Baze podataka 1

Rad sa datotekama u programskom jeziku C

1

Datoteke

- datoteka
  - kao logička struktura podataka (LSP)
    - struktura nad skupom pojava jednog tipa entiteta
      - struktura slogova, nad datim tipom sloga
      - često se posmatra kao linearna struktura slogova
  - kao fizička struktura podataka (FSP)
    - predstavlja jednu LSP koja može biti viđena kao
      - linearna struktura (niz) slogova ili
      - niz znakova ili bajtova
    - smeštenu na eksterni memorijski uređaj
    - zajedno sa informacijama o samom načinu smeštanja LSP na uređaj

2

2

Datoteke

- vrste datoteka
  - binarne
    - sadržaj čine podaci predstavljeni u binarnom obliku
    - proizvoljan tip sadržaja
  - tekstualne
    - sadržaj čine podaci koji odgovaraju karakterima nekog kodnog sistema:
      - ASCII, EBCDIC, UTF-8, UTF-16
    - *plain-text*

3

3

Datoteke

- radnje nad datotekama
  - otvaranje i zatvaranje
  - pristup
    - čitanje, pisanje, pozicioniranje
  - provera statusa
  - kreiranje i brisanje

4

4

C

- podržane vrste datoteka
  - tekstualne datoteke
    - sa konverzijom sadržaja
  - binarne datoteke
    - pristup na nivou bajta
- podržane radnje nad datotekama
  - otvaranje, zatvaranje
  - čitanje, pisanje, pozicioniranje
  - provera statusa
  - kreiranje, brisanje

5

5

C

- pogled na datoteku
  - datoteka kao niz bajtova
    - bez složenije podele na slogove
      - neophodno dodatno programiranje
    - koncept toka - *stream*
- pristup sadržaju datoteke
  - sekvencijalan
    - bajt po bajt (podatak po podatak) po redosledu unutar datoteke
  - direktan
    - pozicioniranje na proizvoljan bajt

6

6

C - Datotečki tip podatka

- tip podatka koji predstavlja datoteku u programu
  - FILE*
    - pokazivač na tip FILE
    - FILE je u osnovi struktura
      - deklarisana u zaglavlju stdio.h
    - primer deklaracije pokazivača datoteke:

```
FILE *f;
```

7

7

C stdio.h //MinGW

```
...
/*
 * The structure underlying the FILE type.
 *
 * Some believe that nobody in their right mind should make use of the
 * internals of this structure. Provided by Pedro A. Aranda Gutiirrez
 * <paag@tid.es>.
 */
#ifndef _FILE_DEFINED
#define _FILE_DEFINED
typedef struct _iobuf
{
       char*   _ptr;
       int     _cnt;
       char*   _base;
       int     _flag;
       int     _file;
       int     _charbuf;
       int     _bufsiz;
       char*   _tmpfname;
} FILE;
#endif /* Not _FILE_DEFINED */
...
```

8

8

C - Datotečki tip podatka

- svaki ulaz i izlaz C programa se posmatra kao datoteka
- tri ugrađene datoteke
  - *stdin* - standardni ulaz
    - podrazumevano učitavanje vrednosti sa tastature
  - *stdout* - standardni izlaz
    - podrazumevano prikazivanje vrednosti na ekranu
  - *stderr* - standardni izlaz za poruke
    - poruke o greškama, prikazuje na ekranu
- infrastruktura za rad sa datotekama
  - `<stdio.h>`

9

9

C - Otvaranje datoteke

- funkcija
- `FILE *fopen (const char *naziv, const char *rezim);`
  - `naziv` - naziv datoteke koja treba da bude otvorena
    - u skladu sa pravilima operativnog sistema
  - `rezim` - oznaka načina korišćenja datoteke
  - povratna vrednost
    - pokazivač na otvorenu datoteku ili
    - NULL vrednost ako otvaranje nije uspešno izvršeno
      - obavljati proveru povratne vrednosti

