak datoteka sadrži jedan znakovni niz maksimalne duljine 100 znakova. Ne treba kontrolirati je li otvaranje ili stvaranje datoteka uspjelo.

Rješenja:

1. a) (15 bodova)

Verzija 1)

```
int *trazi(int *tablica, int velicinaTablice, int trazeniKljuc){
          int velicinaBloka = (int)sqrt((float)velicinaTablice);
          int i = 0, j = 0;
          int traziUZadnjem = 1;

          //Pretrazujemo vodece zapise u blokovima
          for(i = 0; i < velicinaTablice; i+= velicinaBloka){
                    if(tablica[i] > trazeniKljuc){
                              //Trazeni kljuc mora biti u prethodnom bloku pa ne
                              //treba pretrazivati zadnji blok
                              traziUZadnjem = 0;
                              //Vodeci zapis u bloku veci je od trazenog
                              //kljuca - pretrazujemo prethodni blok
                              for(j = i - velicinaBloka; j < i; j++){
                                        if(tablica[j] == trazeniKljuc){
                                                  //Trazeni kljuc je pronadjen
                                                  return &tablica[j];
                                        }
                              }
                              //Trazeni kljuc nije pronadjen
                              return NULL;
                    } else if (tablica[i] == trazeniKljuc){
                              //Vodeci zapis je trazeni kljuc
                              return &tablica[i];
                    }
          }

          //Trazeni kljuc nije pronadjen - pretrazit cemo posljednji blok ako
          //je zadnji element u tablici veći ili jednak trazenom kljucu
          if(traziUZadnjem = 1 && tablica[velicinaTablice - 1] >= trazeniKljuc){
                    for(i = (i - velicinaBloka) + 1; i < velicinaTablice; i++){
                              if(tablica[i] == trazeniKljuc){
                                        return &tablica[i];
                              }
                    }
          }
          //Kljuc nije pronadjen
          return NULL;
}
```

Verzija 2)

```
int *trazi2(int *tablica, int velicinaTablice, int trazeniKljuc){
          int velicinaBloka = (int)sqrt((float)velicinaTablice);
          int i = 0, j = 0;
          int traziUZadnjem = 1;

          do{
                    if(tablica[i] < trazeniKljuc){
                              if(i + velicinaBloka >= velicinaTablice){
                                        i++;
                              }else{
                                        i += velicinaBloka;
                              }
                              continue;
                    }else if(tablica[i] == trazeniKljuc){
                              return &tablica[i];
                    }
          }while(i < velicinaTablice);
}
```

1. b) (3 boda)

O(sqrt(n))

2. (10 bodova)

```
void pisi(char *a){
          if(a[0] == '\0'){
                    return;
          }
          pisi(a+1);
          printf("%c", a[0]);
}
```

3. (14 bodova)

```
int ispisi(cvor *korijen){
	int brPotomaka = 0;
	if(korijen == NULL){
		return 0;
	}
	brPotomaka = ispisi(korijen->lijevo) + ispisi(korijen->desno);
	if(brPotomaka % 2){
		printf("Cvor %d ima neparan broj potomaka - %d\n", korijen->vrijednost, brPotomaka);
	}
	return brPotomaka + 1;
}
```

4. (14 bodova)

```
void brisi(atom **glava){
	while((*glava)){
		if((*glava)->elem % 2){
			(*glava) = (*glava)->sljed;
		}else{
			glava = &(*glava)->sljed;
		}
	}
}
```

5. (14 bodova)

```
void merge(char *prva, char *druga, char *treca){
	FILE *prvaD, *drugaD, *trecaD;
	int imaPrva = 1, imaDruga = 1;
	char redakPrva[ROW_MAX_LEN + 1] = {'\0'};
	char redakDruga[ROW_MAX_LEN + 1] = {'\0'};

	prvaD = fopen(prva, "r");
	drugaD = fopen(druga, "r");
	trecaD = fopen(treca, "w");

	//Procitaj redak iz prve
	if(fscanf(prvaD, "%s\n", redakPrva) == EOF){
		//Prva datoteka je prazna
		imaPrva = 0;
	}
	if(fscanf(drugaD, "%s\n", redakDruga) == EOF){
		//Druga datoteka je prazna
		imaDruga = 0;
	}

	do{
		if(imaPrva && imaDruga){
			if(strcmp(redakPrva, redakDruga) < 0){
				fprintf(trecaD, "%s\n", redakPrva);
				if(fscanf(prvaD, "%s\n", redakPrva) == EOF){
					imaPrva = 0;
				}
			}else if(strcmp(redakPrva, redakDruga) > 0){
				fprintf(trecaD, "%s\n", redakDruga);
				if(fscanf(drugaD, "%s\n", redakDruga) == EOF){
					imaDruga = 0;
				}
			}else{
				fprintf(trecaD, "%s\n", redakPrva);
				fprintf(trecaD, "%s\n", redakDruga);
				if(fscanf(prvaD, "%s\n", redakPrva) == EOF){
					imaPrva = 0;
				}
				if(fscanf(drugaD, "%s\n", redakDruga) == EOF){
					imaDruga = 0;
				}
			}
		}else if(imaPrva){
			fprintf(trecaD, "%s\n", redakPrva);
			if(fscanf(prvaD, "%s\n", redakPrva) == EOF){
				imaPrva = 0;
			}
		}else if(imaDruga){
			fprintf(trecaD, "%s\n", redakDruga);
			if(fscanf(drugaD, "%s\n", redakDruga) == EOF){
				imaDruga = 0;
			}
		}else{
			break;
		}
	} while(1);

	fclose(prvaD);
	fclose(drugaD);
	fclose(trecaD);
}
```