Универзитет у Београду
Факултет организационих наука
Катедра за софтверско инжењерство
Лабораторија за софтверско инжењерство

Програмирање 1
Збирка задатака

Аутори:

Саша Лазаревић

Илија Антовић

Душан Савић

Милош Милић

Војислав Станојевић

Татјана Стојановић

Београд, 2019.

Садржај

1. Структуре 1
   1.1. Решени задаци 1
   1.2. Задаци за вежбање 5
2. Листе 7
   2.1. Решени задаци 7
   2.2. Задаци за вежбање 13
   2.3. Задаци са колоквијума 14
   2.4. Задаци са испитних рокова 16
3. Датотеке 21
   3.1. Решени задаци 21
   3.2. Задаци за вежбање 31
   3.3. Задаци са колоквијума 32
   3.4. Задаци са испитних рокова 36
4. Пример испитног рока 50
5. Прилози 56
   5.1. Фазе процеса израде програма 56
   5.2. Коришћење интегрисаног развојног окружења *Visual Studio 2010* 56
   5.3. Структура заглавља програма 56
   5.4. Процес израде програма – од монолитног до структурираног програма 57
      Верзија 1 57
      Верзија 2 58
      Верзија 3 59
      Верзија 4 60
      Верзија 5 62
      Верзија 6 63
   5.5. Процес израде програма – израда корисничког менија 65
      Верзија 1 65
      Верзија 2 68
      Верзија 3 70

1. Структуре

У овом одељку приказане су структуре. Дати су решени задаци и задаци за вежбање.

1.1. Решени задаци

1. Дата је структура *производ (шифра, назив, цена)*. Иницијализовати производ и приказати његове податке.

```
#include <stdio.h>
#include <string.h>
typedef struct PROIZVOD {
	int sifra;
	char naziv[50];
	double cena;
}TPROIZVOD;
int main(void) {
	struct PROIZVOD p1 = { 1, "Smoki", 90.90 };
	TPROIZVOD p2 = { 2, "Milka", 123.00 };
	printf("Proizvod p1:\n");
	printf("\tSifra: %d\n", p1.sifra);
	printf("\tNaziv: %s\n", p1.naziv);
	printf("\tCena: %8.2lf\n", p1.cena);
	printf("Proizvod p2:\n");
	printf("\tSifra: %d\n", p2.sifra);
	printf("\tNaziv: %s\n", p2.naziv);
	printf("\tCena: %8.2lf\n", p2.cena);
	return 0;
}
```

2. Дата је структура *производ (шифра, назив, цена)*. Написати процедуру за унос новог производа. Написати процедуру за приказ података о производу.

```
#include <stdio.h>
#include <string.h>
typedef struct PROIZVOD {
	int sifra;
	char naziv[50];
	double cena;
}TPROIZVOD;
void stampaj_proizvod(TPROIZVOD p) {
	printf("%7d%20s%8.2lf", p.sifra, p.naziv, p.cena);
	printf("\n");
}
TPROIZVOD prihvati_proizvod(void) {
	TPROIZVOD p;
	printf("Unesi sifru: ");
	scanf("%d", &p.sifra);
	printf("Unesi naziv: ");
	getchar();
	gets(p.naziv);
	printf("Unesi cenu: ");
	scanf("%lf", &p.cena);
	return p;
}
int main(void) {
	struct PROIZVOD p1 = prihvati_proizvod();
	TPROIZVOD p2 = prihvati_proizvod();
	stampaj_proizvod(p1);
	stampaj_proizvod(p2);
	return 0;
```

```
}
```

3. Дата је структура *производ (шифра, назив, цена)*. Написати процедуру за унос новог производа у низ производа. Написати процедуру за приказ података о низу производа.

```
#include <stdio.h>
#include <string.h>
typedef struct PROIZVOD {
	int sifra;
	char naziv[50];
	double cena;
}TPROIZVOD;
void stampaj_proizvod(TPROIZVOD p) {
	printf("%7d%20s%8.2lf", p.sifra, p.naziv, p.cena);
	printf("\n");
}
TPROIZVOD prihvati_proizvod(void) {
	TPROIZVOD p;
	printf("Unesi sifru: ");
	scanf("%d", &p.sifra);
	printf("Unesi naziv: ");
	getchar();
	gets(p.naziv);
	printf("Unesi cenu: ");
	scanf("%lf", &p.cena);
	return p;
}
void dodaj_proizvod(TPROIZVOD x[], int *n, TPROIZVOD p) {
	x[*n] = p;
	*n = *n + 1;
}
void prikaz_proizvoda(TPROIZVOD x[], int n) {
	int i;
	for (i = 0; i < n; i++) {
		stampaj_proizvod(x[i]);
	}
}
int main(void) {
	TPROIZVOD x[100];
	int n = 0;
	TPROIZVOD p1 = prihvati_proizvod();
	dodaj_proizvod(x, &n, p1);
	TPROIZVOD p2 = prihvati_proizvod();
	dodaj_proizvod(x, &n, p2);
	prikaz_proizvoda(x, n);
	return 0;
}
```

4. Дати су низови предмета, студената и пријава. Написати следеће потпрограме:
а)Написати процедуру за унос нових пријава, тако што корисник уноси шифру предмета, број индекса и добијену оцену. Потребно је онемогућити чување нове пријаве уколико предмет и/или студент не постоје у одговарајућим низовима.

```
#include <stdio.h>
typedef struct predmet {
	char sifra[7];
	char naziv[20];
	int broj_ESP;
} TPREDMET;
typedef struct indeks {
	int godina;
	int broj;
```

```
} TINDEKS;
typedef struct prijava {
	TINDEKS indeks;
	char sifra_predmeta[7];
	int ocena;
} TPRIJAVA;
typedef struct student {
	TINDEKS indeks;
	char prezime[20];
	char ime[20];
	int ukupan_ESP;
} TSTUDENT;
typedef TPREDMET xpredmeti[50];
typedef TSTUDENT xstudenti[50];
typedef TPRIJAVA xprijave[50];

xpredmeti predmeti = {
		{"pred-1", "matematika", 6},
		{"pred-2", "programiranje", 4},
		{"pred-3", "OIKT", 8}
};
int broj_predmeta = 3;

xstudenti studenti = {
		{{2015, 5}, "Peric", "Pera", 0},
		{{2015, 55}, "Petrovic", "Petar", 0},
		{{2015, 555}, "Janic", "Jana", 0},
		{{2015, 3}, "Markovic", "Marko", 0},
		{{2015, 33}, "Andjelkovic", "Ana", 0}
};
int broj_studenata = 5;

TPREDMET* vrati_predmet(char sifra_predmeta[7]) {
	int i;
	for (i = 0; i < broj_predmeta; i++) {
		if (strcmp(sifra_predmeta, predmeti[i].sifra) == 0) {
			return &predmeti[i];
		}
	}
	return NULL;
}
TSTUDENT* vrati_studenta(int godina, int broj_indeksa) {
	int i;

	for (i = 0; i < broj_studenata; i++) {
		if (studenti[i].indeks.godina == godina &&
		    studenti[i].indeks.broj == broj_indeksa) {
			return &studenti[i];
		}
	}
	return NULL;
}
void ubaci_prijavu(TPRIJAVA nova_prijava, xprijave xprijave, int * broj_prijava) {
	xprijave[*broj_prijava] = nova_prijava;
	*broj_prijava = *broj_prijava + 1;
}
void unesi_prijavu(xprijave xprijave, int *broj_prijava) {
	TPRIJAVA nova_prijava;
	int godina, broj_indeksa, ocena;
```

```
	char sifra_predmeta[7];
	TPREDMET * predmet;
	TSTUDENT * student;
	printf("\nUnesite broj indeksa u formatu godina/broj:");
	scanf("%d/%d", &godina, &broj_indeksa);
	printf("\nUnesite sifru predmeta:");
	getchar();
	gets(sifra_predmeta);
	printf("\nUnesite ocenu:");
	getchar();
	scanf("%d", &ocena);

	predmet = vrati_predmet(sifra_predmeta);
	if (predmet == NULL) {
		printf("\nPredmet ne postoji u listi predmeta.");
	}
	else {
		student = vrati_studenta(godina, broj_indeksa);
		if (student == NULL) {
			printf("\nStudent ne postoji u listi studenata.");
		}
		else {
			nova_prijava.indeks.godina = godina;
			nova_prijava.indeks.broj = broj_indeksa;
			strcpy(nova_prijava.sifra_predmeta, sifra_predmeta);
			nova_prijava.ocena = ocena;
			ubaci_prijavu(nova_prijava, xprijave, broj_prijava);
			student->ukupan_ESP = student->ukupan_ESP + predmet->broj_ESP;
		}
	}
}
void prikazi_prijave(xprijave xprijave, int broj_prijava) {
	int i;
	printf("\nPrijave:\n");
	for (i = 0; i < broj_prijava; i++) {
		printf("\n%4d/%4d %10s %6d",  xprijave[i].indeks.godina,
					xprijave[i].indeks.broj,
					xprijave[i].sifra_predmeta,
					xprijave[i].ocena);
	}
	printf("\n");
}
int main(void) {
	xprijave prijave;
	int broj_prijava = 0;
	unesi_prijavu(prijave, &broj_prijava);
	unesi_prijavu(prijave, &broj_prijava);
	unesi_prijavu(prijave, &broj_prijava);
	prikazi_prijave(prijave, broj_prijava);
	return 0;
}
```

5. Дати су низови предмета, студената и пријава. Написати следеће потпрограме:
а)Написати процедуру за приказ свих студената који постоје у датом низу,
б)Написати процедуру за сортирање низа студената према оствареним броју ЕСПБ поена, у опадајућем редоследу.

```
#include <stdio.h>
typedef struct indeks {
	int godina;
```

```
	int broj;
} TINDEKS;
typedef struct student {
	TINDEKS indeks;
	char prezime[20];
	char ime[20];
	int ukupan_ESP;
} TSTUDENT;

typedef TSTUDENT xstudenti[50];
xstudenti studenti = {
		{{2015, 5}, "Peric", "Pera", 15},
		{{2015, 55}, "Petrovic", "Petar", 20},
		{{2015, 555}, "Janic", "Jana", 11},
		{{2015, 3}, "Markovic", "Marko", 36},
		{{2015, 33}, "Andjelkovic", "Ana", 18}
};
int broj_studenata = 5;
void prikazi_studente(void) {
	int i;
	printf("\nStudenti:\n");
	for (i = 0; i < broj_studenata; i++) {
		printf("\n%4d/%4d %10s %10s %6d",  studenti[i].indeks.godina,
						studenti[i].indeks.broj,
						studenti[i].prezime,
						studenti[i].ime,
						studenti[i].ukupan_ESP);
	}
	printf("\n");
}
void napravi_rang_listu(void) {
	int i, j;
	TSTUDENT st;
	for (i = 0; i < broj_studenata - 1; i++) {
		for (j = i + 1; j < broj_studenata; j++) {
			if (studenti[i].ukupan_ESP < studenti[j].ukupan_ESP) {
				st = studenti[i];
				studenti[i] = studenti[j];
				studenti[j] = st;
			}
		}
	}
}
int main(void) {
	prikazi_studente();
	napravi_rang_listu();
	printf("\nRANG LISTA----------\n");
	prikazi_studente();
	return 0;
}
```


1.2. Задаци за вежбање

6. Дата је структура *производ (шифра, назив, цена)*. Написати следеће потпрограме:
а) Написати функцију за унос новог производа у низ производа. Сви производи се разликују по шифри, тако да не могу постојати два производа са истом шифром.
б) Написати функцију која сортира производе по називу.

в) Написати функцију која брише производ из низа производа на основу назива који се прослеђује као улазни аргумент. Уколико у низу постоји више производа са истим називом, потребно их је све обрисати.

2. Листе

У овом одељку приказане су листе. Дати су решени задаци, задаци за вежбање, задаци са колоквијума и задаци са испитних рокова.

2.1. Решени задаци

7. Дата је једноструко спрегнута листа целих бројева. Написати следеће потпрограме:
а) Написати процедуру за убацивање новог елемента на почетак листе,
б) Написати функцију за избацивање елемента са почетка листе,
в) Написати процедуру за приказ елемената листе.

```
#include <stdio.h>
#include <stdlib.h>
typedef struct cvor CVOR;
typedef CVOR* PCVOR;
struct cvor {
	int info;
	PCVOR sledeci;
};
void ubaci_na_pocetak(PCVOR* glava, int broj) {
	PCVOR novi;
	novi = malloc(sizeof(CVOR));
	novi->info = broj;
	novi->sledeci = *glava;
	*glava = novi;
}
int izbaci_sa_pocetka(PCVOR* glava) {
	PCVOR cvor_za_brisanje;
	int broj; //broj koji ce se obrisati
	if (*glava != NULL) {
		cvor_za_brisanje = *glava;
		*glava = (*glava)->sledeci;
		broj = cvor_za_brisanje->info;
		free(cvor_za_brisanje);
		return broj;
	}
	return -1;
}
void prikazi_listu(PCVOR glava) {
	PCVOR tekuci;
	tekuci = glava;
	printf("\n*** Sadrzaj liste ***\n");
	while (tekuci != NULL) {
		printf("%d\t", tekuci->info);
		tekuci = tekuci->sledeci;
	}
}
int main(void) {
	PCVOR glava;
	int izbacen_element;
	glava = NULL;
	ubaci_na_pocetak(&glava, 0);
	ubaci_na_pocetak(&glava, 1);
	ubaci_na_pocetak(&glava, 2);
	prikazi_listu(glava);
	izbaci_sa_pocetka(&glava);
	ubaci_na_pocetak(&glava, 3);
```

```
	ubaci_na_pocetak(&glava, 4);
	prikazi_listu(glava);
	return 0;
}
```

8. Дата је једноструко спрегнута листа целих бројева. Написати следеће потпрограме:
а) Написати процедуру за убацивање новог елемента на крај листе,
б) Написати функцију за избацивање елемента са краја листе,
в) Написати процедуру за приказ елемената листе.

```
#include <stdio.h>
#include <stdlib.h>
typedef struct cvor CVOR;
typedef CVOR* PCVOR;
struct cvor {
	int info;
	PCVOR sledeci;
};
void ubaci_na_kraj(PCVOR* glava, int broj) {
	PCVOR novi;
	PCVOR poslednji;
	novi = malloc(sizeof(CVOR));
	novi->info = broj;
	novi->sledeci = NULL;
	if (*glava == NULL) {
		*glava = novi;
	}
	else {
		poslednji = *glava;
		while (poslednji->sledeci != NULL) {
			poslednji = poslednji->sledeci;
		}
		poslednji->sledeci = novi;
	}
}
int izbaci_sa_kraja(PCVOR* glava) {
	int broj; //broj koji ce se obrisati
	PCVOR brzi, spori;
	brzi = *glava;
	spori = NULL;
	while (brzi->sledeci != NULL) {
		spori = brzi;
		brzi = brzi->sledeci;
	}
	if (spori == NULL) {
		broj = brzi->info;
		free(brzi);
		*glava = NULL;
	}
	else {
		broj = brzi->info;
		free(brzi);
		spori->sledeci = NULL;
	}
	return broj;
}
void prikazi_listu(PCVOR glava) {
	PCVOR tekuci;
	tekuci = glava;
	printf("\n*** Sadrzaj liste ***\n");
```

```
	while (tekuci != NULL) {
		printf("%d\t", tekuci->info);
		tekuci = tekuci->sledeci;
	}
}
int main(void) {
	PCVOR glava;
	glava = NULL;
	ubaci_na_kraj(&glava, 0);
	ubaci_na_kraj(&glava, 1);
	ubaci_na_kraj(&glava, 2);
	prikazi_listu(glava);
	izbaci_sa_kraja(&glava);
	ubaci_na_kraj(&glava, 3);
	ubaci_na_kraj(&glava, 4);
	prikazi_listu(glava);
	return 0;
}
```

9. Дата је једноструко спрегнута листа целих бројева. Написати следеће потпрограме:
а) Написати процедуру за додавање новог елемента на крај листе, а за коју су познати почетак (глава) и крај (реп),
б) Написати процедуру која сортира дату листу у нерастућем редоследу,
в) Написати процедуру која убацује елемент у сортирану листу, тако да листа остане сортирана у нерастућем редоследу.

```
#include <stdio.h>
typedef struct cvor* PCVOR;
typedef struct cvor {
	int info;
	PCVOR sledeci;
} TCVOR;
void ubaci_na_kraj(PCVOR* glava, PCVOR* rep, int a) {
	PCVOR novi;
	novi = malloc(sizeof(TCVOR));

	novi->info = a;
	novi->sledeci = NULL;
	if (*glava == NULL) {
		*glava = novi;
		*rep = novi;
	}
	else {
		(*rep)->sledeci = novi;
		*rep = novi;
	}
}
void sortiraj_listu(PCVOR glava, PCVOR rep) {
	PCVOR spori;
	PCVOR brzi;
	int pomocni;
	spori = glava;
	while (spori != rep) {
		brzi = spori->sledeci;
		while (brzi != NULL) {
			if (spori->info > brzi->info) {
				pomocni = brzi->info;
				brzi->info = spori->info;
				spori->info = pomocni;
			}
```

```
                brzi = brzi->sledeci;
            }
            spori = spori->sledeci;
        }
}
void ubaci_u_sortiranu(PCVOR *glava, PCVOR *rep, int a) {
        PCVOR novi;
        PCVOR brzi;
        PCVOR spori;
        int signal;
        brzi = *glava;
        signal = 0;
        spori = NULL;
        while (brzi != NULL && signal == 0) {
            if (brzi->info > a) signal = 1;
            else {
                spori = brzi;
                brzi = brzi->sledeci;
            }
        }
        if (signal == 0) { // Nije nadjena pozicija
            ubaci_na_kraj(glava, rep, a);
        }
        else { // Pozicija je nadjena
            novi = malloc(sizeof(TCVOR));
            novi->info = a;
            novi->sledeci = NULL;
            if (spori == NULL) {
                novi->sledeci = *glava;
                *glava = novi;
            }
            else {
                novi->sledeci = brzi;
                spori->sledeci = novi;
            }
        }
}
int main(void) {
        PCVOR glava = NULL;
        PCVOR rep = NULL;
        ubaci_na_kraj(&glava, &rep, 5);
        ubaci_na_kraj(&glava, &rep, 10);
        ubaci_na_kraj(&glava, &rep, 2);
        ubaci_na_kraj(&glava, &rep, 8);
        ubaci_na_kraj(&glava, &rep, 1);
        sortiraj_listu(glava, rep);
        ubaci_u_sortiranu(&glava, &rep, 4);
        ubaci_u_sortiranu(&glava, &rep, 0);
        ubaci_u_sortiranu(&glava, &rep, 17);
        return 0;
}
```