10

10

C - Otvaranje datoteke

- mogući režimi rada - tekstualne datoteke
  - "r"
    - čitanje postojeće datoteke
  - "w"
    - pisanje u već postojećoj datoteci
      - prethodni sadržaj biva uništen
    - pisanje u novoj datoteci
      - automatski se kreira nova datoteka ako ne postoji neka sa datim nazivom
  - "a"
    - *append*
    - dodavanja sadržaja na kraj postojeće datoteke
    - ako datoteka sa datim nazivom ne postoji, biće kreirana

11

11

C - Otvaranje datoteke

- mogući režimi rada - tekstualne datoteke
  - "r+" - čitanje i pisanje u postojećoj datoteci
  - "w+" - čitanje i pisanje
    - u postojećoj datoteci
      - prethodni sadržaj biva uništen
    - u novoj datoteci
      - automatski se kreira nova datoteka ako ne postoji neka sa datim nazivom
  - "a+"
    - čitanje i dodavanja sadržaja na kraj postojeće datoteke
    - ako datoteka sa datim nazivom ne postoji, biće kreirana
  - primer:
    - `FILE *f = fopen("sadrzaj.txt","r+");`

12

12

C - Otvaranje datoteke

- mogući režimi rada - binarne datoteke
  - dodavanje slova *b* u opis režima
    - logika ostaje ista kao kod tekstualnih datoteka
  - moguće kombinacije
    - "rb", "wb", "ab"
    - "r+b", "w+b", "a+b"
      - "rb+", "wb+", "ab+" (identično kao prethodna tri)
  - primer:
    - `FILE *f = fopen("sadrzaj.bin","r+b");`

13

13

C - Zatvaranje datoteke

- funkcija
- `int fclose(FILE *f);`
  - `f` - pokazivač na prethodno otvorenu datoteku koja treba da bude zatvorena
  - povratna vrednost
    - 0 ako je zatvaranje uspešno
    - konstanta EOF ako je došlo do greške
- automatsko zatvaranje
  - pri završetku izvršavanja programa
    - kraj *main* funkcije
    - sve otvorene datoteke bivaju zatvorene
  - ne oslanjati se na to
    - eksplicitno zatvoriti svaku otvorenu datoteku

14

14

C - Primer 1

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    FILE *f;
    char n[] = "sadrzaj.txt";

    if((f = fopen(n,"r")) == NULL){
        printf("Datoteka <%s> nije uspesno otvorena.\n", n);
        exit(1);
    }

    printf("Datoteka <%s> je uspesno otvorena.\n", n);
    printf("Zatvaranje datoteke <%s>...\n", n);

    if(fclose(f) == EOF)
        printf("\tNastupila je greska tokom zatvaranja!\n");
    else
        printf("\tZatvaranje uspesno!\n");

