Projekt F, prvi dio

Binarna indeksno neslijedna datoteka `studenti.dat` sastoji se od tri logičke cjeline. U prvom dijelu datoteke su podaci o studentima, za svakog studenta je binarno pohranjen zapis definiran struktrom:

```
struct student {
	char ime_prezime[51];
	char mjesto_rodjenja[41];
	char prebivaliste[41];
	char jmbag[11];
	char oib[12];
};
```

Svaki zapis zauzima 156 znakova (bajta), a poredani su slijedno po imenu (i prezimenu). Datoteka ne podržava logičko brisanje (nema praznih zapisa), a za svakog studenta ispravno su pohranjena su sva tri podatka. Kapacitet datoteke je unaprijed definiran i iznosi 10.000 zapisa.

Ovaj dio datoteke zauzima dio datoteke od datotečne adrese 0 do datotečne adrese `BROJ_ZAPISA * sizeof(struct student) - 1` odnosno prvih 1.560.000 bajtova.

U drugom dijelu datoteke pohranjen je indeks prema jmbag-u. Svaki zapis indeksa odnosi se na jedan zapis iz prvog dijela datoteke, a definiran je strukturom:

```
struct idx_jmbag {
	long pozicija;
	char jmbag[11];
};
```

gdje je u `pozicija` pohranjena pozicija zapisa o studentu na kojeg se zapis indeksa odnosi (vrijednost koju vraća funkcija `ftell`, a očito će u ovoj datoteci uvijek biti višekratnik broja `sizeof(struct student)`), a u `jmbag` je pohranjen jmbag iz zapisa studenta na kojeg se odnosi zapis indeksa. Indeksni zapisi pohranjeni su redosljedom rastućeg jmbag-a u leksičkom poretku (vrijedi "0" < "00" < "000" < … < "1" < "10" < "11" <…), tj. prvi pohranjeni indeksni zapis odnosi se na zapis studenta s najmanjim jmbag-om po leksičkom poretku, gdje god da je taj zapis pohranjen unutar prvog dijela datoteke.

Veličina ovog dijela datoteke je `BROJ_ZAPISA * sizeof(struct idx_jmbag)` (odnosno 150.000 bajtova).

U trećem dijelu datoteke pohranjen je indeks prema oib-u na sličan način kao i prethodno opisani indeks prema jmbag-u. Svaki zapis indeksa definiran je strukturom:

```
struct idx_oib {
	long pozicija;
	char oib[12];
};
```

Gdje `pozicija` ima isto značenje kao i ranije, a u `oib` je pohranjen oib studenta. Zapisi ovog indeksa poredani su redosljedom rastućeg oib-a u leksičkom poretku, a indeks u datoteci zauzima mjesta od adrese 1.710.000 do zaključno adrese 1.869.999, tj. očekivano zauzima 160.000 bajtova.

U prvom dijelu projekta potrebno je napisati program koji mjeri performanse pretrage zapisa dohvaćanjem podataka iz datoteke na nekoliko načina: bez korištenja indeksa, korištenjem slijedne pretrage indeksa i korištenjem blokovske pretrage indeksa.

Testni podaci pohranjeni su u pripremljenoj tekstnoj (odnosno slijednoj formatiziranoj) datoteci `uzorci.txt`. U svakom retku datoteke zapisan jedan oib ili jmbag (svaki 10-znamenkasti broj smatra se oib-om, svaki 11-znamenkasti jmbag-om), a retci su odvojeni znakovnom kombinacijom `<CR><LF>`. Neki od oib-a iz testne datoteke su sadržani u `podaci.dat`, a neki nisu. Smije se pretpostaviti da `uzorci.txt` u svakom retku sadrži isključivo 10- ili 11-znamenkasti dekadski broj.

Za svaki od spomenutih načina dohvata potrebno je napisati jednu funkciju koja se poziva iz `main`-a, a funkcije su zadane prototipovima:

```
void stat_zapisi(FILE *podaci, FILE *uzorci);
void stat_idx_slijed(FILE *podaci, FILE *uzorci);
void stat_idx_blok(FILE *podaci, FILE *uzorci);
```

svaka od funkcija prima tokove podataka iz datoteka `studenti.dat` i `uzorci.txt` koji su prethodno (u `main`-u) otvoreni, a pozicija im je postavljena na početak datoteke. Svaka od ovih funkcija slijedi osnovni obrazac: u početku izvršavanja pamti trenutno vrijeme (vrijeme početka) nakon čega počinje dohvaćati redak po redak iz toka `uzorci`. Ako redak predstavlja oib tada funkcija pretražuje datoteku za zapisom s navedenim oib-om, a ako predstavlja jmbag, tada pretražuje datoteku za jmbag-om. Kada dođe do kraja datoteke uzorci, funkcija pamti trenutno vrijeme (vrijeme kraja) i izračunava vrijeme izvođenja. Nakon toga na standardni izlaz ispisuje poruku o mjerenim vremenima ovakvog oblika:

```
** pretraga po zapisima **
pronadjeno oib-a: 720 / 454
pronadjeno jmbag-a: 434 / 546
vrijeme izvodjenja: 13250 ms
```

Funkcija `stat_zapisi` za svaki uzorak pretražuje prvi dio datoteke `studenti.dat` i to zapis po zapis dok ne pronađe zapis sa traženim oib-om ili jmbag-om ili dođe do kraja prvog dijela datoteke.