10. Дата је једноструко спрегнута листа целих бројева. Написати следеће потпрограме:
а) Написати процедуру за додавање новог елемента на крај листе, а за коју су познати почетак (глава) и крај (реп),
б) Написати функцију која израчунава средњу вредност елемената у листи,
в) Написати функцију која израчунава број елемената чија је вредност већа од задате вредности.

```
#include <stdio.h>
typedef struct cvor* PCVOR;
typedef struct cvor {
	int info;
	PCVOR sledeci;
} TCVOR;
void ubaci_na_kraj(PCVOR* glava, PCVOR* rep, int a) {
	PCVOR novi;
	novi = malloc(sizeof(TCVOR));
	novi->info = a;
	novi->sledeci = NULL;
	if (*glava == NULL) {
		*glava = novi;
		*rep = novi;
	}
	else {
		(*rep)->sledeci = novi;
		*rep = novi;
	}
}
double srednja_vrednost(PCVOR glava) {
	PCVOR tekuci = glava;
	int suma = 0;
	int brojac = 0;
	double srednja_vrednost;
	while (tekuci != NULL) {
		suma = suma + tekuci->info;
		brojac++;
		tekuci = tekuci->sledeci;
	}
	if (brojac == 0) {
		return 0;
	}
	srednja_vrednost = (double)suma / brojac;
	return srednja_vrednost;
}

int broj_elemenata_vecih_od(PCVOR glava, double vr) {
	PCVOR tekuci = glava;
	int brojac = 0;
	while (tekuci != NULL) {
		if (tekuci->info > vr) brojac++;
		tekuci = tekuci->sledeci;
	}
	return brojac;
}

int main(void) {
	double srednja_vrednost_liste;
	int veci_od_sr_vred;
	PCVOR glava = NULL;
	PCVOR rep = NULL;
	ubaci_na_kraj(&glava, &rep, 5);
	ubaci_na_kraj(&glava, &rep, 10);
	ubaci_na_kraj(&glava, &rep, 2);
	ubaci_na_kraj(&glava, &rep, 8);
	ubaci_na_kraj(&glava, &rep, 1);
	srednja_vrednost_liste = srednja_vrednost(glava);
	veci_od_sr_vred = broj_elemenata_vecih_od(glava, srednja_vrednost_liste);
```

```
	printf("\nSrednja vrednost je %.2lf, a broj elemenata veci od SRV je %d",
srednja_vrednost_liste, veci_od_sr_vred);
	return 0;
}
```

11. Дата је листа производа, за коју је позната глава и реп. За сваки производ чувају се шифра, назив и цена. Написати потпрограм који омогућава додавање производа на крај листе. Додавање производа треба омогућити само уколико тај производ већ не постоји у листи.

```
#include <stdlib.h>
#include <stdio.h>
typedef struct proizvod {
	int sifra;
	char naziv[20];
	double cena;
} PROIZVOD;
typedef struct cvor* PCVOR;
typedef struct cvor {
	PROIZVOD info;
	PCVOR sledeci;
} CVOR;
int postoji(PCVOR glava, PROIZVOD proizvod) {
	PCVOR tekuci = glava;
	while (tekuci != NULL) {
		if (tekuci->info.sifra == proizvod.sifra) {
			return 1;
		}
		tekuci = tekuci->sledeci;
	}
	return 0;
}
void ubaci(PCVOR* glava, PCVOR* rep, PROIZVOD proizvod) {
	if (postoji(*glava, proizvod) == 1) {
		printf("Postoji, nece biti ubacen!!!");
		return;
	}
	else {
		PCVOR novi = (PCVOR)malloc(sizeof(CVOR));
		novi->info = proizvod;
		novi->sledeci = NULL;
		if (*glava == NULL) {
			*glava = novi;
		}
		else {
			(*rep)->sledeci = novi;
		}
		*rep = novi;
	}
}
int main(void) {
	PCVOR glava, rep;
	glava = NULL;
	rep = NULL;
	PROIZVOD proizvod1 = {1, "Proizvod1", 200};
	PROIZVOD proizvod2 = {2, "Proizvod2", 300};
	ubaci(&glava, &rep, proizvod1);
	ubaci(&glava, &rep, proizvod1);
	ubaci(&glava, &rep, proizvod2);
```

```
	return 0;
}
```

12. Дата је листа производа. Написати потпрограм који приказује листу производа на стандардном излазу.

```
#include <stdlib.h>
#include <stdio.h>
typedef struct proizvod {
	int sifra;
	char naziv[20];
	double cena;
} PROIZVOD;
typedef struct cvor* PCVOR;
typedef struct cvor {
	PROIZVOD info;
	PCVOR sledeci;
} CVOR;
void prikazi_listu(PCVOR glava) {
	PCVOR tekuci = glava;
	while (tekuci != NULL) {
		prikazi_proizvod(tekuci->info);
		tekuci = tekuci->sledeci;
	}
}
void prikazi_proizvod(PROIZVOD proizvod) {
	printf("\n%5d %30s %5lf", proizvod.sifra, proizvod.naziv, proizvod.cena);
}
int main(void) {
	PCVOR glava;
	glava = NULL;
	prikazi_listu(glava);
	return 0;
}
```

2.2. Задаци за вежбање

13. Имплементирати *LIFO* (енгл. *Last In First Out*) листу у којој се чувају цели бројеви.
14. Имплементирати *FIFO* (енгл. *First In First Out*) листу у којој се чувају цели бројеви.
15. Имплементирати функцију која проверава да ли задати број постоји у листи.
16. Имплементирати функцију која проверава колико елемената листе има вредност већу од аритметичке средине парних елемената листе.
17. Имплементирати функцију која избацује задати елемент из листе.
18. Имплементирати функцију која проверава да ли су елементи листе сортирани у неопадајућем редоследу.
19. Имплементирати функцију која проверава да ли су елементи листе сортирани у нерастућем редоследу.
20. Имплементирати функцију која проверава да ли су елементи листе сортирани у опадајућем редоследу.
21. Имплементирати функцију која проверава да ли су елементи листе сортирани у растућем редоследу.
22. Имплементирати функцију која приказује фреквенцију појављивања сваког елемента листе.
23. Дат је низ целих бројева у коме елементи могу да се понављају. Пребацити све елементе из низа у једноструко спрегнуту листу, тако да сви елементи у листи буду међусобно различити. Приказати тако добијену једноструко спрегнуту листу.

24. Дата је матрица целих бројева. Све елементе задате колоне матрице пребацити у једноструко спрегнуту листу. Приказати садржај једноструко спрегнуте листе.
25. Дата је квадратна матрица димензије 4x4. Имплементирати функцију која све парне елементе испод споредне дијагонале матрице пребацује у *LIFO* листу.
26. Дате су две листе у којој елементи могу да се понављају. Формирати нову листу од елемената који се налазе и у једно и у другој листи. У новоформираној листи сви елементи морају бити међусобно различити.
27. Дате су следеће структуре:
    – Структура *Студент (број индекса, име, презиме, укупан број ЕСПБ)*
    – Структура *Предмет (шифра предмета, назив предмета, број ЕСПБ)*
    – Структура *Пријава (број индекса, шифра предмета, оцена, рок)*
    Дат је низ студената и низ предмета. У низу студената сви студенти се разликују по броју индекса, док се у низу предмета сви предмети разликују по шифри предмета. Креирати листу пријава и имплементирати следеће фукције:
    а) Унеси нову пријаву. Пријава може да се унесе за студента који постоји у низу студената и предмет који постоји у низу предмета. За једног студента и један предмет у листи може да се налази само један слог чија је оцена већа од 6 (може постојати више пријава са оценом 5 за једног студента и један предмет). Приликом уноса нове пријаве ажурирати укупан број ЕСПБ за студента који је положио испит са пријаве за онолико ЕСПБ колико ЕСП тај предмет.
    б) Сортирај низ предмета (сортирање извршите према називу предета, абецедно у растућем редоследу).
    в) Креирај ранг листу (сортирану у опадајућем редоследу по броју ЕСПБ) која садржи редни број, број индекса, име и презиме студената и укупан број ЕСПБ. Ранг листа се формира по број остварених бодова по следећој формули: УКУПАН БРОЈ БОДОВА=УКУПАНБРОЈ ЕСПБ * 10 + ПРОСЕЧНА ОЦЕНА * 5.
28. Дата је структура *Студент (број индекса, име, презиме, смер)*. Смер може да узима вредности ИСиТ и МЕН. Написати следеће функције:
    а) Приказати све студенте задатог смера.
    б) Сортирати све студенте према смеру, а затим у оквиру смера према презимену у нерастућем редоследу.
    в) Приказати укупан број студента смера ИСиТ и МЕН.

2.3. Задаци са колоквијума

29. Дата је једноструко-спрегнута листа целих бројева (ЛИСТА 1). Имплементирати следеће функције:
    а) Имплементирати операцију убацивања новог елемента на почетак листе.
    б) Пребацити све елементе једноструко спрегнуте листе целих бројева (ЛИСТА 1) у нову листу, тако да нова листа садржи све елементе листе (ЛИСТА 1) али без понављања и информацију о броју појављивања сваког елемента једноструко спрегнуте листе (ЛИСТА 1).
    в) Приказати онај елемент листе (ЛИСТА 1) који има највећи број појављивања, а уколико их има више приказати све елементе.
30. Дат је низ студената:

```
STUDENT xs[] = {
        { "2016_0001", "Andrej Andric" },
        { "2016_0002", "Sofija_Sofic" },
        { "2016_0003", "Jovan Jovic" },
        { "2016_0004", "Sara Saric" },
        { "2016_0005", "Milica Milic" }
    };
```

    Дат је низ предмета:

```
PREDMET xp [] = {
        { 1,"Programiranje 1" },
        { 2,"Programiranje 2" },
        { 3,"UIS" },
        { 4,"Matematika" }
    };
```

Имплементирати следеће потпрограме:

а) Функцију за унос нове пријаве у једностуко спрегнуту листу. За сваку пријаву чувају се следеће информације: *број индекса студента*, *шифра предмета* и *оцена*. Унос пријаве је могућ само ако су испуњени следећи предуслови:

– Студент за кога се уноси пријава постоји регистрован у низу студената

– Предмет за који се уноси пријава постоји регистрован у низу предмета

– Студент за који се уноси пријава није положио тај предмет

– Оцена може да узме вредности од 5 до 10

– Један студент за један предмет може да има само једну позитиву пријаву (пријаву са оценом већом од 5, али може да има више пријава са оценом 5)

б) Процедуру која на стандардном излазу приказује извештај о ранг листи студената на основу просечне оцене студената у формату који је дат ниже. Уколико два студента имају исту просечну оцену они се рангирају се по броју положених испита.

```
-------------------------------------------------------------------------------
Rang lista studenata na osnovu prosecne ocene
-------------------------------------------------------------------------------
Rb.    Broj indeksa Student             Prosecna ocena          Broj polozenih ispita
-------------------------------------------------------------------------------
1.     2016_0005    Milica Milic        9.5                     2
2.     2016_0004    Sara Saric          9.5                     1
3.     2016_0003    Jovan Jovic         9                       3
4.     2016_0003    Sofija_Sofic        7                       2
5.     2016_0001    Andrej Andric       7                       1
```

в) Функцију која омогућава кориснику да преко одговарајућег корисничког менија позове претходно имплементиране функције из овог задатка:

```
Korisnicki meni:
1) Unesi novu prijavu
2) Formiraj rang listu
3) Kraj programa
Vas izbor:__
```

31. Дат је низ студената:

```
STUDENT xs[] = {
        { "2016_0001", "Andrej Andric" },
        { "2016_0002", "Sofija_Sofic" },
        { "2016_0003", "Jovan Jovic" },
        { "2016_0004", "Sara Saric" },
        { "2016_0005", "Milica Milic" }
    };
```

Дат је низ предмета:

```
PREDMET xp [] = {
        { 1,"Programiranje 1" },
        { 2,"Programiranje 2" },
        { 3,"UIS" },
        { 4,"Matematika" }
    };
```

Имплементирати следеће потпрограме:

а) Функцију за унос нове пријаве у једностуко спрегнуту листу. **Пријаве се уносе на почетак листе.** За сваку пријаву чувају се следеће информације: *број индекса студента*, *шифра предмета* и *оцена*. Унос пријаве је могућ само ако су испуњени следећи предуслови:

– Студент за кога се уноси пријава постоји регистрован у низу студената
– Предмет за који се уноси пријава постоји регистрован у низу предмета
– Студент за који се уноси пријава није положио тај предмет
– Оцена може да узме вредности од 5 до 10
– Један студент за један предмет може да има само једну позитиву пријаву (пријаву са оценом већом од 5, али може да има више пријава са оценом 5)
б) Процедуру која на стандардном излазу приказује извештај о просечној оцени за сваки предмет у формату који је дат ниже. **Извештај је сортиран на основу просечне оцене за предмет.** Уколико је просечна оцена за два или више предмета иста сортирање се врши према броју пријава са оценом већом од 5.

```
-------------------------------------------------------------------------------
Izvestaj za predmete
-------------------------------------------------------------------------------
Rb. Sifra    Predmet            Prosek       Br.prij. (>5)       Br.prij.(=5)
-------------------------------------------------------------------------------
1.     1     UIS 1              9            3                   3
2.     2     Programiranje 2    9            2                   2
3.     3     Programiranje 1    8            2                   3
4.     4     Matematika         8            1                   4
```

в) Функцију која омогућава кориснику да преко одговарајућег корисничког менија позове претходно имплементиране функције из овог задатка:

```
Korisnicki meni:
1) Unesi novu prijavu
2) Formiraj izvestaj
3) Kraj programa
Vas izbor:__
```

2.4. Задаци са испитних рокова

32. Имплементирати следеће потпрограме:
а) Имплементирати функцију која рачуна збир цифара неког задатог броја.
б) Написати главни програм који са стандардног улаза прихвата n бројева и приказује број чији је збир цифара највећи. Искористити претходно имплементирану функцију из задатака а).
в) Написати функцију која рачуна колико има парних троцифрених бројева чији је збир цифара 13. Искористити претходно имплементирану функцију из задатака а). У главном програму позвати имплементирану функцију.
г) Написати функцију која све троцифрене парне бројеве чији је збир цифара једнак неком задатом броју пребацује у једноструко спрегнуту листу тако да листа буде сортирана у опадајућем редоследу (није дозвољено сортирање листе након уноса елемената у листу). У главном програму приказати садржај листе.
33. Имплементирати следеће потпрограме:
а) Написати потпрограм која за неки задати број испитује да ли је палиндром. Тест пример: 12321 (јесте), 2343 (није).
б) Написати потпрограм и главни програм који са стандардног улаза прихвата бројеве, а затим бројеве који су палиндроми уписује у једноструко спегнуту листу. Приказати садржај листе.
в) Написати потпрограм и главни програм који приказује елементе једноструко спрегнуте листе који се налазе између задатих позиција.
Тест пример:
Листа: 1,2,3,4,5,6,7,8
Задате поз: 3,5 Листа 3,4,5
Задате поз: 0,5 Листа – indeks out of bounds [1–8]
Задате поз: 2,11 Листа – indeks out of bounds [1–8]

г) Написати потпрограм и главни програм који у једноструко спрегнутој листи одређује почетак и крај најдуже серије узастопних једнаких елемената листе. Приказати ову серију бројева. Искористити потпрограм из претходног задатка за приказ серије бројева.

34. У једноструко спрегнутој листи се налазе позитивни цели бројеви. Написати програм који ће пресложити бројеве у листи, тако да се прво у листи нађу прости бројеви, а након тога сложени бројеви. Сачувати редослед бројева као у улазној листи.
    Улазна листа:
    8, 9, 1, 6, 5, 11, 19, 22,23
    Излазна листа:
    1,5,11,19,23,8,9,6,22
35. Дате су две једноструко спрегнуте листе L1 и L2 са истим бројем елемената. Написати функцију која формира нову листу L која садржи алтернирајући распоређене елементе листе L1 и L2 (први ел. из L1, први ел. из L2, други ел. из L1, други ел. из L2). Приликом формирања нове листе L не формирати нове чворове, већ само постојеће чворове распоредити у једну листу.
36. Дата је квадратна матрица *M* димензије 4X4. Имплементирати следеће потпрограме:
    а) Имплементирати процедуру која ће од улазне матрице креирати листу целих бројева тако да сви елементи у листи буду међусобно различити.
    б) Имплементирати процедуру која приказује колико се пута сваки елементи листе *L* појављује на споредној дијагонали матрице *M*.
    в) У главном програму позвати имплементиране функције и приказати резултат у формату који је дат ниже.

```
Broj ___ iz liste se na sporednoj dijagonali pojavljuje ___ puta
Broj ___ iz liste se na sporednoj dijagonali pojavljuje ___ puta
Broj ___ iz liste se na sporednoj dijagonali pojavljuje ___ puta
```

37. Дата је квадратна матрица. Написати потпрограм који рачуна унију елемената две задате колоне матрице и елементе уније смешта у једноструко спрегнуту листу при чему није дозвољено понављање елемената у листи (сви елементи у листи су међусобно различити).
38. Дата је квадратна матрица. Написати потпрограм који рачуна унију елемената два задата реда матрице и елементе уније смешта у једноструко спрегнуту листу при чему није дозвољено понављање елемената у листи (сви елементи у листи су међусобно различити).
39. Дата је квадратна матрица. Написати потпрограм који рачуна суму елемената по колонама матрице и све елементе колоне чија је сума највећа пребацује у листу. Приказати садржај листе.
40. Дата је квадратна матрица. Написати потпрограм који рачуна суму елемената по редовима матрице и све елементе реда чија је сума највећа пребацује у листу. Приказати садржај листе.
41. Дат је низ студената и низ предмета. За сваког студента се чувају следеће информације: *број индекса*, *име и презиме* и *број предмета који је положио тај студент*, док се за сваки предмет чувају информације: *шифра предмета*, *назив предмета* и *број студената који су положили тај предмет*. Имплементирати следеће потпрограме:
    а) Функцију за унос нове пријаве у једностуко спрегнуту листу која садржи показивач на **први елемент** листе. **Пријаве се уносе на крај листе**. За сваку пријаву чувају се следеће информације: *број индекса студента*, *шифра предмета* и *оцена*. Унос пријаве је могућ само ако су испуњени следећи предуслови:
    – Студент за кога се уноси пријава постоји регистрован у низу студената.
    – Предмет за који се уноси пријава постоји регистрован у низу предмета.
    – Студент за који се уноси пријава није положио тај предмет.
    – Оцена може да узме вредности од 5 до 10.
    – Један студент за један предмет може да има само једну позитиву пријаву (пријаву са оценом већом од 5, али може да има више пријава са оценом 5).