    return 0;
}
```

15

15

C - Rad sa txt datotekama

- čitanje i pisanje
  - obavlja se sekvencijalno
    - od početka datoteke
  - svaki sledeći pristup
    - iza poslednjeg mesta kojem je pristupano u prethodnom čitanju ili pisanju

16

16

C - Rad sa txt datotekama

- rad sa znakovima – čitanje
- `int fgetc(FILE *f);`
  - čitanje pojedinačnog znaka
  - `f` - pokazivač na prethodno otvorenu datoteku
  - povratna vrednost
    - kôd pročitanog znaka
    - konstanta EOF ako se došlo do kraja datoteke ili se desila neka greška
- `char *fgets(char *tekst, int n, FILE *f);`
  - čitanje teksta iz datoteke
    - do znaka '\n' ili
    - ukupno n-1 znakova
  - upisuje '\0' na kraj stringa
  - `tekst` - string koji prihvata učitani sadržaj
  - `n` - maksimalni broj znakova za učitavanje (uključujući '\0')
  - `f` - pokazivač na prethodno otvorenu datoteku
  - povratna vrednost
    - adresa na koju je upisan novi sadržaj - adresa stringa `tekst`
    - konstanta NULL ako se došlo do kraja datoteke ili se desila neka greška

17

C - Rad sa txt datotekama

- rad sa znakovima - pisanje
- `int fputc(int c, FILE *f);`
  - upisivanje pojedinačnog znaka
  - `c` - kod znaka za upis
  - `f` - pokazivač na prethodno otvorenu datoteku
  - povratna vrednost
    - kôd upisanog znaka
    - konstanta EOF ako se desila neka greška
- `int fputs(const char *tekst, FILE *f);`
  - upisivanje stringa u formi jednog reda datoteke
    - znak '\n' automatski dodaje na kraj
  - `tekst` - string koji predstavlja sadržaj za upis
  - `f` - pokazivač na prethodno otvorenu datoteku
  - povratna vrednost
    - ne-negativna vrednost
    - konstanta EOF ako se desila neka greška

18

18

C - Rad sa txt datotekama

- rad sa znakovima - ulazna i izlazna konverzija
- `int fscanf(FILE *f, const char *form, a1, a2, ...);`
  - učitavanje, izvršenje konverzije za a1, a2 ... prema zadatom formatu
  - `f` - pokazivač na prethodno otvorenu datoteku
  - `form` - specifikacija konverzije
  - `a1, a2, ...` - adrese za smeštanje sadržaja učitanog iz datoteke
  - povratna vrednost
    - broj uspešno konvertovanih i smeštenih podataka
    - konstanta EOF ako se pojavio kraj datoteke ili desila neka greška pre prve konverzije
- `int fprintf(FILE *f, const char *form, a1, a2, ...);`
  - upisivanje, izvršenje konverzije za a1, a2 ... prema zadatom formatu
  - `f` - pokazivač na prethodno otvorenu datoteku
  - `form` - specifikacija konverzije
  - `a1, a2, ...` - adrese za čitanje sadržaja za upis u datoteku
  - povratna vrednost
    - broj ispisanih znakova
    - negativna vrednost (-1) ako se desila neka greška pri konverziji

19

19

C – Primer 2

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    FILE *f;
    char n[] = "test.txt";
    char c;

    if((f = fopen(n,"r")) == NULL){
        printf("Datoteka <%s> nije uspesno otvorena.\n", n);
        exit(1);
    }

    printf("Datoteka <%s> je uspesno otvorena.\n", n);
    printf(">>Tekstualni sadrzaj datoteke <%s>...\n", n);

    while((c = fgetc(f)) != EOF)
        putchar(c);

    printf("\n>>Kraj sadrzaja datoteke <%s>...\n", n);
    fclose(f);

    return 0;
}
```

20

20

C - Zadatak 1

- učitati niz od *n* celih brojeva sa tastature
  - pri čemu je korisnik prethodno zadao *n* isto preko tastature
- kreirati tekstualnu datoteku sa nazivom *naz* koja će u svakom redu sadržati po jedan učitani broj, razmak i vrednost tog broja dignutu na drugi stepen
  - pri čemu je korisnik prethodno zadao *naz* preko tastature
  - primer
    - za niz {3,5,7} datoteka može izgledati ovako

```
3 9
5 25
7 49
```

21

21

C - Rad sa bin datotekama

- čitanje i pisanje
  - obavlja se sekvencijalno
    - od početka datoteke
    - svaki sledeći pristup
      - iza poslednjeg mesta kojem je pristupano u prethodnom čitanju ili pisanju
  - mogućnost pozicioniranja
    - zadavanje pomeraja u odnosu na referentnu tačku
      - početak, trenutna pozicija, kraj datoteke
    - sledeći pristup od poslednje postavljene pozicije
  - programer vodi računa o strukturi datoteke

22

22

C - Rad sa bin datotekama

- rad sa podacima - ulaz
- `int fread(void *sadrzaj, int duz, int n, FILE *f);`
  - učitavanje zadatog broja podataka u memoriju iz datoteke od prethodne pozicije
  - nova pozicija u datoteci na mestu iza poslednjeg učitanog bajta
  - `sadrzaj` - memorijska adresa od koje počinje smeštanje učitanih podataka
  - `duz` - dužina pojedinačnog podatka za učitavanje (u bajtovima)
  - `n` - broj podataka koji treba učitati iz datoteke
  - `f` - pokazivač na prethodno otvorenu datoteku
  - povratna vrednost
    - broj uspešno pročitanih podataka
  - indikator greške učitavanja
    - `feof(), ferror()`