Funkcija `stat_idx_slijed` pretražuje zapis po zapis sve zapise indeksa u drugom ili u trećem dijelu datoteke (ovisno traži li se po jmbag-u ili oib-u) dok ne pronađe zapis sa traženim jmbag-om ili oib-om ili dođe do kraja drugog ili trećeg dijela datoteke. Prva linija ispisa ove funkcije je „`** slijedna pretraga indeksa **`“.

Funkcija `stat_idx_blok` pretražuje indeksne zapise, ali po blokovima od sto elemenata: uspoređivati će traženi jmbag ili oib s onim pohranjenim u prvom zapisu odgovarajućeg indeksa, zatim u sto i prvom, dvjesto i prvom itd dok ne nađe odgovarajući blok te zatim unutar pravog bloka tražiti jedan po jedan zapis dok ne nađe pravi. Prva linija ispisa ove funkcije je „`** pretraga indeksa po blokovima **`“.

Projekt F, drugi dio

U drugom dijelu projekta potrebno je dugotrajne pretrage po datoteci zamijeniti pretragama po memorijskom indeksu implementiranom dvostruko povezanom listom te indeksom implementiranom višerazinskom dvostruko povezanom listom.

Jednostavan indeks implementiran je dvostruko povezanom listom definiranom strukturama:

```
/* atom liste */

struct at {
       char *podaci;
       long pozicija;
       struct at *sljed, *preth;
};
typedef struct at atom;
```

```
/* enkapsulira pocetak i kraj liste */

typedef struct {
       atom *pocetak, *kraj;
} indeks;
```

Svaki atom liste pokazuje na jedan zapis u datoteci tako što sadrži adresu zapisa u elementu `pozicija`. Element `*podaci` je pokazivač na niz znakova koji sadrži ili jmbag ili oib ovisno o indeksu.

Potrebno je napisati funkciju `izgradi_indekse` prototipa:

```
void izgradi_indekse(FILE *podaci, indeks *idx_jmbag, indeks *idx_oib);
```

koja prima adrese dviju listi (definiranih u `main`-u) i prvo gradi indeks po jmbag-u, zatim indeks po oib-u. Funkcija za izgradnju indeksa po jmbag-u pristupa drugom dijelu datoteke gdje su pohranjeni zapisi s datotečnim indeksom. Za svaki zapis kreira novi atom liste dinamičkom alokacijom memorije, inicijalizira potrebne vrijednosti (dinamički alocira i prostor za pohranu niza jmbag) i dodaje atom u listu `idx_jmbag`. Funkcija nakon uspješnog kreiranja indeks-liste po jmbag-u slično kreira indeks-listu po oib-u.

Za mjerenje performansi pretrage korištenjem indeksa napisati funkciju po uzoru na funkcije iz prvog dijela projekta, prototipa:

```
void stat_idx_1(indeks idx_jmbag, indeks idx_oib, FILE *uzorci);
```

Funkcija koristi podatke iz datoteke `uzorci.txt` i pronalazi ih korištenjem indeksa. Na kraju ispisuje poruku s rezultatima sličnu ranijim porukama uz prvu liniju izmjenjenu u „`** pretraga jednorazinskim indeksom **`“.

Za implementaciju višerazinskog indeksa pored spomenutih dodatno se koriste tipovi:

```
/* atom liste druge razine */

struct at_2 {
       char *podaci;
       atom osnovni;
       struct at_2 *sljed, *preth;
};
typedef struct at_2 atom_2;
```

```
/* enkapsulira pocetak i kraj liste te
osnovnu listu */

typedef struct {
       atom_2 *pocetak, *kraj;
       indeks *osnovna;
} indeks_2;
```

Umjesto pohrane pozicije u `atom_2` pohranjena je adresa atoma osnovnog indeksa na kojeg pokazuje ovaj atom. Lista druge razine sadrži 101 atom od kojih prvi pokazuje na prvi atom osnovnog indeksa, drugi na sto prvi atom osnovnog indeksa, treći na dvjesto prvi, a posljednji na posljednji element osnovnog indeksa.

Napisati funkciju za izgradnju indeksa druge razine na temelju indeksa prve razine, protip je:

```
void izgradi_indekse_2(indeks idx_jmbag, indeks idx_oib,
                       indeks_2 *idx_2_jmbag, indeks_2 *idx_2_oib);
```

Nakon izgradnje dvorazinskog indeksa ponoviti mjerenja korištenjem tog indeksa i ispisati rezultate po uzoru na ranije ispise, ali uz izmjenjenu prvu liniju ispisa u „`** pretraga dvorazinskim indeksom **`“. Za ovo mjerenje potrebno je napisati funkciju s prototipom:

```
void stat_idx_2(indeks_2 idx_jmbag, indeks_2 idx_oib, FILE *uzorci);
```

Gdje su argumenti funkcije slični onima za funkciju `stat_idx_1`.