– При сваком уносу позитивне пријаве ажурирати број студената који су положили предмет и број предмета који је студент са пријаве положио.
б) Процедуру која на стандардном излазу креира **извештај о предметима који је сортиран по броју студената који су положили тај предмет у опадајућем редоследу** у формату који је дат ниже.

```
--------------------------------------------------------------------------------
Izvestaj za predmete
--------------------------------------------------------------------------------
Rb. Sifra    Predmet            Br.prij.
--------------------------------------------------------------------------------
1.     1     UIS 1              4
2.     2     Programiranje 2    3
3.     3     Programiranje 1    3
4.     4     Matematika         1
```

в) Функцију која омогућава кориснику да преко одговарајућег корисничког менија позове претходно имплементиране функције из овог задатка. Овај део задатка се оцењује само ако је урађен задатак под а) или б).

```
Korisnicki meni:
1) Unesi novu prijavu
2) Formiraj izvestaj
3) Kraj programa
Vas izbor:__
```

42. Дат је низ студената и низ предмета. За сваког студента се чувају следеће информације: *број индекса*, *име и презиме* и *број предмета који је положио тај студент*, док се за сваки предмет чувају информације: *шифра предмета*, *назив предмета* и *број студената који су положили тај предмет*. Имплементирати следеће потпрограме:
а) Функцију за унос нове пријаве у једностуко спрегнуту листу која садржи показивач на **први елемент** листе. **Пријаве се уносе на крај листе**. За сваку пријаву чувају се следеће информације: *број индекса студента*, *шифра предмета* и *оцена*. Унос пријаве је могућ само ако су испуњени следећи предуслови:
– Студент за кога се уноси пријава постоји регистрован у низу студената.
– Предмет за који се уноси пријава постоји регистрован у низу предмета.
– Студент за који се уноси пријава није положио тај предмет.
– Оцена може да узме вредности од 5 до 10.
– Један студент за један предмет може да има само једну позитиву пријаву (пријаву са оценом већом од 5, али може да има више пријава са оценом 5).
– При сваком уносу позитивне пријаве ажурирати број студената који су положили предмет и број предмета који је студент са пријаве положио.
б) Процедуру која на стандардном излазу креира **извештај о студентима који је сортиран по броју предмета које је студент положио у опадајућем редоследу** у формату који је дат ниже.

```
--------------------------------------------------------------------------------
Izvestaj za studente
--------------------------------------------------------------------------------
Rb. Br.ind   Student            Br.pred.
--------------------------------------------------------------------------------
1. 2016/0001 Student1           4
2. 2016/0001 Student2           4
3. 2016/0001 Student3           3
```

в) Функцију која омогућава кориснику да преко одговарајућег корисничког менија позове претходно имплементиране функције из овог задатка. Овај део задатка се оцењује само ако је урађен задатак под а) или б).

```
Korisnicki meni:
1) Unesi novu prijavu
2) Formiraj izvestaj
```

```
3) Kraj programa
Vas izbor:__
```

43. Дат је низ репрезентација. За сваку репрезентацију се чувају следеће информације: *шифра*, *назив*, *број одиграних утакмица*, *број победа*, *број нерешених*, *број пораза* и *број поена*. Имплементирати следеће потпрограме:
а) Потпрограм за унос нове утакмице у једноструко спрегнуту листу која садржи **показивач на први елемент** листе. **Утакмице се уносе на крај листе**. За сваку утакмицу чувају се следеће информације: *репрезентација домаћин*, *репрезентација гост*, *број голова домаћин* и *број голова гост*. Унос утакмице је могућ само ако су испуњени следећи предуслови:
– Домаћин и гост на утакмици постоје регистровани у низу репрезентација,
– Домаћин и гост морају бити различити,
– Пар *домаћин-гост* или *гост-домаћин* не постоје у листи утакмица,
– Број голова (домаћина/госта) мора бити већи или једнак нули,
– При сваком уносу утакмице ажурирати број одиграних утакмица, број победа, број нерешених, број пораза и број поена репрезентација из утакмице. За сваку победу репрезентација добија три бода а за нерешен резултат један бод.
б) Потпрограм који на стандардном излазу формира извештај о учинку репрезентација који је уређен према броју остварених бодова у опадајућем редоследу у формату који је дат ниже.

```
-------------------------------------------------------------------------------
Izvestaj o ucinku reprezentacija
-------------------------------------------------------------------------------
Rang. Sifra        Naziv  Br. bodova
-------------------------------------------------------------------------------
1.     1           R1     6
2.     4           R3     4
3.     3           R2     1
```

в) Потпрограм који омогућава кориснику да преко одговарајућег корисничког менија позове претходно имплементиране функције из овог задатка (корисник може више пута да позове функције менија, све док се не одабере опција крај програма).

```
Korisnicki meni:
1) Unesi nove utakmice
2) Formiraj izvestaj
3) Kraj programa
Vas izbor:__
```

44. Креирати једноструко спрегнуту листу у којој се чувају подаци о студентским пријавама (број индекса, шифра предмета и оцена). Број индекса је стринг максималне дужине 9 карактера, шифра предмета је стринг максималне дужине 7 карактера, а оцена је цео број у интервалу од 5 до 10. Имплементирати следеће потпрограме:
а) Имплементирати функцију која додаје нову пријаву у листу, а тако да она остане уређена на следећи начин:
– убацивање нове пријаве се врши увек на крај,
– у листи се не могу наћи две идентичне пријаве (пријаве су идентичне уколико имају исти број индекса, шифру предмета и оцену)
– за један број индекса и један предмет може постојати више пријава, али тако да последња унета пријава мора да садржи највећу оцену. На пример у листи се могу наћи пријаве: {(1/2012, pred-1,6),(1/2012, pred-1,8),(2/2012, pred-2,6),(1/2012, pred-1,9)} је за број индекса 1/2012 и предмет са шифром pred-1 оцене су редом 6, 8, 9 (уређене у растућем редоследу, последња мора бити увек већа од свих претходних).
б) Имплементирати функцију која приказује садржај једноструко спрегнуте листе.

45. Креирати једноструко спрегнуту листу у којој се чувају подаци о студентским пријавама (број индекса, шифра предмета и оцена). Број индекса је стринг максималне дужине 9 карактера, шифра предмета је стринг максималне дужине 7 карактера, а оцена је цео број у интервалу од

5 до 10. Листа може да садржи две или више пријаве једног студента (број индекса) за један предмет (шифра предмета).
Креирати нову једноструко спрегнуту листу која садржи податке о броју положених испита и просечној оцени за сваког студента. Приликом рачунања просека и броја положених испита у обзир се узима последња пријава (уколико је оцена 5, студент није положио испит).
Садржај тако формиране једноструко спрегнуте листе уписати у текстуалну датотеку у следећем формату:

```
Број индекса            Број положених испита          Просечна оцена
15/2012                          4                            8.05
1/2012                           12                           8.16
```

3. Датотеке

У овом одељку приказане су датотеке. Дати су решени задаци, задаци за вежбање, задаци са колоквијума и задаци са испитних рокова.

3.1. Решени задаци

46. Написати програм који чита карактер по карактер из датотеке *podaci.txt* и исписује сваки прочитани карактер на стандардном излазу.

```
#include <stdio.h>
int main(void) {
	FILE * datoteka;
	char znak;
	datoteka = fopen("podaci.txt", "r");
	if (datoteka == NULL) {
		printf("\nDatoteka ne postoji");
	}
	else {
		printf("\nDatoteka postoji\nSadrzaj datoteke:\n");
		while ((znak = fgetc(datoteka)) != EOF) {
			printf("%c", znak);
		}
		fclose(datoteka);
	}
	return 0;
}
```

47. Написати програм који чита линију по линију садржаја датотеке *podaci.txt* и исписује сваки прочитани ред на стандардном излазу.

```
#include <stdio.h>
int main(void) {
	FILE * datoteka;
	char s[1000];
	datoteka = fopen("podaci.txt", "r");
	if (datoteka == NULL) {
		printf("\nDatoteka ne postoji");
	}
	else {
		printf("\nDatoteka postoji\nSadrzaj datoteke:\n");
		while (fgets(s, 1000, datoteka) != NULL) {
			printf("%s", s);
		}
		fclose(datoteka);
	}
	return 0;
}
```

48. Написати програм који чита податке из датотеке *podaci.txt* која у сваком реду садржи два цела броја, раздвојена знаком размака и приказује их.

Пример садржаја датотеке:

```
2	5
9	8
13	1
```

```
#include <stdio.h>
int main(void) {
	FILE * datoteka;
	char s[1000];
	int broj1, broj2;
```

```
	datoteka = fopen("podaci.txt", "r");
	if (datoteka == NULL) {
		printf("\nDatoteka ne postoji");
	}
	else {
		printf("\nDatoteka postoji\nSadrzaj datoteke:\n");

		while (fscanf(datoteka, "%d %d\n", &broj1, &broj2) == 2) {
			printf("\n%d %d", broj1, broj2);
		}
		fclose(datoteka);
	}
	return 0;
}
```

49. Написати програм који чита податке из датотеке *podaci.txt* која у првом реду садржи име и презиме студента, док се у другом реду налази број индекса у формату: година уписа (цео број) / број (цео број).
Пример садржаја датотеке:
Pera Perić
18/231

```
#include <stdio.h>
int main(void) {
	FILE * datoteka;
	char s[1000];
	int godina, broj;
	datoteka = fopen("podaci.txt", "r");
	if (datoteka == NULL) {
		printf("\nDatoteka ne postoji");
	}
	else {
		printf("\nDatoteka postoji\nSadrzaj datoteke:\n");
		while (fgets(s, 1000, datoteka) != NULL) {
			fscanf(datoteka, "%d/%d\n", &godina, &broj);
			printf("\n%s %d/%d", s, godina, broj);
		}
		fclose(datoteka);
	}
	return 0;
}
```

50. Написати програм који чита податке из текстуалне датотеке *podaci.txt*. Датотека садржи податке о студентима. Подаци су форматирани тако да се у првом реду налази број индекса (година уписа / број).
Пример садржаја датотеке:
18/231
Pera Perić

```
#include <stdio.h>
int main(void) {
	FILE * datoteka;
	char s[1000];
	int godina, broj;
	datoteka = fopen("podaci.txt", "r");
	if (datoteka == NULL) {
		printf("\nDatoteka ne postoji");
	}
```

```
	else {
		printf("\nDatoteka postoji\nSadrzaj datoteke:\n");
		while (fscanf(datoteka, "%d/%d\n", &godina, &broj) == 2) {
			fgets(s, 1000, datoteka);
			printf("%d/%d\t%s", godina, broj, s);
		}
		fclose(datoteka);
	}	return 0;
}
```

51. Написати процедуру која омогућава кориснику унос текста (линију по линију) у текстуалну датотеку. Процедура прихвата име датотеке у којој треба сачувати унети текст.

```
#include <stdio.h>
void upisi_u_fajl(char* ime_datoteke) {
	FILE* datoteka;
	int n;
	char str[100];
	datoteka = fopen(ime_datoteke, "w");
	if (datoteka == NULL) {
		printf("Datoteka ne postoji.\n");
	}
	else {
		printf("Koliko linija zelis da uneses ");
		scanf("%d", &n);
		getchar();
		for (int i = 1; i <= n; i++) {
			printf("%d: ", i);
			gets(str);
			fputs(str, datoteka);
			fputs("\n", datoteka);
		}
		fclose(datoteka);
	}
}
int main(void) {
	upisi_u_fajl("ulaz.txt");
	return 0;
}
```

52. Написати процедуру која пребацује садржај једне текстуалне датотеке у другу, тако да се у излазној датотеци налази текст из улазне датотеке, али без знакова размака. Процедура треба да прихвата називе улазне и излазне датотеке.

Пример:
Улазна датотека: Ово је неки текст.
Излазна датотека: Овојенекитекст.

```
#include <stdio.h>
void prebacivanje(char* ulazna_datoteka, char* izlazna_datoteka) {
	FILE* datoteka_ulaz;
	FILE* datoteka_izlaz;
	datoteka_ulaz = fopen(ulazna_datoteka, "r");
	datoteka_izlaz = fopen(izlazna_datoteka, "w");
	if (datoteka_ulaz == NULL || datoteka_izlaz == NULL) {
		printf("\nGreška!");
	}
	else {
		char ch;
		while ((ch = fgetc(datoteka_ulaz)) != EOF) {
			if (ch != ' ') {
				fputc(ch, datoteka_izlaz);
```

```
                }
            }
            fclose(datoteka_izlaz);
        }
        fclose(datoteka_ulaz);
    }
    int main(void) {
        prebacivanje("ulaz.txt", "izlaz.txt");
        return 0;
    }
```

53. Написати процедуру која пребацује садржај једне текстуалне датотеке у другу, тако да се у излазној датотеци налази текст из улазне датотеке, али тако да се знакови размака замењују знаком тарабе. Уколико између два карактера постоји један или више знакова размака треба их заменити једним знаком тарабе (#) у излазној датотеци. Процедура треба да прихвата називе улазне и излазне датотеке.
    Пример:
    Улазна датотека: Ово је неки текст.
    Излазна датотека: Ово#је#неки#текст.

```
#include <stdio.h>
void prebacivanje_taraba(char* ulazna_datoteka, char* izlazna_datoteka) {
    FILE* datoteka_ulaz;
    FILE* datoteka_izlaz;
    datoteka_ulaz = fopen(ulazna_datoteka, "r");
    datoteka_izlaz = fopen(izlazna_datoteka, "w");
    if (datoteka_ulaz == NULL || datoteka_izlaz == NULL) {
        printf("\nGreška!");
    }
    else {
        char ch, taraba;
        taraba = '#';
        while ((ch = fgetc(datoteka_ulaz)) != EOF) {
            if (ch != ' ') {
                fputc(ch, datoteka_izlaz);
                taraba = '#';
            }
            else {
                if (taraba == '#') {
                    fputc(taraba, datoteka_izlaz);
                    taraba = ' ';
                }
            }
        }
        fclose(datoteka_izlaz);
        fclose(datoteka_ulaz);
    }
}
int main(void) {
    prebacivanje_taraba("ulaz.txt", "izlaz.txt");
    return 0;
}
```

54. Написати функцију која од улазног низа карактера креира излазни низ који садржи само цифре. Излатни низ садржи цифре које се налазе у улазном низу, при чему редослед њихових појављивања мора да буде исти у оба низа.
    Пример:
    Улазни низ: 5sad5102 21asd 049a
    Излазни низ: 5510221049

```
#include <stdio.h>
int da_li_je_cifra(char ch) {
	if ((ch - '0' >= 0) && (ch - '0' <= 9)) return 1;
	return 0;
}
char* formiraj_string_cifara(char* str_ulaz) {
	char* str_izlaz;
	int brel = strlen(str_ulaz);
	int n = 0;
	str_izlaz = (char*)malloc((brel + 1) * sizeof(char));
	for (int i = 0; i < brel; i++) {
		if (da_li_je_cifra(str_ulaz[i]) == 1) {
			str_izlaz[n] = str_ulaz[i];
			n++;
		}
	}
	str_izlaz[n] = '\0';
	return str_izlaz;
}
int main(void) {
	char* izlaz = formiraj_string_cifara("5sad5102 21asd 049a");
	printf("%s", izlaz);
	return 0;
}
```

55. Написати програм који омогућава кориснику да унесе податке о студенту (име, презиме, број индекса) и уписује их у текстуалну датотеку *podaci.txt*.

```
#include <stdio.h>
int main(void) {
	FILE * datoteka;
	char ime[15], prezime[15], broj[15];
	datoteka = fopen("podaci.txt", "w");
	if (datoteka == NULL) {
		printf("\nDatoteka ne postoji");
	}
	else {
		printf("\nUnesite ime");
		gets(ime);

		printf("\nUnesite prezime");
		gets(prezime);

		printf("\nUnesite broj indeksa");
		gets(broj);

		fprintf(datoteka, "%s\t%s\n%s\n", ime, prezime, broj);
		fclose(datoteka);
	}
	return 0;
}
```

56. Написати програм који чита целе бројеве из бинарне датотеке *podaci.dat* и исписује их на стандардном излазу.

```
#include <stdio.h>
int main(void) {
	FILE * datoteka;
	int br;
	datoteka = fopen("podaci.dat", "rb");
	if (datoteka == NULL) {
		printf("\nDatoteka ne postoji");
	}
```

```
	else {
		printf("\nSadrzaj datoteke:\n");
		while (fread(&br, sizeof(int), 1, datoteka) != 0) {
			printf("\n%d", br);
		}
		fclose(datoteka);
	}
	return 0;
}
```

57. Дат је низ студената. О сваком студенту чувају се следећи подаци: име и презиме, број положених испита, просек. Написати програм који чита податке о студентима из текстуалне датотеке *podaci.txt*. Пријава садржи следеће податке: име и презиме студента, предмет и оцену. За сваку пријаву треба проверити да ли студент са пријаве постоји у датом низу студената. Приликом поређења студената треба користити име и презиме студента.
Пример садржаја датотеке:
Zika Zikic
Matematika
6
Marko Markovic
Matematika
8
Jana Janic
Programiranje
6

```
#include <stdio.h>
#include <string.h>
typedef struct student {
	char ime_prezime[50];
	int brojPolozenih;
	double prosek;
} TSTUDENT;
typedef TSTUDENT xstudenti[10];
int vrati_studenta(xstudenti x, int nx, char ime[50]) {
	int i;
	for (i = 0; i < nx; i++) {
		if (strcmp(x[i].ime_prezime, ime) == 0) //0 - stringovi jednaki
			return i;
	}
	return -1;
}
int main(void) {
	xstudenti x;
	int nx = 0;
	FILE * datoteka;
	char ime_prezime[50], predmet[50];
	int ocena, i;
	datoteka = fopen("podaci.txt", "r");
	if (datoteka == NULL) {
		printf("Datoteka ne postoji!");
	}
	else {
		while (fgets(ime_prezime, 50, datoteka) != NULL) {
			fgets(predmet, 50, datoteka);
			fscanf(datoteka, "%d\n", &ocena);
			i = vrati_studenta(x, nx, ime_prezime);
			if (i == -1) {
```

```
                        printf("\n%s ne postoji jos u nizu", ime_prezime);
                }
                else {
                        printf("\n%s postoji jos u nizu", ime_prezime);
                }
        }
        fclose(datoteka);
    }
    return 0;
}
```