23

23

C - Rad sa bin datotekama

- rad sa podacima - izlaz
- `int fwrite(const void *sadrzaj, int duz, int n, FILE *f);`
  - upisivanje zadatog broja podataka zadate dužine iz memorije u datoteku od mesta završetka poslednjeg pristupa
  - ako pozicija nije kraj datoteke, dolazi do prepisivanja sadržaja
  - ako kapacitet datoteke nije dovoljan, datoteka se povećava
  - nova pozicija u datoteci na mestu iza poslednjeg upisanog bajta
  - `sadrzaj` - memorijska adresa od koje počinje čitanje podataka za smeštanje u datoteku
  - `duz` - dužina pojedinačnog podatka za smeštanje (u bajtovima)
  - `n` - broj podataka koji treba smestiti u datoteku
  - `f` - pokazivač na prethodno otvorenu datoteku
  - povratna vrednost
    - broj uspešno upisanih podataka
    - ako je došlo greške onda je ta vrednost manja od `n`

24

24

C - Rad sa bin datotekama

- rad sa podacima - pozicioniranje
- `int fseek(FILE *f, long offset, int ref);`
  - postavljanje trenutne pozicije unutar datoteke
  - može i za tekstualne datoteke
  - neophodna kada posle radnje čitanja dolazi pisanje (i obrnuto)
  - `f` - pokazivač na prethodno otvorenu datoteku
  - `offset` - udaljenost nove pozicije od referentne tačke (u bajtovima)
  - `ref` - referentna tačka u odnosu na koju se posmatra udaljenost nove pozicije
    - `SEEK_SET` - početak datoteke
    - `SEEK_CUR` - trenutna pozicija unutar datoteke
    - `SEEK_END` - kraj datoteke
  - povratna vrednost
    - oznaka greške - ako vrednost različita od 0

25

25

C - Rad sa bin datotekama

- rad sa podacima - pozicioniranje
- `long ftell(FILE *f);`
  - nalaženje trenutne pozicije unutar datoteke
  - `f` - pokazivač na prethodno otvorenu datoteku
  - povratna vrednost
    - trenutna pozicija u bajtovima u odnosu na početak datoteke
    - -1L - oznaka greške
- `void rewind(FILE *f);`
  - postavljanje pozicije na početak datoteke
  - `f` - pokazivač na prethodno otvorenu datoteku
  - nema povratnu vrednost

26

26

C - Rad sa datotekama

- indikacija greške
  - vrednosti indikatora se postavljaju kroz bočne efekte mnogih prethodnih funkcija
  - dva indikatora
    - indikator kraja datoteke
    - indikator greške
  - funkcije
  - `void clearerr(FILE *f);`
    - briše vrednosti oba indikatora
    - `f` - pokazivač na prethodno otvorenu datoteku
  - `int feof(FILE *f);`
    - provera indikatora kraja datoteke
    - `f` - pokazivač na prethodno otvorenu datoteku
    - povratna vrednost različita od 0 ako je indikator uključen
  - `int ferror(FILE *f);`
    - provera indikatora greške za datoteku
    - `f` - pokazivač na prethodno otvorenu datoteku
    - povratna vrednost različita od 0 ako je indikator uključen
  - globalna promenljiva - `errno` u `<errho.h>` - kod greške

27

27

C – Primer 3

```
#include <stdio.h>
#include <stdlib.h>
#define ROWCOUNT 16

int main() {
    FILE *f;
    char c, n[] = "test.txt";
    long cnt = 0;

    if((f = fopen(n,"rb")) == NULL){
        printf("Datoteka <%s> nije uspesno otvorena.\n", n);
        exit(1);
    }

    printf(">>Binarni sadrzaj datoteke <%s>...\n", n);

    while(fread(&c, sizeof(char), 1, f)){

        if(cnt%ROWCOUNT == 0)
            printf("\n");

        printf("0x%02X ", c);
        cnt++;
    }

    printf("\n>>Kraj sadrzaja datoteke <%s>...\n", n);
    fclose(f);
    return 0;
}
```

28

28

C - Zadatak 2

- učitati niz od *n* celih brojeva sa tastature
  - pri čemu je korisnik prethodno zadao *n* isto preko tastature
- kreirati binarnu datoteku sa nazivom *naz* koja će za svaki učitani broj sadržati njegovu vrednost kao i tu vrednost dignutu na drugi stepen
  - pri čemu je korisnik prethodno zadao *naz* preko tastature
  - primer
    - za niz {3,5,7} datoteka može izgledati ovako -*little endian*

```
03 00 00 00 09 00 00 00
05 00 00 00 19 00 00 00
07 00 00 00 31 00 00 00
```

29

29

C - Zadatak 3

- za datoteku čiji je naziv uneo korisnik preko tastature
  - prikazati veličinu datoteke
    - u B, kB i MB
    - oslanjajući se na prethodne funkcije
  - prikazati odgovarajuću poruku u slučaju greške prilikom nalaženja i otvaranja datoteke

30

30

C - Zadatak 4

- omogućiti korisniku da bira jednu od četiri opcije u radu sa binarnom datotekom
  - otvaranje datoteke sa nazivom koji korisnik unese
    - kao i zatvaranje prethodne a otvaranje nove datoteke
  - prikaz bajta iz otvorene datoteke pri čemu korisnik zadaje rednu poziciju sa koje želi da učita bajt
  - izmenu bajta u otvorenoj datoteci pri čemu korisnik zadaje rednu poziciju kao i vrednost koja treba da se upiše u datoteku na toj poziciji
  - kraj rada sa programom
    - kao i zatvaranje otvorene datoteke

31

31

C - Zadatak 5

- omogućiti korisniku da bira jednu od tri opcije
  - unos koordinata tačke u 3D prostoru i upis te tačke u datoteku *tacke.bin* (realizovati kao posebnu funkciju)
    - koordinata je tipa *double*
  - automatsko računanje obima najmanje sfere koja obuhvata sve tačke koje su sačuvane u *tacke.bin* (realizovati kao posebnu funkciju)
    - sa prolazom kroz datoteku radi čitanja koordinata tačaka
  - kraj rada sa programom
    - kao i zatvaranje datoteke

32

32

C - Zadatak 5-2

- za prethodni program
  - napisati funkciju koja će omogućiti korisniku da prikaže koordinate *i*-te tačke iz datoteke
  - napisati funkciju koja će omogućiti korisniku da upiše u datoteku nove vrednosti koordinata *i*-te tačke
    - umesto postojećih vrednosti
  - napisati funkciju za brisanje *i*-te tačke iz datoteke
    - realizovati kao logičko brisanje
  - dodati prethodne opcije u korisnički meni
  - prilagoditi funkcije da podrže logičko brisanje

33

33

Baze podataka 1

Osnovni pojmovi struktura podataka

sa realizacijom u programskom jeziku C

34

Klasifikacija struktura podataka

Klasifikacija 1

- prema dozvoljenom broju neposrednih prethodnika i sledbenika jednog čvora strukture
  - linearne strukture
  - strukture stabla
  - mrežne strukture

35

35

Klasifikacija struktura podataka

Klasifikacija 2

- prema nivou apstraktnosti skupa čiji elementi snabdevaju čvorove i ivice grafa semantikom
  - strukture nad skupom obeležja (strukture obeležja)
    - nazivaju i logičkim strukturama
    - visok nivo apstraktnosti
  - strukture nad skupom podataka (strukture podataka) - mogu biti
    - logičke strukture podataka
    - fizičke strukture podataka

36

36

Linearne strukture podataka

- linearne strukture podataka - LinSP
- svaki čvor grafa
  - može da ima najviše jednog direktnog prethodnika
  - može da ima najviše jednog direktnog sledbenika
- klasifikacija
  - aciklične – lanci, otvorene liste i nizovi
  - ciklične – zatvorene liste ili prsten

37

37

Predstavljanje LinSP

- u zavisnosti od postupka dodavanja i brisanja elemenata, aciklične linearne strukture se nazivaju:
  - n-torka ili vektor (ako se dodavanje i brisanje ne vrši)
  - stek (ako se dodavanje i brisanje vrši samo na jednom kraju)
  - red (ako se dodavanje vrši na jednom, a brisanje na drugom kraju)
  - dek (ako se i dodavanje i brisanje vrši na oba kraja)