58. Написати програм који омогућава кориснику унос n целих бројева у бинарну датотеку *brojevi.dat*.

```
#include <stdio.h>
int main(void) {
    FILE * datoteka;
    int i, br, n;
    datoteka = fopen("brojevi.dat", "wb");

    if (datoteka == NULL) {
        printf("\nDatoteka ne postoji");
    }
    else {
        printf("\nUnesite n:");
        scanf("%d", &n);
        for (i = 0; i < n; i++) {
            printf("\nUnesite broj:");
            scanf("%d", &br);
            fwrite(&br, sizeof(int), 1, datoteka);
        }
        fclose(datoteka);
    }
    return 0;
}
```

59. Написати програм који чита целе бројеве из бинарне датотеке *brojevi.dat* и исписује њихове вредности на стандардном излазу. Задатак решити коришћењем *while-do* наредбе.

```
#include <stdio.h>
int main(void) {
    int broj;
    FILE* datoteka;
    datoteka = fopen("brojevi.dat", "rb");
    if (datoteka == NULL) {
        printf("\nDatoteka ne postoji");
    }
    else {
        while (fread(&broj, sizeof(int), 1, datoteka) != 0) {
            printf("%4d", broj);
        }
        fclose(datoteka);
    }
    return 0;
}
```

60. Написати програм који чита целе бројеве из бинарне датотеке *brojevi.dat* и исписује њихове вредности на стандардном излазу. Задатак решити коришћењем *for* наредбе.

```
#include <stdio.h>
int main(void) {
    FILE* dat = fopen("brojevi.dat", "rb");
    if (dat != NULL) {
        fseek(dat, 0, SEEK_END);
```

```
		int x = ftell(dat) / sizeof(int);
		printf("brel=%d\n", x);
		fseek(dat, 0, SEEK_SET);
		for (int i = 0; i < x; i++) {
			int br;
			fread(&br, sizeof(int), 1, dat);
			printf("%d ", br);
		}
		fclose(dat);
	}
	else {
		printf("\ngreska");
	}
	return 0;
}
```

61. Написати програм који, у обрнутом редоследу, чита целе бројеве из бинарне датотеке *brojevi.dat* и исписује њихове вредности на стандардном излазу. Бројеви треба да буду прочитани и исписани од последњег ка првом.

```
#include <stdio.h>
int main(void) {
	FILE* dat = fopen("brojevi.dat", "rb");
	if (dat != NULL) {
		fseek(dat, 0, SEEK_END);
		int x = ftell(dat) / sizeof(int);
		printf("brel=%d\n", x);
		for (int i = 1; i <= x; i++) {
			int a = -i;
			fseek(dat, a * sizeof(int), SEEK_END);
			int br;
			fread(&br, sizeof(int), 1, dat);
			printf("%d ", br);
		}
		fclose(dat);
	}
	else {
		printf("\ngreska");
	}
	return 0;
}
```

62. Написати програм који, коришћењем *for* петље, чита целе бројеве из бинарне датотеке *brojevi.dat* и исписује њихове вредности на стандардном излазу.

```
#include <stdio.h>
int main(void) {
	FILE* dat = fopen("brojevi.dat", "rb");
	if (dat != NULL) {
		fseek(dat, 0, SEEK_END);
		int x = ftell(dat) / sizeof(int);
		printf("brel=%d\n", x);
		fseek(dat, 0, SEEK_SET);
		for (int i = 0; i < x; i++) {
			int br;
			fread(&br, sizeof(int), 1, dat);
			printf("%d ", br);
		}
		fclose(dat);
	}
	else {
		printf("\ngreska");
```

```
	}
	return 0;
}
```

63. Написати програм који чита податке из бинарне датотеке *brojevi.dat*, која садржи целе бројеве. Програм треба да прочита и испише на стандардном излазу број који се:
а) налази на првој позицији,
б) налази на задатој позицији,
в) налази на последњој позицији.

```
#include <stdio.h>
int main(void) {
	int broj, pozicija;
	FILE* datoteka;
	datoteka = fopen("brojevi.bin", "rb");
	if (datoteka == NULL) {
		printf("\nDatoteka ne postoji");
	}
	else {
		printf("\nUnesite poziciju:");
		scanf("%d", &pozicija);

		fseek(datoteka, 0 * sizeof(int), SEEK_SET);
		fread(&broj, sizeof(int), 1, datoteka);
		printf("\n1.=%d", broj);

		fseek(datoteka, (pozicija - 1) * sizeof(int), SEEK_SET);
		fread(&broj, sizeof(int), 1, datoteka);
		printf("\n%d.=%d", pozicija, broj);

		fseek(datoteka, (-1) * sizeof(int), SEEK_END);
		fread(&broj, sizeof(int), 1, datoteka);
		printf("\nposlednji=%d", broj);
		fclose(datoteka);
	}
	return 0;
}
```

64. Написати програм који ажурира бинарну датотеку *brojevi.dat* која садржи целе бројеве. Датотеку треба ажурирати тако да се број који је паран замени са нулом (непарни бројеви остају исти).
Пример:
Пре ажурирања: 5 2 3 1 4 8 7
После ажурирања: 5 0 3 1 0 0 7

```
#include <stdio.h>
int main(void) {
int broj, nula = 0;
	FILE* datoteka;
	datoteka = fopen("brojevi.dat", "r+b");
	if (datoteka != NULL) {
		while (fread(&broj, sizeof(int), 1, datoteka) != 0) {
			if (broj % 2 == 0) {
				fseek(datoteka, (-1) * sizeof(int), SEEK_CUR);
				fwrite(&nula, sizeof(int), 1, datoteka);
				fflush(datoteka);
			}
		}

		fclose(datoteka);
	}
```

```
	return 0;
}
```

65. Написати програм који ажурира бинарну датотеку *brojevi.dat* која садржи целе бројеве. Датотеку треба сортирати у нерастућем редоследу.

```
#include <stdio.h>
int main(void) {
	int br_elemenata, i, j, br_i, br_j;
	FILE* datoteka;
	datoteka = fopen("brojevi.dat", "r+b");
	if (datoteka == NULL) {
		printf("Datoteka ne postoji!\n");
	}
	else {
		fseek(datoteka, 0, SEEK_END);
		br_elemenata = ftell(datoteka) / sizeof(int);
		printf("\nBroj elemenata =%d", br_elemenata);
		for (i = 0; i < br_elemenata - 1; i++) {
			for (j = i + 1; j < br_elemenata; j++) {
				fseek(datoteka, i * sizeof(int), SEEK_SET);
				fread(&br_i, sizeof(int), 1, datoteka);
				fseek(datoteka, j * sizeof(int), SEEK_SET);
				fread(&br_j, sizeof(int), 1, datoteka);
				if (br_i > br_j) {
					fseek(datoteka, i * sizeof(int), SEEK_SET);
					fwrite(&br_j, sizeof(int), 1, datoteka);
					fseek(datoteka, j * sizeof(int), SEEK_SET);
					fwrite(&br_i, sizeof(int), 1, datoteka);
					fflush(datoteka);
				}
			}
		}
		fclose(datoteka);
	}
	return 0;
}
```

66. Написати потпрограм који омогућава кориснику унос података за n производа у бинарну датотеку *proizvodi.dat*.

```
#include <stdio.h>
typedef struct proizvod {
	int sifra;
	char naziv[20];
	double cena;
}TPROIZVOD;
void unesi_proizvode(int n) {
	FILE* datoteka;
	int i;
	TPROIZVOD pr;
	datoteka = fopen("proizvodi.dat", "wb");
	if (datoteka == NULL) {
		printf("Datoteka ne postoji!\n");
	}
	else {
		for (i = 0; i < n; i++) {
			printf("\nSifra:");
			scanf("%d", &pr.sifra);
			printf("\nNaziv:");
			fflush(stdin);
			gets(&pr.naziv);
```

```
                        fflush(stdin);
                        printf("\nCena:");
                        scanf("%lf", &pr.cena);
                        fwrite(&pr, sizeof(TPROIZVOD), 1, datoteka);
                }
                fclose(datoteka);
        }
}
int main(void) {
        unesi_proizvode(5);
        return 0;
}
```

67. Написати потпрограм који чита производе из бинарне датотеке *proizvodi.dat* и креира извештај *izvestaj.txt*. Извештај чине подаци о свим производима који се налазе у бинарној датотеци.

```
#include <stdio.h>
typedef struct proizvod {
        int sifra;
        char naziv[20];
        double cena;
}TPROIZVOD;
void napravi_izvestaj(void) {
        TPROIZVOD pr;
        int rb;
        FILE* ulaz;
        FILE* izlaz;
        rb = 0;
        ulaz = fopen("proizvodi.dat", "rb");
        if (ulaz == NULL) {
                printf("Datoteka ne postoji!\n");
        }
        else {
                izlaz = fopen("izvestaj.txt", "w");
                fprintf(izlaz, "%-5s%-8s%-20s%8s\n", "rb.", "sifra", "naziv", "cena");
                while (fread(&pr, sizeof(TPROIZVOD), 1, ulaz) != 0) {
                        rb++;
                        fprintf(izlaz, "%-5d%-8d%-20s%8lf\n", rb, pr.sifra, pr.naziv,
pr.cena);
                }
                fclose(izlaz);
                fclose(ulaz);
        }
}
int main(void) {
        napravi_izvestaj();
        return 0;
}
```

3.2. Задаци за вежбање

68. Дат је низ студената. Написати следеће потпрограме:
    а) Написати процедуру која чита податке о студентима из текстуалне датотеке *podaci.txt*. Пријава садржи следеће податке: име и презиме студента, предмет и оцену.
    Пример садржаја датотеке:
    Zika Zikic
    Matematika
    6
    Marko Markovic

Matematika
8
Jana Janic
Programiranje
6

б) Написати процедуру која сортира низ студената према просеку у нерастућем редоследу.

69. Дата је текстуална датотека *ulaz.txt*. У једној линији текстуалне датотеке речи су одвојене једне од других једним или више празних знакова. Написати програм који приказује линију која има највише речи.
70. Дата је једноструко-спрегнута листа пријава студената. Пријава садржи: број индекса, име и презиме студента, назив предмета и оцену. Написати програм који прави извештај у текстуалној датотеци. Извештај приказује просечну оцену за сваки предмет који се налази у листи пријава.
71. Дата је једноструко-спрегнута листа пријава студената. Пријава садржи: број индекса, име и презиме студента, назив предмета и оцену. Написати програм који уписује просечну оцену и број положених предмета за сваког студента који се налази у листи пријава.
72. Дата је текстуална датотека. Написати програм који приказује линију датотеке у којој се налази највише самогласника.
73. Дате су следеће бинарне датотеке:
    proizvodi.bin – садржи производе. За сваки производ чувају се: шифра, назив, цена и количина,
    dobavljaci.bin – садржи податке о добављачима. За сваког добављача чувају се: назив, адреса и број телефона,
    dobavlja.bin – садржи податке и томе који добављач добавља који производа. За свако добављање чувају се назив добављача и шифра производа.
    Креирати функцију која ће у текстуалној датотеци креирати извештај *izvestaj.txt* о добављачима који добављају производе чија је количина стања мања од неке задате вредности.

3.3. Задаци са колоквијума

74. Имплементирати следеће потпрограме:
    а) Функцију која проверава да ли је задати карактер слово. Уколико вам је потребно можете да искористите чињеницу да карактер *A* има ASCII вредност 65, а карактер *a* има ASCII вредност 97.
    б) Функцију која на основу улазног стринга креира најдужи подниз кога чине слова (A-Z и a-z). Тестни пример: улазни стринг s23d f e4**asdd**44f.ff//, излазни стринг **asdd.**
    в) Функцију која чита улазну текстну датотеку **(ulaz.txt)** и формира излазну тексту датотеку **(izlaz.txt)** на следећи начин:
    – Излазна датотека има исти број редова као и улазна датотека.
    – Један ред улазне датотеке у излазној датотеци се замењује најдужим поднизом кога чине слова из одговарајућег реда улазне датотеке.
    Тестни пример:
    Уколико је у трећем реду улазне датотеке стринг: 123 33 3 3-,. тада је трећи ред излазне датотеке празан.
    Уколико је у петом реду улазне датотеке стринг: g h 62 sjhg asj7 asd ;dkd **kays** d22 dff f f тада је пети ред излазне датотеке: **kays.**
75. Имплементирати следеће потпрограме:
    а) Функцију која проверава да ли је задати карактер цифра.
    б) Функцију која на основу улазног стринга креира најдужи подниз кога чине цифре. Тестни пример: улазни стринг 12 23 3553 =((=)&, излазни стринг 3553.

в) Функцију која чита улазну текстну датотеку **(ulaz.txt)** и формира излазну тексту датотеку **(izlaz.txt)** на следећи начин:
– Излазна датотека има исти број редова као и улазна датотека.
– Један ред улазне датотеке у излазној датотеци се замењује најдужим поднизом кога чине цифре из одговарајућег реда улазне датотеке.
Тестни пример:
Уколико је у трећем реду улазне датотеке стринг: 123 33 3 3-, тада је трећи ред излазне датотеке 123.
Уколико је у петом реду улазне датотеке стринг: dff f f, тада је пети ред излазне датотеке празан.

76. У текстуалној датотеци *ulaz.txt* налази се текст. Поред текста у овом фајлу се налазе неозначени бројеви. Неозначени број чини низ узастопних цифара. Написати програм који рачуна суму и средњу вредност ових бројева. Бројеви се не преносе у нови ред.
Структура улазне датотеке:
asdd12 ddd15 dddgh51hh3
3adb jk !!!*
Тестни пример:
```
Сума је: 12+15+51+3+3
Средња вредност је: (12+15+51+3+3)/5
```

77. У текстуалној датотеци *ulaz.txt* налази се текст у једној линији чија је дужина мања од 100 карактера. Све карактере који се у датотеци *ulaz.txt* појављују два или више пута уписати у текстуалну датотеку *izlaz.txt* у формату који је ниже наведен (карактер->број појављивања X).
Тестни подаци:

ulaz.txt	izlaz.txt (формат излазне датотеке)
288081220fghk=+	2->3X; 8->4X; 0->2X;

78. Дат је низ студената:
```
STUDENT xs[] = {
        { "2016_0001", "Andrej Andric" },
        { "2016_0002", "Sofija_Sofic" },
        { "2016_0003", "Jovan Jovic" },
        { "2016_0004", "Sara Saric" },
        { "2016_0005", "Milica Milic" }
    };
```
Дат је низ предмета:
```
PREDMET xp [] = {
        { 1,"Programiranje 1" },
        { 2,"Programiranje 2" },
        { 3,"UIS" },
        { 4,"Matematika" }
    };
```
Имплементирати следеће потпрограме:
а) Функцију за унос нове пријаве у једностуко спрегнуту листу. За сваку пријаву чувају се следеће информације: *број индекса студента*, *шифра предмета* и *оцена*. Унос пријаве је могућ само ако су испуњени следећи предуслови:
– Студент за кога се уноси пријава постоји регистрован у низу студената
– Предмет за који се уноси пријава постоји регистрован у низу предмета
– Студент за који се уноси пријава није положио тај предмет
– Оцена може да узме вредности од 5 до 10
– Један студент за један предмет може да има само једну позитиву пријаву (пријаву са оценом већом од 5, али може да има више пријава са оценом 5)

б) Функцију која у текстној датотеци *rang_lista.txt* креира извештај о ранг листи студената на основу просечне оцене студената у формату који је дат ниже. Уколико два студента имају исту просечну оцену они се рангирају се по броју положених испита.

```
-------------------------------------------------------------------------------------
Rang lista studenata na osnovu prosecne ocene
-------------------------------------------------------------------------------------
Rb.    Broj indeksa Student             Prosecna ocena           Broj polozenih ispita
-------------------------------------------------------------------------------------
1.     2016_0005    Milica Milic        9.5                      2
2.     2016_0004    Sara Saric          9.5                      1
3.     2016_0003    Jovan Jovic         9                        3
4.     2016_0003    Sofija_Sofic        7                        2
5.     2016_0001    Andrej Andric       7                        1
```

в) Функцију која омогућава кориснику да преко одговарајућег корисничког менија позове претходно имплементиране функције из овог задатка:

```
Korisnicki meni:
1) Unesi novu prijavu
2) Kreiraj rang listu
3) Kraj programa
Vas izbor:__
```

79. Дат је низ студената:

```
STUDENT xs[] = {
        { "2016_0001", "Andrej Andric" },
        { "2016_0002", "Sofija_Sofic" },
        { "2016_0003", "Jovan Jovic" },
        { "2016_0004", "Sara Saric" },
        { "2016_0005", "Milica Milic" }
    };
```

Дат је низ предмета:

```
PREDMET xp [] = {
        { 1,"Programiranje 1" },
        { 2,"Programiranje 2" },
        { 3,"UIS" },
        { 4,"Matematika" }
    };
```

Имплементирати следеће потпрограме:

а) Функцију за унос нове пријаве у једностуко спрегнуту листу. **Пријаве се уносе на почетак листе.** За сваку пријаву чувају се следеће информације: *број индекса студента*, *шифра предмета* и *оцена*. Унос пријаве је могућ само ако су испуњени следећи предуслови:

– Студент за кога се уноси пријава постоји регистрован у низу студената

– Предмет за који се уноси пријава постоји регистрован у низу предмета

– Студент за који се уноси пријава није положио тај предмет

– Оцена може да узме вредности од 5 до 10

– Један студент за један предмет може да има само једну позитиву пријаву (пријаву са оценом већом од 5, али може да има више пријава са оценом 5)

б) Функцију која у текстној датотеци *rang_lista.txt* креира извештај о просечној оцени за сваки предмет у формату који је дат ниже. **Извештај је сортиран на основу просечне оцене за предмет.** Уколико је просечна оцена за два или више предмета иста сортирање се врши према броју пријава са оценом већом од 5.

```
-------------------------------------------------------------------------------------
Izvestaj za predmete
-------------------------------------------------------------------------------------
Rb. Sifra    Predmet            Prosek       Br.prij. (>5)      Br.prij.(=5)
-------------------------------------------------------------------------------------
1.    1      UIS 1              9            3                  3
```

```
2.     2     Programiranje 2    9            2                  2
3.     3     Programiranje 1    8            2                  3
4.     4     Matematika         8            1                  4
```

в) Функцију која омогућава кориснику да преко одговарајућег корисничког менија позове претходно имплементиране функције из овог задатка:

```
Korisnicki meni:
1) Unesi novu prijavu
2) Kreiraj izvestaj
3) Kraj programa
Vas izbor:__
```

80. Креирати једноструко спрегнуту листу у којој се чувају подаци о резултатима студената на првом и другом колоквијуму из предмета Програмирање 1. Један елемент листе садржи следеће податке: *број индекса* (стринг максималне дужине 9 карактера), *име и презиме студента* (стринг максималне дужине 30 карактера), *редни број колоквијума* (могуће вредности 1 и 2), *број поена на колоквијуму* (вредност од 0 – 50). Имплементирати следеће потпрограме:

а)Потпрограм који додаје нови елемент у листу. У листи за једног студента може постојати максимално два резултата (резултат са 1.колоквијума и резултат са 2.колоквијума). За један колоквијум за једног студента може постојати само један резултат. Приликом убацивања новог резултата у листу на стандардном излазу приказати одговарајућу поруку:

– Уколико је резултат успешно убачен у листу приказати поруку: `резултат је успешно убачен` у листу

– Уколико резултат није убачен у листу приказати поруку: `резултат није убачен у листу`

б) У текстуалној датотеци *polozili.txt* креирати извештај који приказује све студенте који су положили предмет Програмирање 1, сортиран у опадајућем редоследу по броју поена у формату који је дат ниже. Студент је положио испит, уколико у збиру на 1. и 2. колоквијуму има више од 45 поена, а при томе мора на сваком колоквијуму да оствари више од 15 поена.