38

38

Predstavljanje LinSP

- dva osnovna postupka za predstavljanje linearnih struktura podataka u operativnoj memoriji:
  - sekvencijalna reprezentacija – niz
  - spregnuta reprezentacija – lista

39

39

Sekvencijalna predstava LinSP

- lokacije operativne memorije su uređene putem celobrojnog adresnog mehanizma
- čvorovi linearne strukture podataka se mogu tako memorisati da se sledećem čvoru može pristupiti jednostavnim povećanjem adrese lokacije tekućeg čvora

40

40

Stek

- stek se može sekvencijalno reprezentovati na sličan način kao i niz fiksne dužine
- problem: ograničen broj elemenata
- metode:
  - pop (skini element sa vrha steka)
  - push (stavi element na vrh steka)
  - peek (pročitaj element na vrhu steka)
  - empty (da li je stek prazan)

41

41

Red

- realizacija pomoću niza fiksne dužine
- ponovo se koriste lokacije oslobođene pri čitanju elemenata iz reda, tako da do prekoračenja može doći tek ako treba upisati N-ti element u red
- metode:
  - ubaci
  - čitaj
  - nađi
- referentni pokazivači: *levi* i *desni*
  - *levi*: pokazuje na poziciju sa koje treba pročitati sledeći element
  - *desni*: pokazuje na prvu slobodnu poziciju na koju je moguće upisati sledeći element

42

42

Spregnuta predstava LinSP

- svaki čvor strukture se smešta u jednu lokaciju
- lokacija ima jedno polje za smeštanje pokazivača ka lokaciji neposredno narednog čvora
- neposredno susedni čvorovi nisu, u opštem slučaju, u fizički neposredno susednim lokacijama

43

43

Spregnuta predstava LinSP

- Stek
  - pokazivač na vrh steka
- Red
  - dva referentna pokazivača: *čelo* i *kraj*
    - *čelo* sadrži adresu lokacije čvora koji treba prvi da bude opslužen (pročitan i eliminisan iz reda)
    - *kraj* sadrži adresu lokacije čvora koji je poslednji upisan u red
  - elementi reda su spregnuti od čela ka kraju

44

44

Lista

- zadatak 1
  - spregnuta reprezentacija linearne strukture podataka tipa lista

45

45

Zadatak 1

Napisati C program za evidenciju studenata.

- Svaki student je opisan:
  - imenom (do 20 karaktera),
  - prezimenom (do 20 karaktera),
  - brojem indeksa (do 10 karaktera) i
  - godinom upisa studija.
- Program treba da obezbedi sledeće:
  - Unos podataka o studentima na proizvoljnu poziciju u odgovarajuću strukturu,
  - Prikaz svih studenata iz odgovarajuće strukture,
  - Brisanje studenta sa zadatim brojem indeksa, iz odgovarajuće strukture
  - Čitanje podataka o studentima iz datoteke *studenti.txt*, i smeštanje u odgovarajuču strukturu
    - podaci su delimitirani jednim blank znakom
  - Snimanje strukture u datoteku *studenti.txt*.

46

46

Čvor dinamičke strukture

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

#define BRINDEKS_SIZE 11
#define IME_SIZE 21
#define PREZIME_SIZE 21

typedef struct student {
   char ime[IME_SIZE];
   char prezime[PREZIME_SIZE];
   char brindeks[BRINDEKS_SIZE];
   int godupis;
   struct student *next;
} TStudent;

int main() {
    TStudent *glava = NULL;
    return 0;
}
```

47

47

Prikaz svih studenata iz dinamičke strukture

```
void prikazStudenata(TStudent* glava) {

    puts("PRIKAZ STUDENATA");
    puts("---------------------------------------");

    while (glava){
        printf("Student: %s, %s, %s, %d\n",
         glava->ime, glava->prezime,
         glava->brindeks, glava->godupis);
         glava = glava->next;
    }

    puts("---------------------------------------");
}
```

48

48

Broj studenata u dinamičkoj strukturi