Формат текстуалне датотеке *polozili.txt*

```
Predmet: programiranje 1
Rb.    Broj indeksa           Student            1.kol        2.kol.       ukupno
1 .          12/2014          Tamara Nizic       50           50           100
2 .          11/2015          Jelena Stringic    50           38           88
```

в) У датотеци *nisu_polozili.txt* креирати извештај који приказује све студенте који нису положили предмет Програмирање 1, сортиран у опадајућем редоследу по броју поена у формату који је дат ниже. Студент је положио испит, уколико у збиру на 1. и 2. колоквијуму има више од 45 поена, а при томе мора на сваком колоквијуму да оствари више од 15 поена.

Формат текстуалне датотеке *nisu_polozili.txt*

```
Predmet: programiranje 1
Rb.    Broj indeksa           Student            1.kol        2.kol.       ukupno
1 .          65/2015          Marko Listic       40           14           54
2 .          78/2015          Darko Datotecic    25           15           40
```

81. Дата је једноструко спрегнута листа (листа добављача) у којој се чувају подаци о добављачима (шифра, назив). У листи се сви елементи разликују према шифри добављача и дата је једноструко спрегнута листу (листа производа) у којој се чувају подаци о производима који се набављају од добављача (шифра добављача, шифра производа, назив производа, цена). Имплементирати следеће потпрограме:

а) Имплементирати функцију која омогућава унос производа у листу, тако да није могуће унети производ за добављача који не постоји у листи добављача.

б) Имплементирати функцију која креира текстуалну датотеку *izvestaj.txt* у којој су приказани сви добављачи и сви производи који се добављају од тог добављача.

Формат текстуалне датотеке *izvestaj.txt*

Dobavljac:

Sifra Naziv
D1 Dobavljac 1
Proizvodi
Sifra Naziv Cena
P1 Proizvod 1 30.50
P2 Proizvod 2 50.40
Ukupno 2 proizvod.
Dobavljac:
Sifra Naziv
D2 Dobavljac 2
Proizvodi
Sifra Naziv Cena
P3 Proizvod 3 30.50
Ukupno 1 proizvod.

3.4. Задаци са испитних рокова

82. Имплементирати следеће потпрограме:
а) Имплементирати функцију која рачуна збир цифара неког задатог броја.
б) Написати главни програм који са стандардног улаза прихвата n бројева и приказује број чији је збир цифара највећи. Искористити претходно имплементирану функцију из задатака а).
в) Написати функцију која рачуна колико има парних троцифрених бројева чији је збир цифара 13. Искористити претходно имплементирану функцију из задатака а). У главном програму позвати имплементирану функцију.
г) Написати функцију која све троцифрене парне бројеве чији је збир цифара једнак неком задатом броју пребацује у једноструко спрегнуту листу тако да листа буде сортирана у опадајућем редоследу (није дозвољено сортирање листе након уноса елемената у листу). У главном програму приказати садржај листе.
д) Написати функцију која све троцифрене парне бројеве чији је а пребацује у текстуалне датотеке тако да се сви троцифрени бројеви прве стотине налазе у датотеци 100.txt, друге стотине у датотеци 200.txt и тако редом. Након уписа елемената у текстуалну датотеку, приказати садржај ових датотека на стандардном излазу. Формат уписа у текстуалну датотеку је дат ниже.
Бројеви 1.стотине:
158;176;194;
83. Текстуална датотека ulaz.txt у првом реду садржи број елемената низа А, а у другом реду елементе низа А који су између одвојени знаком ;. У трећем реду у датотеци се налази број елемената низа Б, а у четвртом реду елементи низа Б који су између одвојени знаком ;. Формирати нови низ Ц у неопадајућем редоследу од елемената из низа А и низа Б.
84. Дата је текстуална датотека brojevi.txt у коју су уписани цели бројеви. Формирати једноструко спрегнуту листу од бројева који се налазе у датотеци brojevi.txt у обрнутом редоследу. Избацити све бројеве из листе који се понављају, тако да у листи сви бројеви буду међусобно различити.
Тестни пример:
brojevi.txt
1 4 1 9 1 6 3 3 9 1 4 8
Резултат:
Листа: 8 4 1 9 3 3 6 1 9 1 4 1
Листа без понављања: 8 4 1 9 3 6
85. Имплементирати следеће потпрограме:

а) Написати функцију која за улазни декадни број креира њему одговарајући бинарни број.
б) Написати потпрограм који из текстуалне датотеке *ulazDek.txt* чита све декадне бројеве и у текстуалну датотеку *izlazBin.txt* уписује одговарајуће бинарне бројеве.

86. Написати програм који све прелепе бројеве у неком задатом интервалу уписује у текстуалну датотеку *divota.txt*. Број је прелеп уколико су све цифре броја различите.
87. Дата је текстуална датотека *nizovi.txt* у коју су уписани подаци о два улазна низа. У првом реду датотеке уписана је димензија оба низ у следећем формату **димензија I низа - димезија II низа** (пример 1: 6 - 12). Имплементирати следеће потпрограме:
а) Написати функцију која учитава низове из датотеке и приказује елементе низова.
б) Написати функцију која прави сортирану листу од ова два низа. Резултујућа листа два низа представља њихову унију, али тако да се елементи не понављају. Сортирање вршити приликом убацивања елемената у листу.
в) Написати функцију која приказује елементе листе.
Формат датотеке *nizovi.txt*:
7-6
1 3 4 6 7 3 5
2 5 7 9 1 6
88. Дата је текстуална датотека *nizovi.txt* у коју су уписани подаци о два улазна низа. У првом реду датотеке уписана је димензија оба низ у следећем формату **димензија I низа - димезија II низа** (пример 1: 6 - 12). Имплементирати следеће потпрограме:
а) Написати функцију која учитава низове из датотеке и приказује елементе низова.
б) Написати функцију која садржи све елементе из првог низа. За сваки елемент у чвору листе се чува информација колико пута се тај број појављује у другом низу.
в) Написати функцију која приказује елементе листе и за сваки елемент приказује број појављивања елемента у листи. Формат приказа листе:

```
Број   Број појављивања
-------------------------------------------------------------------------------
1      4
4      2
9      2
6      1
3      2
99     0
```

89. Имплементирати следеће потпрограме:
а) Имплементирати потпрограм која за неки задати број испитује да ли је палиндром. Тест пример: 12321 (јесте), 2343 (није).
б) Написати потпрограм и главни програм који са стандардног улаза прихвата n бројева (све док корисник не унесе 0) и у текстуалну датотеку *palindrom.txt* уписује све бројеве које је корисник унео и који су палиндроми, а у текстуалну датотеку *ostali_brojevi.txt* уписује бројеве који нису палиндроми Написати потпорграм који приказује садржај ових текстуалних датотека.
в) Дата је квадратна матрица димензије nXn. Написати потпрограм и главни програм који чита садржај текстуалне датотеке *palindrom.txt* и бројеве уписује у матрицу по ободу (у круг).
Тестни пример:
Палиндром: 101, 202, 33, 44, 55.
У наставку је дат приказ задате матрице и излазне матрице.
Задата матрица:

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

Излазна матрица:

101	202	33	44
202	1	1	55
101	1	1	101
55	44	33	202

90. Дата је квадратна матрица димензије 5X5. Имплементирати следеће потпрограме:
    а) Имплементирати функцију која рачуна суму задате колоне матрице.
    б) Имплементирати процедуру која за задату квадратну матрицу у текстуалну датотеку *matrica.txt* уписује суму елемената по колонама. Текстуална датотека треба бити форматирана на следећи начин:
```
Сума ел. 1. колоне је:
Сума ел. 2. колоне је:
Сума ел. 3. колоне је:
Сума ел. 4. колоне је:
Сума ел. 5. колоне је:
```
91. Дата је квадратна матрица димензије 5X5. Имплементирати следеће потпрограме:
    а) Имплементирати функцију која рачуна суму задатог реда матрице.
    б) Имплементирати процедуру која за задату квадратну матрицу у текстуалну датотеку *matrica.txt* уписује суму елемената по редовима. Текстуална датотека треба бити форматирана на следећи начин:
```
Сума ел. 1. реда је:
Сума ел. 2. реда је:
Сума ел. 3. реда је:
Сума ел. 4. реда је:
Сума ел. 5. реда је:
```
92. Дата је квадратна матрица димензије 4x4. Написати функцију која у текстуалну датотеку *izlaz.txt* колико пута се сваки елемент листе појављује на споредној дијагонали дате матрице у следећем формату:
    izlaz.txt:
```
1 - se pojavljuje 0x
4 - se pojavljuje 1x
7 - se pojavljuje 2x
8 - se pojavljuje 0x
```
93. Дата је текстуална датотека *matrica.txt* у којој се налазе елементи квадратне матрице форматирани на следећи начин:
    – У једном реду текстуалне датотеке налазе се елементи једног реда матрице,
    – Елементи једног реда матрице међусобно су одвојени празним местом,
    – У текстуалној датотеци не постоји информација о броју редова матрице, али се зна да број елемената једног реда квадратне матрице није већи од 5.
    Пример текстуалне датотеке:
    1 2
    11 22
    Написати потпрограм који рачуна унију елемената две задате колоне матрице која се налази у датотеци *matrica.txt* и смешта елементе уније у једноструко спрегнуту листу при чему није дозвољено понављање елемената у листи (сви елементи у листи су међусобно различити).
94. Дата је текстуална датотека *matrica.txt* у којој се налазе елементи квадратне матрице форматирани на следећи начин:
    – У једном реду текстуалне датотеке налазе се елементи једног реда матрице
    – Елементи једног реда матрице међусобно су одвојени празним местом
    – У текстуалној датотеци не постоји информација о броју редова матрице, али се зна да број елемената једног реда квадратне матрице није већи од 5.

Пример текстуалне датотеке:
1 2
11 22
Написати потпрограм који рачуна унију елемената два задата реда матрице која се налази у датотеци *matrica.txt* и смешта елементе уније у једноструко спрегнуту листу при чему није дозвољено понављање елемената у листи (сви елементи у листи су међусобно различити).

95. Дата је текстуална датотека *matrica.txt* у којој се налазе елементи квадратне матрице целих бројева не већи од 10x10. У првом реду ове датотеке налази се информација о димензији матрице, а у сваком следећем се налазе елементи одговарајућег реда матрице (у 2. реду датотеке налазе се елементи 1. реда матрице итд.). Имплементирати следеће потпрограме:
а) Функцију која садржај текстне датотеке *matrica.txt* пребацује у матрицу.
б) Функцију која рачуна средњу вредност задате колоне матрице.
в) Функцију која све елементе изнад споредне дијагонале матрице уписује у једноструко спрегнуту листу.
г) Функцију која приказује елементе једноструко спрегнуте листе. Приказати све елементе једноструко спрегнуте листе уназад. *Пример*: уколико се у листи налазе елементи: 1,2,3,4 приказати елементе редом: 4,3,2,1. Не користити помоћне структуре података ни типове.
д) Функцију која све елементе испод споредне дијагонале матрице уписује у *LIFO* низ (нови елемент се убацује на почетак низа).
ђ) Функцију која приказује садржај низа целих бројева.
е) Функцију која формира текстну датотеку *razlika.txt* коју чине сви елементи изнад споредне дијагонале квадратне матрице којих нема испод споредне дијагонале. У текстној датотеци *razlika.txt* сви елементи су међусобно различити.
ж) Функцију кориснички мени преко које се позивају претходно имплементиране функције.

96. Дата је улазна текстуална датотека *matrica.txt* у којој се чувају елементи матрице целобројног типа. У првом реду текстуалне улазне датотеке чува се димензија матрице у формату nXm (нпр. 2X3), након чега датотека садржи онолико редова колико има колона матрице (сваки ред улазне текстуалне датотеке представља елементе те колоне матрице). Максимална димензија улазне матрице може бити 10X10.
Пример улазне текстуалне датотеке:
2X3
12 22
12 34
28 22
Матрица:
12 12 28
22 34 22
а) Написати функцију која елементе матрице који је чувају у улазној текстуалној датотеци *matrica.txt* пребацује у матрицу. Написати функцију која приказује тако формирану матрицу.
б) Написати функцију која на основу улазне матрице произвољне димензије (максимална димензија матрице може бити 10X10) формира једноструко спрегнуту листу која садржи:
– онолико елемената колико има различитих елемената у улазној матрици,
– број појављивања сваког броја који се налази у улазној матрици
Написати функцију која приказује садржај тако формиране листе.

97. Дате су текстуалне датотеке *matrica1.txt* и *matrica2.txt* у којима се налазе елементи матрица форматирани на следећи начин:
– У првом реду датотеке уписан је број редова и колона матрице,
– У сваком наредном реду налазе се елементи i-тог реда матрице,
– Елементи једног реда матрице међусобно су одвојени празним местом.
Пример текстуалне датотеке:

matrica1.txt *matrica2.txt*

2 3 3 2
1 2 2 4 2
3 1 1 3 1
1 5

Имплементирати следеће потпрограме:

а) Написати потпрограм који учитава матрицу из датотеке и учитати матрицу 1 и матрицу 2. Потребно је написати један потпрограм и позвати га два пута.

б) Написати потпрограм који приказује садржај неке матрице.

в) Написати потпрограм који формира нову матрицу 3 тако што рачуна производ матрице 1 и матрице 2. Приказати садржај матрице 1, матрице 2 и матрице 3. Обавезно искористити потпрограм из претходног захтева.

98. Дата је текстуална датотека matrica.dat у коју су уписани подаци квадратне матрице. У првом реду датотеке уписана је димензија матрице, а сваки наредни ред одговара реду матрице. Елементи једног реда матрице међусобно су одвојени празним местом. Имплементирати следеће функције:

а) Написати функцију која учитава матрицу из датотеке и приказује елементе матрице.

б) Написати функцију која креира листу од елемената који се налазе испод главне дијагонале матрице тако да листа буде сортирана и без подављања елемената. Сортирање вршити приликом убацивања елемената у листу.

в) Написати функцију која приказује елементе листе, а који се налазе и изнад главне дијагонале матрице.

Формат датотеке *matrica.txt*:

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

99. Дат је низ студената и низ предмета. За сваког студента се чувају следеће информације: *број индекса*, *име и презиме* и *број предмета који је положио тај студент*, док се за сваки предмет чувају информације: *шифра предмета*, *назив предмета* и *број студената који су положили тај предмет*. Имплементирати следеће потпрограме:

а) Функцију за унос нове пријаве у једностуко спрегнуту листу која садржи показивач на **први елемент** листе. **Пријаве се уносе на крај листе**. За сваку пријаву чувају се следеће информације: *број индекса студента*, *шифра предмета* и *оцена*. Унос пријаве је могућ само ако су испуњени следећи предуслови:

– Студент за кога се уноси пријава постоји регистрован у низу студената.

– Предмет за који се уноси пријава постоји регистрован у низу предмета.

– Студент за који се уноси пријава није положио тај предмет.

– Оцена може да узме вредности од 5 до 10.

– Један студент за један предмет може да има само једну позитиву пријаву (пријаву са оценом већом од 5, али може да има више пријава са оценом 5).

– При сваком уносу позитивне пријаве ажурирати број студената који су положили предмет и број предмета који је студент са пријаве положио.

б) Функцију која у текстној датотеци *izvestaj.txt* креира **извештај о предметима који је сортиран по броју студената који су положили тај предмет у опадајућем редоследу** у формату који је дат ниже.

```
--------------------------------------------------------------------------------
Izvestaj za predmete
```

```
-------------------------------------------------------------------------------
Rb. Sifra    Predmet            Br.prij.
-------------------------------------------------------------------------------
1.     1     UIS 1              4
2.     2     Programiranje 2    3
3.     3     Programiranje 1    3
4.     4     Matematika         1
```

в) Функцију која омогућава кориснику да преко одговарајућег корисничког менија позове претходно имплементиране функције из овог задатка. Овај део задатка се оцењује само ако је урађен задатак под а) или б).

```
Korisnicki meni:
1) Unesi novu prijavu
2) Kreiraj izvestaj
3) Kraj programa
Vas izbor:__
```

100. Дат је низ студената и низ предмета. За сваког студента се чувају следеће информације: *број индекса*, *име и презиме* и *број предмета који је положио тај студент*, док се за сваки предмет чувају информације: *шифра предмета*, *назив предмета* и *број студената који су положили тај предмет*. Имплементирати следеће потпрограме:

а) Функцију за унос нове пријаве у једностуко спрегнуту листу која садржи показивач на **први елемент** листе. **Пријаве се уносе на крај листе**. За сваку пријаву чувају се следеће информације: *број индекса студента*, *шифра предмета* и *оцена*. Унос пријаве је могућ само ако су испуњени следећи предуслови:

– Студент за кога се уноси пријава постоји регистрован у низу студената.

– Предмет за који се уноси пријава постоји регистрован у низу предмета.

– Студент за који се уноси пријава није положио тај предмет.

– Оцена може да узме вредности од 5 до 10.

– Један студент за један предмет може да има само једну позитиву пријаву (пријаву са оценом већом од 5, али може да има више пријава са оценом 5).

– При сваком уносу позитивне пријаве ажурирати број студената који су положили предмет и број предмета који је студент са пријаве положио.

б) Функцију која у текстној датотеци *izvestaj.txt* креира **извештај о студентима који је сортиран по броју предмета које је студент положио у опадајућем редоследу** у формату који је дат ниже.

```
-------------------------------------------------------------------------------
Izvestaj za studente
-------------------------------------------------------------------------------
Rb. Br.ind   Student            Br.pred.
-------------------------------------------------------------------------------
1. 2016/0001 Student1           4
2. 2016/0001 Student2           4
3. 2016/0001 Student3           3
```

в) Функцију која омогућава кориснику да преко одговарајућег корисничког менија позове претходно имплементиране функције из овог задатка. Овај део задатка се оцењује само ако је урађен задатак под а) или б).

```
Korisnicki meni:
1) Unesi novu prijavu
2) Kreiraj izvestaj
3) Kraj programa
Vas izbor:__
```

101. У свакој линији текстуалне улазне датотеке *studenti.txt*, налазе се подаци о студентима и подаци о положеним испитима. Свака линија у овој улазној датотеци садржи податке о једном студенту и састоји се од више колона које су међусобно одвојене знаком тачказарез (;). У првој колони се налази име и презиме студента (максималан број карактера је 50). У другој колони

се налази број индекса студента (максималан број карактера је 9), након чега следе подаци о положеним испитима (уколик их има). За сваки испит чувају се подаци о шифри испита (максималан број карактера је 6) и оцени са тог испита (оцене су у интервалу од 6 до 10).
Тестни пример – приказ садржаја улазне датотеке studenti.txt:

```
Marko Markovic;0001/2016;P1;8;OIKT;9;UIS;6;
Dejan Denic;0011/2016;
```

Имплементирати следеће потпрограме:
а) Имплементирати функцију која пребацује податке из текстуалне датотеке **studenti.txt** у једноструко спрегнуту листу студената.
б) Имплементирати функцију која приказује садржај листе студената. У једном реду приказати основне податке о студентима: име и презиме, број индекса, број положених испита као и списак положених испита (за сваки положени испит приказати шифру предмета и оцену).
в) Имплементирати функцију која сортира листу студената у опадајућем редоследу према просечној оцени. Уколико је просечна оцена за два студента иста, сортирање извршити према броју положених испита.
г) Имплементирати функцију која приказује просечну оцену за сваки положени предмет.
д) Имплементирати функцију за унос нове пријаве (унос новог елемента у листу студената). Приликом уноса нове пријаве корисник уноси број индекса студента, име и презиме (само у случају да студент не постоји у листи студената), шифру предмета и оцену. Приликом уноса нове пријаве извршити проверу да ли је студент положио тај предмет. Уколико је студент положио тај предмет приказти оцену коју је студент добио, уколико није положио убацити нову пријаву.
ђ) Имплементрати функцију која приказује кориснички мени. Омогућити кориснику да избором одговарајуће опције менија позове одговарајућу функцију. Обавезан део задатка је имплементација корисничког менија. Уколико ова функција није имплементирана задатак неће бити прегледан.

```
Korisnicki meni:
1) Prebaci podatke iz liste u datoteku
2) Prikazi listu studenata
3) Sortiraj listu studenata
4) Prikazi prosecne ocene za sve predmete
5) Kraj programa
Vas izbor:__
```

102. Подаци о потрошачкој корпи се налазе у текстној датотеци *korpa.txt*. У једном реду ове датотеке налазе се подаци о потрошачкој корпи за једног потрошача. Након имена и презимена потрошача следе информације о броју артикала и скупу артикала који се налазе у тој корпи за тог потрошача. За сваки арткикал се чувају следећи подаци: *назив артикла* (максимална дужина је 50 карактера), *цена* и *количина*.
Дат је пример једне потрошачке корпе (један ред у текстној датотеци korpa.txt) :

```
Jana Janic-3:mleko:85.00:25:meso:550.00:5:sir:450.00:4
```

Дат је пример једне потрошачке корпе (један ред у текстној датотеци korpa.txt) :

```
Marko Maric-0
```

Датотека је добро форматирана тако да је цена једног артикла иста за сваког потрошача (у свакој потрошачкој корпи).
Имплементирати следеће потпрограме:
а) Имплементирати функцију која пребацује податке из текстуалне датотеке *korpa.txt* у једноструко спрегнуту листу потрошача.
б) Имплементирати функцију која приказује садржај листе потрошача.
в) Имплементирати функцију која сортира листу потрошачка у опадајућем редоследу према укупном износу потрошачке корпе. Уколико је износ потрошачке корпе за два потрошача исти, сортирање извршити према броју артикала у корпи.

г) Имплементирати функцију која приказује укупну количину за сваки артикал у листи потрошача.

д) Имплементирати функцију за унос новог артикла у потрошачку корпу за конкретног потрошача. Приликом уноса новог артикла у корпу корисник уноси име и презиме потрошача, назив артикла, цену артикла (само уколико тог артикла нема у потрошачким корпама) и количину. Приликом уноса новог артикла извршити проверу да ли се тај артикл налази у корпи. Уколико артикл постоји у потрошачкој корпи увећати количину артикла у корпи, а уколико не постоји убацити артикл у корпу.

ђ) Имплементрати функцију која приказује кориснички мени. Омогућити кориснику да избором одговарајуће опције менија позове одговарајућу функцију. Обавезан део задатка је имплементација корисничког менија. Уколико ова функција није имплементирана задатак неће бити прегледан.

```
Korisnicki meni:
1) Prebaci podatke iz liste u datoteku
2) Prikazi listu potrosaca
3) Sortiraj listu potrosaca
4) Prikazi sve artikle sa ukupnim kolicinama koji se nalaze u potrosackim korpama
5) Kraj programa
Vas izbor:__
```

103. У текстној датотеци *student.txt* налазе подаци о студентима. У једном реду ове датотеке налазе се подаци о једном студенту: *број индекса*, *име*, *презиме* и *семестар*. Познато је да у овој датотеци не може бити више од 1000 студената.

У текстној датотеци *predmeti.txt* налазе се подаци о предметима. У једном реду ове датотеке налазе се подаци о једном предмету: *шифра предмета*, *назив* и *семестар*. Познато је да у овој датотеци не може бити више од 100 предмета.

Имплементирати следеће потпрограме:

а) Имплементирати функцију која садржај датотеке *student.txt* пребацује у низ студената и функцију која садржај датотеке *predmeti.txt* пребацује у низ предмета.

б) Имплементирати функцију која приказује садржај низ студената и функцију која приказује садржај низа предмета.

в) Подаци о пријавама студената налазе се у једноструко спрегнутој листи. Имплементирати функцију за унос нове пријаве. Приликом уноса пријаве проверити да ли се студент налази у низу студената и предмет у низу предмета. Студент може да полаже само предмете из семестра које је одслушао. Пример: ако је студент 7. семестра он може да полаже само предмете од 1. до 6. семестра. Оцена мора да се унесе у интервалу од 5 до 10. За једног студента, за један предмет у листи може бити унета само једна позитивна оцена (већа од 6).

г) Имплементирати функцију која сортира низ студената на основу просечне оцене у опадајућем редоследу.

д) Креирати посебан извештај за сваког студента. Извештај о једном студенту налази се у посебном документу (фајлу). Назив датотеке се формира на основу имена студента је у формату: *име студента_презиме студента*. У овом документу приказати основне податке о студенту (број индекса, име, презиме, семестар), број положених испита, просечну оцену и списак свих предмета које је студент положио и оцену коју је добио.

ђ) Имплементрати функцију која приказује кориснички мени. Омогућити кориснику да избором одговарајуће опције менија позове одговарајућу функцију. Обавезан део задатка је имплементација корисничког менија. Уколико ова функција није имплементирана задатак неће бити прегледан.

```
Korisnicki meni:
1) Prebaci podatke iz datoteke student.txt
2) Prebaci podatke iz datoteke predmeti.txt
3) Prikazi studente
4) Prikazi predmete
```

```
5) Unesi prijavu
6) Kreiraj izvestaj za sve studente
7) Sortiraj studente
8) Kraj programa
Vas izbor:__
```

104. Дат је низ репрезентација. За сваку репрезентацију се чувају следеће информације: *шифра*, *назив*, *број одиграних утакмица*, *број победа*, *број нерешених*, *број пораза* и *број поена*. Имплементирати следеће потпрограме:

а) Потпрограм за унос нове утакмице у једноструко спрегнуту листу која садржи **показивач на први елемент** листе. **Утакмице се уносе на крај листе**. За сваку утакмицу чувају се следеће информације: *репрезентација домаћин*, *репрезентација гост*, *број голова домаћин* и *број голова гост*. Унос утакмице је могућ само ако су испуњени следећи предуслови:
– Домаћин и гост на утакмици постоје регистровани у низу репрезентација,
– Домаћин и гост морају бити различити,
– Пар *домаћин-гост* или *гост-домаћин* не постоје у листи утакмица,
– Број голова (домаћина/госта) мора бити већи или једнак нули,
– При сваком уносу утакмице ажурирати број одиграних утакмица, број победа, број нерешених, број пораза и број поена репрезентација из утакмице. За сваку победу репрезентација добија три бода а за нерешен резултат један бод.

б) Потпрограм који у текстној датотеци *izvestaj.txt* креира извештај о учинку репрезентација који је уређен према броју остварених бодова у опадајућем редоследу у формату који је дат ниже.

```
-------------------------------------------------------------------------------
Izvestaj o ucinku reprezentacija
-------------------------------------------------------------------------------
Rang. Sifra        Naziv  Br. bodova
-------------------------------------------------------------------------------
1.     1           R1     6
2.     4           R3     4
3.     3           R2     1
```

в) Потпрограм који омогућава кориснику да преко одговарајућег корисничког менија позове претходно имплементиране функције из овог задатка (корисник може више пута да позове функције менија, све док се не одабере опција крај програма).

```
Korisnicki meni:
1) Unesi nove utakmice
2) Kreiraj izvestaj
3) Kraj programa
Vas izbor:__
```

105. У улазној текстној датотеци поред текста налазе се цели неозначени бројеви. Написати програм који рачуна њихову суму и средњу вредност. Број је низ узастопних цифара између белих (бланко) знакова. Нема преноса бројева у нови ред.

Тестни пример:

```
as125 123 gh50hh
3 sdf sdf 556ggg
4
Suma je: 123 + 3 + 4
```

106. У текстуалној датотеци *ulaz.txt* налази се текст. Познато је да једна линија ове датотеке није дужа од 50 карактера. Имплементирати функцију која све речи из улазне датотеке пребацује у излазну датотеку *izlaz.txt* при чему се свака реч у излазној датотеци налази у новом реду. Реч је скуп карактера без бланко знака, која садржи само слова (велика и/или мала), минималне дужине 3 знака.

107. У текстуалној датотеци *ulaz.txt* налази се текст. Познато је да у улазној датотеци не постоји реч која је дужа од 10 знакова и да ниједна линија у датотеци нема више од 500 знакова. Имплементирати следеће потпрограме:
а) Имплементирати функцију која чита садржај улазне текстне датотеке и све речи у којој су сви знаци међусобно различити убацује у једноструко спрегнуту листу. Реч је скуп карактера без бланко знака, која садржи само слова (велика и/или мала), минималне дужине 3 знака у којој два иста знака не могу да се налазе један до другог.
б) Имплементирати функцију која приказује садржај ове листе.

108. Имплементирати функцију која чита садржај улазне датотеке у којој се налазе речи (свака реч се налази у новом реду где је реч скуп карактера без бланко знака, која садржи само слова (велика и/или мала), минималне дужине 3 знака) и која све речи које су палиндроми убацује у једноструко спрегнуту листу. Имплементирати функцију која приказује садржај ове листе.

109. У текстуалној датотеци *ulaz.txt* налази се текст. Креирати нову текстуалну датотеку *izlaz.txt* која садржи све палиндроме из датотеке *ulaz.txt* која је форматирана тако да се сваки палиндром налази у новој линији и да поред сваке речи пише њен редни број. Реч је скуп знакова минималне дужине 3 и која не садржи празна (бланко) места, док је палиндром реч која се исто чита и с лева на десно и с десна на лево. На стандардном излазу приказати укупан број палиндрома.

110. На стандардном излазу приказати укупан број палиндрома. У текстуалној датотеци *ulaz.txt* налази се текст. Имплементирати следеће потпрограме:
а) Имплементирати функцију која креира нову текстуалну датотеку *izlaz.txt* која садржи све речи из датотеке *ulaz.txt* која је форматирана тако да се свака реч налази у новој линији и да поред сваке речи пише и њена дужина. Реч је скуп знакова минималне дужине 2 и која не садржи празна (бланко) места.
б) Имплементирати функцију која на стандардном излазу приказује најдужу реч, а уколико их има више приказати све речи.

111. Компанија Тојота Јапан има развијен систем производње (енгл. *Toyota Production System*) у циљу оптимизације производње и елиминисања губитака. У том смислу је потребно направити програм који ће извршити оцену учинка запосленог на производној линији. **Укупан учинак запосленог се рачуна као сума броја бодова остварених за производњу ауто-дела (нпр. шасије, кочионог система, електро-инсталација итд.) и просечног броја бодова остварених за квалитет израде**. Бодовање се врши према следећем критеријуму:
– Бодови за производњу ауто-дела се обрачунавају на следећи начин: као гранично време за производњу изабрано је 20 мин за које запослени добија 120 бодова. За време које је испод 20 мин, на сваких пола минута испод границе од 20 минута запослени добија додатних 1.4 бода (*увећање за учинак*). За време које је изнад 20 мин, на сваки 0.5 минута изнад границе од 20 минута запосленом се одузима 1.2 бода (*умањење за учинак*).
– Бодове за квалитет даје 7 инжењера за управљање квалитетом, оценама од 10 до 20 (оцене могу бити бројеви са две децимале нпр. 11.34 или 15.65). Број бодова за квалитет се добија када се најмања и највећа оцена одбаце, а на основу осталих оцена израчуна аритметичка средина.
Имплементирати следеће потпрограме:
а) Имплементирати функцију рачуна број бодова које је остварио запослени на производној линији за произведени ауто-део чије је време (м минута) и чији су квалитет инжењери оценили оценама од 10 до 20.
б) Имплементирати процедуру која формита извештај у текстуалној датотеци *ucinak.txt* о учинку запосленог у следећем формату.

```
Toyota Japan
Zaposleni: <ime i prezime zaposlenog>
Vreme izrade: ______ min, broj bodova: =  ______
```

```
Ocene za kvalitet: {10.5; 12.5; 15.5; 12.25; 17.5; 15.5; 15.5}, prosecno bodova =
_____
Ukupno bodova: _____
```

Тестни пример:

Време израде: 21.4 мин, број бодова = 117.6

Оцене за квалитет: {**10.5**; 12.5; 15.5; 12.25; **17.5**; 15.5; 15.5}, просечно бодова = 14.25

Укупно бодова: 117.6 + 14.25 = 131.85

112. Марко Марковић је студент 1. године ФОН-а и на свом рачуну у банци има X динара. Марко је имао договор са својим оцем да уколико положи Програмирање 1 у јунском испитном року, отац Марку почев од 1. септембра уплаћује на рачун у банци Y динара следећих D дана на следећи начин:

– 1. септембра отац Марку треба да уплати Z динара.

– Сваког наредног дана отац Марку треба да уплаћује V динара више него претходног дана.

Имплементирати функцију која рачуна колико Марко има динара у банци након D дана и креира извештај у текстуалној датотеци (*banka.txt*) у формату који је дат ниже.

Формат извештаја у текстуалној датотеци *banka.txt* за X = 100, Y = ..., V= 2, Z= 10, D=30 биће:

```
BANKOVNI IZVESTAJ
Marko Markovic, stanje = 100 dinara
1. dana: uplata = 10 dinara, saldo 110 dinara
2. dana: uplata = 12 dinara, saldo 122 dinara
3. dana: uplata = 14 dinara, saldo 136 dinara
...
30. dana: uplata = ... dinara, saldo ... dinara
Hvala tata.
```

113. Дата је бинарна датотека *mister.bin* која садржи целе бројеве. Имплементирати следеће потпрограме:

а) Функцију која чита из бинарне датотеке целе бројеве и од прочитаних бројева формира једноструко спрегнуту листу, али тако да сви елементи листе буду међусобно различити.

б) Функцију која као аргумент прихвата показивач на почетак већ формиране једноструко спрегнуте листе целих бројева (сви елементи листе су међусобно различити) и враћа разлику највећег и најмањег елемента листе. Ако је листа празна фукција треба да врати -1. У главном програму извршити дефиницију листе, позвати дефинисану функцију и на стандардном излазу приказати резултат у следећем формату: Разлика највећег и најмањег ел. листе је ___.

в) Функцију која као аргумент прихвата показивач на почетак већ формиране једноструко спрегнуте листе и најмањи елемент листе увећава за вредност највећег елемента листе.

114. Дата је бинарна датотека *mister.bin* која садржи целе бројеве. Имплементирати следеће потпрограме:

а) Функцију која чита из бинарне датотеке целе бројеве и од прочитаних бројева формира једноструко спрегнуту листу, али тако да се прочитани елемент из датотеке увек убацује на крај једноструко спрегнуте листе.

б) Функцију која као аргумент прихвата показивач на почетак већ формиране једноструко спрегнуте листе целих бројева и враћа разлику збира парних и збира непарних елемента листе. Ако је листа празна фукција не сме бити позвана. У главном програму извршити дефиницију листе, позвати дефинисану функцију и на стандардном излазу приказати резултат у следећем формату: Разлика збира парних и збира непарних елемената листе је ___.

в) Функцију која као аргумент прихвата показивач на почетак већ формиране једноструко спрегнуте листе и најмањи елемент листе увећава за разлику збира парних и збира непарних бројева. У случају да је разлика збира парних и непарних бројева негативна вредност, вредност елемент треба да остане непромењена.

115. Дата је бинарна датотека *prijemni.bin* у којој се налазе подаци о резултатима који су кандидати остварили на пријемном испиту. Структуру подака о кандидатима који су полагали пријемни испит чине следећа поља:
– *imePrezime* – низ карактера максималне дужине 50,
– *brPoenaSkola* – реалан број (double), максималан број поена је 40,
– *brPoenaPrijemniIspit* – реалан број (double), максималан број поена је 60,
– *smer* – низ карактера масималне дужине 4 (Могуће вредности су ISIT, MEN).
Пребацити све податке о кандидатима из датотеке у листу и приказати њен садржај. На основу ове листе формирати ранг листу за смер ISIT сортирану по укуппном броју бодова које имају кандидати. За чување ранг листе корисити низовну структуру. Сортирање се може вршити након формирања низа, а може се вршити и приликом формирања низа. На основу добијене ранг листе формирати извештај (у текстуалној датотеци) у којем се приказује број кандидата који су остварили максимални број поена на пријемном испиту.