```
int brojStudenata(TStudent* glava) {

    int brojStd = 0;

    while (glava) {
        brojStd++;
        glava = glava->next;
    }

    return brojStd;
}
```

49

49

Dodavanje novog čvora (studenta) u dinamičku strukturu na proizvoljnu poziciju

```
void dodajStudenta(TStudent **glava,char *ime, char *prezime, char*
   brindeks, int godupis, int pozicija) {
    if((pozicija >= 0) && (pozicija <= brojStudenata(*glava)+1)){
        int i;
        TStudent *noviS= (TStudent*)malloc(sizeof(TStudent));
        TStudent *tekuci= *glava, *prethodni = *glava;
        strcpy(noviS->ime, ime);
        strcpy(noviS->prezime, prezime);
        strcpy(noviS->brindeks, brindeks);
        noviS->godupis = godupis;
        noviS->next = NULL;
        for(i = 0; i < pozicija; i++){
            prethodni = tekuci;
            tekuci = tekuci->next;
        }
        noviS->next = tekuci;
        if(tekuci == *glava)
            *glava = noviS;
        else
            prethodni->next = noviS;
    }
}
```

50

50

Brisanje studenta iz dinamičke strukture po broju indeksa

```
void obrisiStudenta(TStudent **glava, char *indeks) {
    TStudent *tekuci = *glava, *prethodni = NULL,
     *temp = NULL;
    while (tekuci) {
        if (!strcmp(tekuci->brindeks, indeks)) {
            temp = tekuci;
            tekuci = tekuci->next;
            if (temp == *glava) {
                *glava = (*glava)->next;
            } else {
                prethodni->next = tekuci;
            }
            free(temp);
        } else {
            prethodni= tekuci;
            tekuci = tekuci->next;
        }
    }
}
```

51

51

Brisanje svih čvorova dinamičke strukture

```
void obrisiStudente(TStudent **glava) {

    TStudent *temp;

    while (*glava) {
        temp = *glava;
        *glava = (*glava)->next;
        free(temp);
    }
}
```

52

52

Učitavanje studenata iz tekstualne datoteke *studenti.txt*

```
void ucitajStudente(TStudent** glava) {

    FILE *f = fopen("studenti.txt", "r");
    char ime[IME_SIZE], prezime[PREZIME_SIZE],
    brindeks[BRINDEKS_SIZE];
    int godupis;

    while (fscanf(f, "%s %s %s %d", ime, prezime,
          brindeks,  &godupis) != EOF)
         dodajStudenta(glava, ime, prezime,
          brindeks, godupis, brojStudenata(*glava));

    fclose(f);
}
```

53

53

Čuvanje podataka o studentima iz dinamičke strukture u tekstualnoj datoteci *studenti.txt*

```
void sacuvajStudente(TStudent *glava) {

    FILE *f = fopen("studenti.txt", "w");

    while (glava != NULL) {
       fprintf(f, "%s %s %s %d\n", glava->ime,
        glava->prezime, glava->brindeks, glava->godupis);
       glava = glava->next;
    }

    fclose(f);
}
```

54

54

Zadatak 2

- Zadata je binarna datoteka *tacke.bin*, koja sadrži koordinate tačke u ravni. Program treba da omogući:
  - Unos nove tačke u odgovarajuću strukturu,
    - Tačke se dodaju u strukturu u neopadajućem redosledu prema udaljenosti od koordinatnog početka.
  - Prikaz svih tacaka iz odgovarajuće strukture,
  - Brisanje tačaka iz strukture koje imaju manju udaljenost od zadate, od koordinatnog početka,
  - Učitavanje koordinata tačaka iz datoteke *tacke.bin* u odgovarajuću strukturu,
  - Snimanje strukture u datoteku *tacke.bin.*

55

55

Čvor dinamičke strukture

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

typedef struct {
    double x, y;
    struct tacka *next;
} Tacka;

int main() {
    Tacka *glava = NULL;

    return 0;
}
```

56

56

Rastojanje tačke od koordinatnog početka

```
double rastojanje (double x, double y) {
    return sqrt(x*x + y*y);
}
```