116. Дата је бинарна датотека *prijemni.bin* у којој се налазе подаци о резултатима који су кандидати остварили на пријемном испиту. Структуру подака о кандидатима који су полагали пријемни испит чине следећа поља:
– *imePrezime* – низ карактера максималне дужине 50,
– *brPoenaSkola* – реалан број (double), максималан број поена је 40,
– *brPoenaPrijemniIspit* – реалан број (double), максималан број поена је 60,
– *smer* – низ карактера масималне дужине 4 (Могуће вредности су ISIT, MEN).
Пребацити све податке о кандидатима из датотеке у низ и приказати садржај низа. На основу низа формирати ранг листу за смер MEH сортирану по укуппном броју бодова које имају кандидати. За чување ранг листе корисити једноструко спрегнуту листу. Сортирање се може вршити након формирања листе, а може се вршити и приликом формирања саме листе. Приказати формирану ранг листу. На основу добијене ранг листе формирати извештај (у текстуалној датотеци) у којем се приказује број кандидата који су остварили максимални укупан број поена.

117. Дата је бинарна датотека *drzava.bin* у којој се налазе подаци о државама. Структуру подака о државама чине следећа поља:
– *naziv* – низ карактера максималне дужине 50,
– *povrsina* – реалан број (double),
– *brojStanovnika* – цео број.
Пребацити све податке о државама из датотеке у листу и приказати њен садржај. На основу листе формирати низ сортиран на основу густине насељености држава. Сортирање се може вршити након формирања низа, а може се вршити и приликом формирања самог низа. Приказати формирани низ. На основу добијеног низа формирати текстуалну датотеку која садржи податке о државама које имају велику густину насељености (више од 500 *становника/km*^{2})

118. Дата је бинарна датотека *drzava.bin* у којој се налазе подаци о државама. Структуру подака о државама чине следећа поља:
– *naziv* – низ карактера максималне дужине 50,
– *povrsina* – реалан број (double),
– *brojStanovnika* – цео број.
Пребацити све податке о државама из датотеке у низ и приказати садржај низа. На основу низа формирати листу сортирану на основу густине насељености држава. Сортирање се може вршити након формирања листе, а може се вршити и приликом формирања саме листе. Приказати формирану листу држава. На основу добијене листе формирати текстуалну датотеку која садржи информацију о броју држава које имају густину насељености већу од просека.

119. Дате су две бинарне датотеке. У датотеци *utakmica.dat* се налазе подаци о саставу тимова на датој утакмици. Структуру података о тимовима чине следећи подаци:

– *sifraIgraca* – цео број,
– *imePrezime* – низ карактера максималне дужине 25,
– *brojNaDresu* – цео број,
– *pozicija* – низ карактера максималне дужине 15 (плеј, бек, крило, центар...),
– *nazivTima* – низ карактера максималне дужине 20.

У другој датотеци *Statistika.dat* налазе се статистички подаци које је играч остварио на утакмици. Структуру података о статистици играча чине следећи подаци:

– *sifraIgraca* – цео број,
– *nazivProtivnickogTima* – низ карактера максималне дужине 20,
– *brPoena* – цео број,
– *brSkokova* – цео број,
– *brAsistencija* – цео број.

Имплементирати следеће потпрограме:

а) Написати функцију која учитава податке о играчима у низ и приказује га.

б) Написати функцију која учитава податке о статистичким показатељима у једноструко спрегнуту листу. Приказати редослед догађаја (односно упућених шутева) на утакмици.

ц) Формирати текстуалну датотеку *prikazStatistike.txt* која треба да прикаже просечне статистичке параметре за сваког играча као што је приказано у наставку, као и најбоље у свакој од категорија брПоена, брСкокова и брАсистенција. Прогласити МВП играча првенстава. МВП играч је онај који има најбољи просечни индекс корисности (рачуна се: prosecanBrPoena*0.8+prosecanBrSkokova*0.5+prosecanBrAsistencija*0.7).

prikazStatistike.txt

```
MVP:___________________________
Najbolji poenter: __________________
Najbolji skakac:__________________
Najbolji asistent: __________________
br. dresa    ime i prezime      pros. poena  pros.skokova   pros. asis    index
===============================================================================
```

120. Дате су две бинарне датотеке. У датотеци *utakmica.dat* се налазе подаци о саставу тимова на датој утакмици. Структуру података о тимовима чине следећи подаци:

– *sifraIgraca* – цео број,
– *imePrezime* – низ карактера максималне дужине 25,
– *brojNaDresu* – цео број,
– *pozicija* – низ карактера максималне дужине 15 (плеј, бек, крило, центар...),
– *nazivTima* – низ карактера максималне дужине 20.

Датотека је сортирана по шифри играча тако што првих 12 играча чине играче домаћег тима, а других 12 чине играчи гостујућег тима, тако да огрaчи домаћег тима имају шифре од 1 до 12, а гостујући од 13 до 24.

У другој датотеци *Dogadjaji.dat* налазе се подаци о битним "догађајима", односно упућеним шутевима. Структуру података о догађајима чине следећи подаци:

– *sifraIgraca* – цео број,
– *periodIgre* – цео број (може имати вредности 1,2, 3, 4 за четвртине и 5, 6... за продужетке),
– *minutPerioda* – цео број,
– *sekundPerioda* – цео број,
– *tipSuta* – цео број (може имати вредности 1,2, 3),
– *realizovano* – цео број (1 – реализовано, 2 – шут није реализован).

Датотека је сортирана по редоследу дешавања догађаја на утакмици.

Имплементирати следеће потпрограме:

а) Написати функцију која учитава податке о играчима у низ и приказује га.
б) Написати функцију која учитава податке о статистичким показатељима у једноструко спрегнуту листу, тако што догађаје додаје на крај листе. Приказати редослед догађаја (односно упућених шутева) на утакмици.
ц) Формирати текстуалну датотеку *izvestaj.txt* која треба да прикаже резултат утакмице по четвртинама као и статистичке параметре за сваког од играча.
Пример извештаја:

```
                    IZVESTAJ SA UTAKMICE

              Partizan 101:78   Crvena Zvezda
                    I  II III  IV
                   28  24  19  30
                   17  26  15  20


                                Partizan
br.dresa                     Igrac 1 poen 2 poena 3 poena
================================================================================
       4      Cvetkovic Aleksandar   0/  1   1/  2   1/  2
       5          Aranitovic Petar   2/  2   2/  2   0/  0
       6        Andrew Jones Kevin   1/  1   1/  3   2/  3
       8                 Muric Edo   1/  2   5/  5   2/  3
       9          Marinkovic Vanja   1/  2   3/  6   1/  4
      12         Velickovic Novica   0/  2   1/  1   0/  2
      13          Vitkovac Cedomir   0/  1   3/  3   0/  5
       4               Djumic Bozo   0/  1   1/  2   1/  4
      22       Milutinovic Andrija   1/  2   2/  3   1/  4
      25          Williams Darrell   1/  4   3/  3   1/  3
      27               Vrabac Adin   0/  3   0/  1   1/  6
      31              Wilson Jamar   1/  1   3/  3   3/  3


                              Crvena Zvezda
br.dresa                     Igrac 1 poen 2 poena 3 poena
================================================================================
       1    Anthony Kinsey Tarence   1/  2   0/  0   1/  2
       4               Rebic Nikola   2/  3   1/  3   0/  3
       6          Dangubic Nemanja   2/  3   4/  4   1/  3
       9             Mitrovic Luka   0/  3   2/  3   1/  2
      10              Lazic Branko   1/  2   2/  4   2/  4
      13            Micic Vasilije   1/  2   1/  1   2/  4
      19           Simonovic Marko   1/  1   2/  3   0/  4
      23             Guduric Marko   0/  0   2/  3   0/  3
      24              Jovic Stefan   1/  2   1/  1   2/  2
      30             Quincy Miller   1/  1   0/  2   2/  5
      33               Zibres Maik   1/  2   0/  0   0/  2
      51                     Stimac   0/  1   1/  4   1/  3
```

121. У једној земљи Дембелији говори се дембелијски језик који има врло једноставна лексичка правила:
– Свака реч се састоји од једног или више слогова,
– Сваки слог се састоји од једног самогласника и једног сугласника (искључиво, али није битан редослед да ли прво самогласник па сугласник или обрнуто),
– Постоје везници а, е, и , о.
На основу ових правила написати примитиван спелчекер (spellchecker) за дембелијски језик. Спелчекер треба да на основу текста који се задаје у улазној датотеци *ulaz.txt* на стандардном излазу испише све неправилне речи оним редоследом како се појављују у тексту. Све речи се у улазној датотеци задају у једној линији.
Тестни пример:
ulaz.txt
1a ann anghjka mame i tatemi o dede
Излаз:
1a (нема сугласника)
ann (нема самогласника у другом слогу)
anghjka (други слог нема самогласник)

4. Пример испитног рока

У овом одељку дат је пример испитног рока. Најпре се наводи задатак, а затим наводи решење посматраног задатка.

122. Дата је улазна текстуална датотека *matrica.txt* у којој се чувају елементи матрице целобројног типа. У првом реду текстуалне улазне датотеке чува се димензија матрице у формату nXm (нпр. 2X3), након чега датотека садржи онолико редова колико има колона матрице (сваки ред улазне текстуалне датотеке представља елементе те колоне матрице). Максимална димензија улазне матрице може бити 10X10.
Пример улазне текстуалне датотеке:
2X3
12 22
12 34
28 22
Матрица:
12 12 28
22 34 22

а) Написати функцију која елементе матрице који је чувају у улазној текстуалној датотеци *matrica.txt* пребацује у матрицу. Написати функцију која приказује тако формирану матрицу.

```
#include <stdio.h>
#include <stdlib.h>
#define FAJL "matrica.txt"
typedef int TMATRICA[10][10];
void prikazi_matricu(int n, int m, TMATRICA mat) {
	for (int i = 0; i < n; i++) {
		for (int j = 0; j < m; j++) {
			printf("%5d", mat[i][j]);
		}
		printf("\n");
	}
}
void formiraj_matricu(int *n, int *m, TMATRICA mat) {
	int i, j;
	FILE *datoteka = fopen(FAJL, "r");
	if (datoteka == NULL) {
		printf("\nDatoteka ne postoji");
	}
	else {
		fscanf(datoteka, "%dX%d\n", n, m);
		for (j = 0; j < *m; j++) {
			for (i = 0; i < *n; i++) {
				fscanf(datoteka, "%d ", &mat[i][j]);
			}
		}
		fclose(datoteka);
	}
}
```

б) Написати функцију која на основу улазне матрице произвољне димензије (максимална димензија матрице може бити 10X10) формира једноструко спрегнуту листу која садржи:
– Онолико елемената колико има различитих елемената у улазној матрици,
– Број појављивања сваког броја који се налази у улазној матрици.
Написати функцију која приказује садржај тако формиране листе.

```
typedef struct cvor CVOR;
```

```
typedef CVOR* PCVOR;
struct cvor {
	int info;
	int frek;
	PCVOR sledeci;
};
int frekvencija(int niz[], int n, int broj) {
	int i;
	int brojac = 0;
	for (i = 0; i < n; i++) {
		if (niz[i] == broj) {
			brojac++;
		}
	}
	return brojac;
}
void formiraj_niz(int n, int m, TMATRICA mat, int niz[], int *brel) {
	int i, j;
	for (i = 0; i < n; i++) {
		for (j = 0; j < m; j++) {
			niz[*brel] = mat[i][j];
			*brel = *brel + 1;
		}
	}
}
void ubaci_na_pocetak(PCVOR* glava, int broj, int frek) {
	PCVOR novi;
	novi = malloc(sizeof(CVOR));
	novi->info = broj;
	novi->frek = frek;
	novi->sledeci = *glava;
	*glava = novi;
}
void formiraj_listu(int n, int m, TMATRICA mat, PCVOR *glava) {
	int i, obradjen, frek;
	int niz[100];
	int brel = 0;
	formiraj_niz(n, m, mat, niz, &brel);
	for (i = 0; i < brel; i++) {
		obradjen = frekvencija(niz, i, niz[i]);
		if (obradjen == 0) {
			frek = frekvencija(niz, brel, niz[i]);
			ubaci_na_pocetak(glava, niz[i], frek);
		}
	}
}
void prikazi_listu(PCVOR glava) {
	PCVOR tekuci;
	tekuci = glava;
	printf("\n*** Sadrzaj liste ***\n");
	while (tekuci != NULL) {
		printf("Element: %d, Frekvencija: %d\n", tekuci->info, tekuci->frek);
		tekuci = tekuci->sledeci;
	}
}
```

в) У главном програму учитати матрицу из датотеке и приказати њен садржај. Након тога формирати једноструко спрегнуту листу и приказати садржај листе. Искористити претходно имплементиране потпрограме.

```
int main(void) {
	TMATRICA mat;
	PCVOR glava = NULL;
	int n = 0;
	int m = 0;
	formiraj_matricu(&n, &m, mat);
	prikazi_matricu(n, m, mat);
	formiraj_listu(n, m, mat, &glava);
	prikazi_listu(glava);
	return 0;
}
```

123. Имплементирати функцију која проверава да ли је број прост.

```
#include <stdio.h>
int prost_broj(int broj) {
	int i;
	for (i = 2; i <= broj / 2; i++) {
		if (broj % i == 0) {
			return 0;
		}
	}
	return 1;
}
int main(void) {
	int broj;
	int prost;
	do {
		printf("Uneste broj: ");
		scanf("%d", &broj);
	} while (broj <= 1);
	prost = prost_broj(broj);
	if (prost == 0) {
		printf("Broj %d nije prost\n", broj);
	}
	else {
		printf("Broj %d je prost\n", broj);
	}
	return 0;
}
```

124. Имплементирати функцију која на основу унете 4 цифре формира четвороцифрени број A на следећи начин:

– Прва унета цифра представља цифру хиљаде, друга цифру стотине, трећа цифру десетице и четврта цифру јединице.

– Цифре могу да се понављају.

– Цифре морају бити у интервалу од 1 до 5. У случају да је корисник унео цифру која није у интервалу од 1 до 5 поновити унос. Максималан број покушаја за унос сваке цифре је 4. Уколико корисник не унесе исправно цифру из четири покушаја прекинути унос броја.

```
#include <stdio.h>
#define MAX_BR_POKUSAJA 4
int unos_cifre(void) {
	int cifra;
	int brPokusaja = MAX_BR_POKUSAJA;
	while (brPokusaja > 0) {
		printf("Preostali broj pokusaja: %d\n", brPokusaja);
		printf("Unesite cifru:");
		scanf("%d", &cifra);
		if (cifra >= 1 && cifra <= 5) {
			return cifra;
```

```
            }
            brPokusaja--;
      }
      return -1;
}
int unos_broja(void) {
      int broj = 0;
      int uneta_cifra;
      for (int i = 1; i <= 1000; i = i * 10) {
            printf("Unos cifre %d\n", i);
            uneta_cifra = unos_cifre();
            if (uneta_cifra == -1) {
                  printf("Niste dobro uneli cifru!\n");
                  break;
            }
            broj = broj + uneta_cifra * i;
      }
      return (broj < 1000) ? -1 : broj;
}
int main(void) {
      int broj = unos_broja();
      if (broj != -1) {
            printf("Unet je broj %d\n", broj);
      }
      return 0;
}
```

125. Имплементирати функцију која прихвата два четвороцифрена броја (А и Б) и која проверава да ли се цифре броја Б поклапају по редоследу са цифрама броја А и приказује поруку у следећем формату: број погођених цифара које су на месту, број погођених цифара које нису на месту (слично игри „Скочко“ у „Слагалици“). Четвороцифрени бројеви А и Б садрже само цифре у интервалу од 1 до 5.

Тестни пример 1:

А = 1224

Б = 2221

Порука: 2 на месту, 1 није на месту

Тестни пример 2:

А = 1115

Б = 2221

Порука: 0 на месту, 1 није на месту

```
#include <stdio.h>
#define MAX_BR_POKUSAJA 4
int unos_cifre(void) {
      int cifra;
      int brPokusaja = MAX_BR_POKUSAJA;
      while (brPokusaja > 0) {
            printf("Unesite cifru:");
            scanf("%d", &cifra);
            if (cifra >= 1 && cifra <= 5) {
                  return cifra;
            }
            brPokusaja--;
      }
      return -1;
}
int unos_broja(void) {
      int broj = 0;
```

```
	int uneta_cifra;
	for (int i = 1; i <= 1000; i = i * 10) {
		printf("Unos cifre %d\n", i);
		uneta_cifra = unos_cifre();
		if (uneta_cifra == -1) {
			printf("Niste dobro uneli cifru!\n");
			break;
		}
		broj = broj + uneta_cifra * i;
	}
	return (broj < 1000) ? -1 : broj;
}
void pretvori_u_niz(int broj, int niz[]) {
	int pozicija = 3;
	while (broj > 0) {
		niz[pozicija] = broj % 10;
		broj = broj / 10;
		pozicija--;
	}
}
int na_mestu(int resenje[], int pokusaj[]) {
	int br = 0;
	for (int i = 0; i < 4; i++) {
		if (resenje[i] == pokusaj[i]) {
			br++;
		}
	}
	return br;
}
int nije_na_mestu(int resenje[], int pokusaj[]) {
	int br = 0;
	for (int i = 0; i < 4; i++) {
		//ukoliko je broj na mestu, treba preci na sledeci
		if (pokusaj[i] == resenje[i]) {
			continue;
		}
		for (int j = 0; j < 4; j++) {
			//nije na mestu ukoliko broj nije pogodjen i ako su brojevi jednaki
			if (pokusaj[j] != resenje[j] && resenje[j] == pokusaj[i]) {
				br++;
				resenje[j] = -1;
				break;
			}
		}
	}
	return br;
}
void skocko(int resenje, int pokusaj) {
	int resenje_niz[4];
	int pokusaj_niz[4];
	int br_nije_na_mestu = 0;
	int br_na_mestu = 0;
	pretvori_u_niz(resenje, resenje_niz);
	pretvori_u_niz(pokusaj, pokusaj_niz);
	br_na_mestu = na_mestu(resenje_niz, pokusaj_niz);
	br_nije_na_mestu = nije_na_mestu(resenje_niz, pokusaj_niz);
	printf("\n%d na mestu, %d nije na mestu\n", br_na_mestu, br_nije_na_mestu);
}
int main(void) {
```

```
	printf("Unesi resenje: \n");
	int resenje = unos_broja();
	printf("Pogadjaj:\n");
	int pokusaj = unos_broja();
	skocko(resenje, pokusaj);
	return 0;
}
```