57

57

Unos nove tačke u odgovarajuću strukturu

```
void dodajTacku(Tacka** glava, double x, double y) {
    Tacka *novaTacka = (Tacka*) malloc(sizeof(Tacka));
    novaTacka->x = x;
    novaTacka->y = y;
    novaTacka->next = NULL;
    Tacka *tek = *glava, *pre = *glava;
    if (*glava) {
        while (tek) {
            if (rastojanje(novaTacka->x, novaTacka->y) <
               rastojanje(tek->x, tek->y)){
                novaTacka->next = tek;
                if (tek == *glava)
                    *glava = novaTacka;
                else
                    pre->next = novaTacka;
                break;
            }
            pre = tek;
            tek = tek->next;
            if (tek == NULL)
                pre->next = novaTacka;
        }
    }
    else
        *glava = novaTacka;
}
```

58

58

Brisanje tačaka iz strukture koje imaju manju udaljenost od zadate, od koordinatnog početka

```
void obrisiTacku(Tacka **glava, double ras) {
    Tacka *tekuci = *glava, *prethodni = NULL, *temp = NULL;
    printf("BRISANJE  TACAKA SA RASTOJANJEM OD (0,0) MANJIM OD  %.2lf\n",
           ras);
    puts("-------------------------------------------------");
    while (tekuci){
        if (rastojanje(tekuci->x, tekuci->y) <= ras){
            temp = tekuci;
            tekuci = tekuci->next;
            if (temp == *glava) {
                *glava = (*glava)->next;
            } else {
                prethodni->next = tekuci;
            }
            printf("Brisanje tacke : (%.2lf,%.2lf)\n",tekuci->x,
            tekuci->y);
            free(tekuci);
        } else {
            prethodni= tekuci;
            tekuci = tekuci->next;
        }
    }
    puts("-------------------------------------------------");
}
```

59

59

Prikaz svih tačaka

```
void ispisiTacke(Tacka *glava) {
    puts("TACKE:");
    puts("----------------------------------");
    while (glava!=NULL) {
        printf("Tacka (%.2lf, %.2lf) sa ", glava->x,
         glava->y);
        printf("rastojanjem od  tacke (0,0): %.2lf\n",
         rastojanje(glava->x, glava->y));

        glava = glava->next;
    }
    puts("----------------------------------");
}
```

60

60

Brisanje dinamičke strukture

```
void obrisiTacke(Tacka **glava) {
    Tacka *temp;
    while(*glava) {
        temp = (*glava);
        *glava = (*glava)->next;
        printf("Brisanje tacke : ");
        printf("(%.2lf,%.2lf)\n", temp->x, temp->y);
        free(temp);
    }
}
```

61

61

Snimanje strukture u datoteku *tacke.bin*

```
void sacuvajTacke(Tacka *glava) {
    FILE *f = fopen("tacke.bin", "w");

    while (glava != NULL){
        fwrite(&glava->x, sizeof(double), 1, f);
        fwrite(&glava->y, sizeof(double), 1, f);
        glava = glava->next;
    }

    fclose(f);
}
```

62

62

Učitavanje koordinata tačaka iz datoteke tacke.bin u odgovarajuću strukturu

```
void ucitajTacke(Tacka** glava) {
    FILE *f = fopen("tacke.bin", "r");

    double posX, posY;

    while (1) {
        if ((!fread(&posX, sizeof(double), 1, f)) ||
            (!fread(&posY, sizeof(double), 1, f)))
            break;
        dodajTacku(glava, posX, posY);
    }

    fclose(f);
}
```

63

63

Zadatak 3

- Proširiti zadatak 1 sledećim funkcionalnostima
  - Prilikom dodavanja novog studenta, omogućiti proveru da li već postoji student sa tim brojem indeksa,
  - Čuvanje podataka o studentima u binarnoj datoteci *studenti.bin,*
  - Čitanje podataka o studentima iz binarne datoteke *studenti.bin* i njihovo smeštanje u odgovarajuću strukturu,
  - Za svakog studenta potrebno je evidentirati broj položenih ispita,
    - omogućiti izmenu broja položenih ispita za studente, na osnovu broja indeksa.

64

64