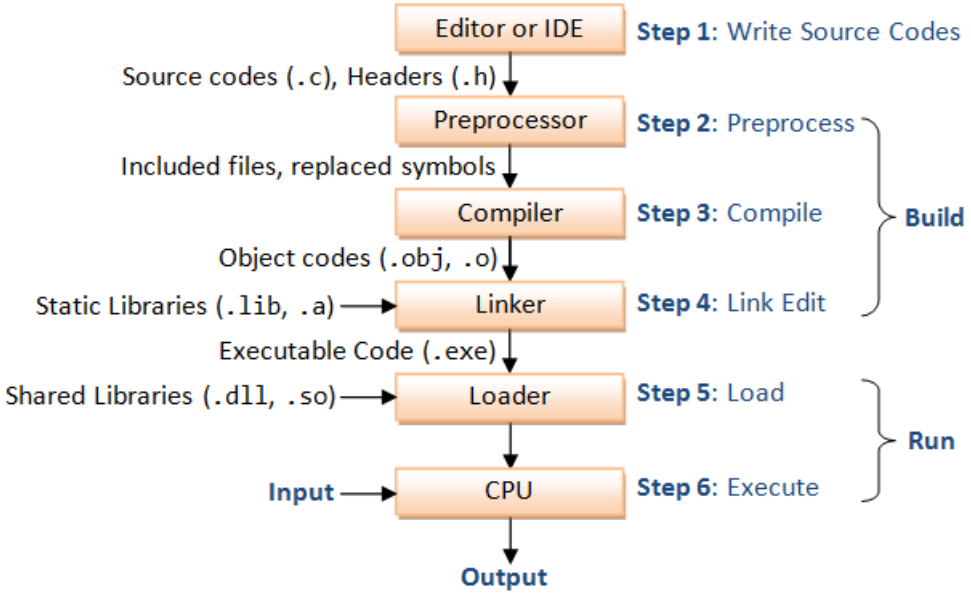
126. Палиндромски број је број који се исто чита са обе стране. Имплементирати функцију за унос бројева све док се не унесе нула и одредити највећи палиндромски број међу њима или приказати поруку да палиндромски број не постоји.

```
#include <stdio.h>
int palindrom(int broj) {
	int obrnut = 0;
	int pom = broj;
	int cifra;
	while (pom > 0) {
		cifra = pom % 10;
		obrnut = obrnut * 10 + cifra;
		pom /= 10;
	}
	return broj == obrnut;
}
void unos_brojeva(int brojevi[], int *n) {
	int broj;
	do {
		printf("Unesi broj: ");
		scanf("%d", &broj);
		brojevi[*n] = broj;
		*n = *n + 1;
	} while (broj != 0);
	*n = *n - 1;
}
int max_palindrom(int brojevi[], int n) {
	int max = -1;
	for (int i = 0; i < n; i++) {
		if (max < brojevi[i] && palindrom(brojevi[i])) {
			max = brojevi[i];
		}
	}
	return max;
}
int main(void) {
	int brojevi[10];
	int n = 0;
	unos_brojeva(brojevi, &n);
	int max = max_palindrom(brojevi, n);
	if (max > -1) {
		printf("Najveci palindromski broj je %d.\n", max);
	}
	else {
		printf("Ne postoji palindromski broj!\n");
	}
	return 0;
}
```

5. Прилози

У овом одељку налазе се следећи прилози:
– Фазе процеса израде програма
– Коришћење интегрисаног развојног окружења *Visual Studio 2010*
– Структура заглавља програма
– Процес израде програма – од монолитног до структурираног програма
– Процес израде програма – израда корисничког менија

5.1. Фазе процеса израде програма



Слика 1. Фазе процеса израде програма

5.2. Коришћење интегрисаног развојног окружења *Visual Studio 2010*

Погледати датотеку Креирање C програма у VS2010.

5.3. Структура заглавља програма

Датотека: *Pr_3. ZaglavljePrograma.c*

```
/*
** PROJEKAT : <Naziv projekta. Npr: Osnove programiranja u programskom jeziku C>
** DATOTEKA : <Naziv datoteke. Npr. P1_promenljive.c ili P1_promenljive.h>
** OPIS     : <Opis sadržaja datoteke ili verbalna formulacija zahteva.
**            Npr: Napisati program u programskom jeziku C koji
**            prikazuje globalne i lokalne promenljive.>
** DATUM    : <Datum kreiranja datoteke. Npr: 24.03.2016.>
** AUTOR    : <Ime i prezime autora programskog koda; može i e-mail adresa.>
** PROMENE  :
**   21.07.2017. - promenjen je prototip f-je za prikaz rezultata (SDL)
**   xx.xx.xxxx. - <opis promene> (<programer>)
**
** Copyright (C) FON-LSI, 2016.
*/
```

5.4. Процес израде програма – од монолитног до структурираног програма

Верзија 1

Датотека: *P02_01_krug.c*

```
/*
** PROJEKAT : Osnove programiranja u programskom jeziku C
** DATOTEKA : P02_01_krug.c
** OPIS     : Napisati program u programskom jeziku C koji na standardnom
**            izlaznom uredjaju
**            prikazuje površine i obime krugova ciji su poluprecnici zadati:
**            11.1, 12.2, 13.3, 14.4, 15.5 .
** DATUM    : 26.02.2016.
** AUTOR    : S.D.L.
** PROMENE  :
**  xx.xx.xxxx. - <opis promene> (<programer>)
**
** Copyright (C) FON-LSI, 2016.
*/

#include <stdio.h>

#define      PI   3.1415926536

const double pi = 3.1415926536;

double pp1, pp2 = 12.2, pp3, pp4, pp5;
double pv1, pv2, pv3, pv4, pv5,
       ob1, ob2, ob3, ob4, ob5;

int  main(void)
{
       printf(">>> P02_01_krug.c  <<<\n\n");
       // Dodeljivanje vrednosti poluprecnicima: 11.1, 12.2, 13.3, 14.4, 15.5
       pp1 = 11.10000;
       pp5 = 1.1 + (pp4 = (1.1 + (pp3 = 13.3)));
       // Izracunavanje površine
       pv1 = pp1 * pp1 * pi;
       pv2 = pp2 * pp2 * pi;
       pv3 = pp3 * pp3 * pi;
       pv4 = pp4 * pp4 * pi;
       pv5 = pp5 * pp5 * pi;
       // Izracunavanje obima
       ob1 = 2 * pp1 * PI;
       ob2 = 2 * pp2 * PI;
       ob3 = 2 * pp3 * PI;
       ob4 = 2 * pp4 * PI;
       ob5 = 2 * pp5 * PI;
       // Prikaz dobijenih rezultata
       printf(" POVRSINE I OBIMI KRUGOVA\n\n");
       printf(" Poluprecnik     Povrsina     Obim\n"
              "=================================\n");
       printf("%12.1lf %12.3lf %8.3lf\n", pp1, pv1, ob1);
       printf("%12.1lf %12.3lf %8.3lf\n", pp2, pv2, ob2);
       printf("%12.1lf %12.3lf %8.3lf\n", pp3, pv3, ob3);
       printf("%12.1lf %12.3lf %8.3lf\n", pp4, pv4, ob4);
       printf("%12.1lf %12.3lf %8.3lf\n", pp5, pv5, ob5);
       return 0;
}
```

Верзија 2

Датотека: *P02_02_krug.c*

```
/*
** PROJEKAT : Osnove programiranja u programskom jeziku C
** DATOTEKA : P02_02_krug.c
** OPIS     : Napisati program u programskom jeziku C koji na standardnom
**            izlaznom uredjaju
**            prikazuje površine i obime krugova ciji su poluprecnici zadati
**            (11.1, 12.2, 13.3, 14.4, 15.5)
**            upotrebom funkcija.
** DATUM    : 26.02.2016.
** AUTOR    : S.D.L.
** PROMENE  :
**  xx.xx.xxxx. - <opis promene> (<programer>)
**
** Copyright (C) FON-LSI, 2016.
*/

#include <stdio.h>

#define      PI   3.1415926536

const double pi = 3.1415926536;

double pp1, pp2 = 12.2, pp3, pp4, pp5;
double pv1, pv2, pv3, pv4, pv5,
       ob1, ob2, ob3, ob4, ob5;

void IzracunajPovrsinu(void) {
      pv1 = pp1 * pp1 * pi;
      pv2 = pp2 * pp2 * pi;
      pv3 = pp3 * pp3 * pi;
      pv4 = pp4 * pp4 * pi;
      pv5 = pp5 * pp5 * pi;
}

void IzracunajObim(void) {
      ob1 = 2 * pp1 * PI;
      ob2 = 2 * pp2 * PI;
      ob3 = 2 * pp3 * PI;
      ob4 = 2 * pp4 * PI;
      ob5 = 2 * pp5 * PI;
}

void PrikazZaglavlja(void) {
     printf(" POVRSINE I OBIMI KRUGOVA\n\n");
     printf(" Poluprecnik     Povrsina     Obim\n"
            "===================================\n");
}

void PrikazRezultata(void) {
      printf("%12.1lf %12.3lf %8.3lf\n", pp1, pv1, ob1);
      printf("%12.1lf %12.3lf %8.3lf\n", pp2, pv2, ob2);
      printf("%12.1lf %12.3lf %8.3lf\n", pp3, pv3, ob3);
      printf("%12.1lf %12.3lf %8.3lf\n", pp4, pv4, ob4);
      printf("%12.1lf %12.3lf %8.3lf\n", pp5, pv5, ob5);
}
```

```
int  main(void)
{
	printf(">>> P02_02_krug.c  <<<\n\n");
	// Dodeljivanje vrednosti poluprecnicima: 11.1, 12.2, 13.3, 14.4, 15.5
	pp1 = 11.10000;
	pp5 = 1.1 + (pp4 = (1.1 + (pp3 = 13.3)));
	// Izracunavanje povrsine
	IzracunajPovrsinu();
	// Izracunavanje obima
	IzracunajObim();
	// Prikaz dobijenih rezultata
	PrikazZaglavlja();
	PrikazRezultata();

	return 0;
}
```

Верзија 3

Датотека: *P02_03_krug.c*

```
/*
** PROJEKAT : Osnove programiranja u programskom jeziku C
** DATOTEKA : P02_03_krug.c
** OPIS     : Napisati program u programskom jeziku C koji na standardnom
**            izlaznom uredjaju
**            prikazuje površine i obime krugova ciji su poluprecnici zadati
**            (11.1, 12.2, 13.3, 14.4, 15.5)
**            upotrebom funkcija i funkcijskih prototipova.
** DATUM    : 26.02.2016.
** AUTOR    : S.D.L.
** PROMENE  :
**  xx.xx.xxxx. - <opis promene> (<programer>)
**
** Copyright (C) FON-LSI, 2016.
*/

#include <stdio.h>

#define      PI   3.1415926536

const double pi = 3.1415926536;

double pp1 = 11.1, pp2 = 12.2, pp3 = 13.3, pp4 = 14.4, pp5 = 15.5;
double pv1, pv2, pv3, pv4, pv5,
       ob1, ob2, ob3, ob4, ob5;

void IzracunajPovrsinu(void);
void IzracunajObim(void);
void PrikazZaglavlja(void);
void PrikazRezultata(void);

int  main(void)
{
	printf(">>> P02_03_krug.c  <<<\n\n");
	IzracunajPovrsinu();
	IzracunajObim();
	PrikazZaglavlja();
	PrikazRezultata();
```

```
	return 0;
}

void IzracunajPovrsinu(void) {
	pv1 = pp1 * pp1 * pi;
	pv2 = pp2 * pp2 * pi;
	pv3 = pp3 * pp3 * pi;
	pv4 = pp4 * pp4 * pi;
	pv5 = pp5 * pp5 * pi;
}

void IzracunajObim(void) {
	ob1 = 2 * pp1 * PI;
	ob2 = 2 * pp2 * PI;
	ob3 = 2 * pp3 * PI;
	ob4 = 2 * pp4 * PI;
	ob5 = 2 * pp5 * PI;
}

void PrikazZaglavlja(void) {
	printf(" POVRSINE I OBIMI KRUGOVA\n\n");
	printf(" Poluprecnik     Povrsina     Obim\n"
		"==================================\n");
}

void PrikazRezultata(void) {
	printf("%12.1lf %12.3lf %8.3lf\n", pp1, pv1, ob1);
	printf("%12.1lf %12.3lf %8.3lf\n", pp2, pv2, ob2);
	printf("%12.1lf %12.3lf %8.3lf\n", pp3, pv3, ob3);
	printf("%12.1lf %12.3lf %8.3lf\n", pp4, pv4, ob4);
	printf("%12.1lf %12.3lf %8.3lf\n", pp5, pv5, ob5);
}
```

Верзија 4

Датотека: *P02_04_krug.c*

```
/*
** PROJEKAT : Osnove programiranja u programskom jeziku C
** DATOTEKA : P02_04_krug.c
** OPIS     : Napisati program u programskom jeziku C koji na standardnom
**            izlaznom uredjaju
**            prikazuje površine i obime krugova ciji su poluprecnici zadati
**            (11.1, 12.2, 13.3, 14.4, 15.5)
**            upotrebom funkcija, funkcijskih prototipova i pomocnih funkcija
**            za izracunavanje povrsine i obima.
** DATUM    : 26.02.2016.
** AUTOR    : S.D.L.
** PROMENE  :
**  xx.xx.xxxx. - <opis promene> (<programer>)
**
** Copyright (C) FON-LSI, 2016.
*/

#include <stdio.h>

#define PI    3.1415926536

double  pp1 = 11.1, pp2 = 12.2, pp3 = 13.3, pp4 = 14.4, pp5 = 15.5;
double  pv1, pv2, pv3, pv4, pv5,
```

```
        ob1, ob2, ob3, ob4, ob5;

void IzracunajPovrsinu(void);
void IzracunajObim(void);
void PrikazZaglavlja(void);
void PrikazRezultata(void);

int  main(void)
{
       printf(">>> P02_04_krug.c  <<<\n\n");
       IzracunajPovrsinu();
       IzracunajObim();
       PrikazZaglavlja();
       PrikazRezultata();
       return 0;
}

////////////////////////////////////////////////////////
double PovKruga(double pp) {
       double pov;
       pov = pp * pp * PI;
       return pov;
}

void IzracunajPovrsinu(void) {
       pv1 = PovKruga(pp1);
       pv2 = PovKruga(pp2);
       pv3 = PovKruga(pp3);
       pv4 = PovKruga(pp4);
       pv5 = PovKruga(pp5);
}
////////////////////////////////////////////////////////

double ObKruga(double pp) {
       return 2 * pp * PI;
}

void IzracunajObim(void) {
       ob1 = ObKruga(pp1);
       ob2 = ObKruga(pp2);
       ob3 = ObKruga(pp3);
       ob4 = ObKruga(pp4);
       ob5 = ObKruga(pp5);
}
////////////////////////////////////////////////////////

void PrikazZaglavlja(void) {
       printf(" POVRSINE I OBIMI KRUGOVA\n\n");
       printf(" Poluprecnik     Povrsina     Obim\n"
              "===================================\n");
}

void PrikazRezultata(void) {
       printf("%12.1lf %12.3lf %8.3lf\n", pp1, pv1, ob1);
       printf("%12.1lf %12.3lf %8.3lf\n", pp2, pv2, ob2);
       printf("%12.1lf %12.3lf %8.3lf\n", pp3, pv3, ob3);
       printf("%12.1lf %12.3lf %8.3lf\n", pp4, pv4, ob4);
       printf("%12.1lf %12.3lf %8.3lf\n", pp5, pv5, ob5);
}
```

Верзија 5

Датотека: *P02_05_krug.c*

```
/*
** PROJEKAT : Osnove programiranja u programskom jeziku C
** DATOTEKA : P02_05_krug.c
** OPIS     : Napisati program u programskom jeziku C koji na standardnom
**            izlaznom uredjaju
**            prikazuje povrsšine i obime krugova ciji su poluprecnici zadati
**            (11.1, 12.2, 13.3, 14.4, 15.5)
**            upotrebom funkcija, funkcijskih prototipova i pomocnih funkcija
**            za izracunavanje povrsine i obima.
**            Umesto skalara (promenljivih skalarnog tipa) koristiti
**            nizove skalara (promenljive tipa niz skalara).
** DATUM    : 26.02.2016.
** AUTOR    : S.D.L.
** PROMENE  :
**  xx.xx.xxxx. - <opis promene> (<programer>)
**
** Copyright (C) FON-LSI, 2016.
*/

#include <stdio.h>

#define PI    3.1415926536

double  pp[5] = {11.1, 12.2, 13.3, 14.4, 15.5};
double  pv[5], ob[5];

void IzracunajPovrsinu(void);
void IzracunajObim(void);
void PrikazZaglavlja(void);
void PrikazRezultata(void);

int  main(void)
{
       printf(">>> P02_05_krug.c  <<<\n\n");
       IzracunajPovrsinu();
       IzracunajObim();
       PrikazZaglavlja();
       PrikazRezultata();
       return 0;
}

////////////////////////////////////////////////////////
double PovKruga(double pp) {
      double pov;
      pov = pp * pp * PI;
      return pov;
}

void IzracunajPovrsinu(void) {
       int i;
       for (i = 0; i<=4; i = i+1)
              pv[i] = PovKruga(pp[i]);
}
////////////////////////////////////////////////////////
```

```
double ObKruga(double pp) {
	return 2 * pp * PI;
}

void IzracunajObim(void) {
	for (int i = 0; i<=4; i++)
		ob[i] = ObKruga(pp[i]);
}
//////////////////////////////////////////////////////

void PrikazZaglavlja(void) {
	printf(" POVRSINE I OBIMI KRUGOVA\n\n");
	printf(" Poluprecnik	Povrsina	Obim\n"
		"==================================\n");
}

void PrikazRezultata(void) {
	for (int i = 0; i<=4; i++)
	printf("%12.1lf %12.3lf %8.3lf\n", pp[i], pv[i], ob[i]);
}
```

Верзија 6

Датотека: *P02_06_krug.c*

```
/*
** PROJEKAT : Osnove programiranja u programskom jeziku C
** DATOTEKA : P02_06_krug.c
** OPIS     : Napisati program u programskom jeziku C koji na standardnom
**            izlaznom uredjaju
**            prikazuje povrsšine i obime krugova ciji su poluprecnici zadati
**            (11.1, 12.2, 13.3, 14.4, 15.5)
**            upotrebom funkcija, funkcijskih prototipova i pomocnih funkcija
**            za izracunavanje povrsine i obima.
**            Umesto skalara (promenljivih skalarnog tipa) koristiti
**            nizove skalara (promenljive tipa niz skalara).
**            Program podeliti u tri datoteke:
**              1) P02_06_krug.c - sadrzi glavnu f-ju main()
**              2) P02_krug.h –
**                 sadrzi specifikaciju operacije koje se pozivaju iz glavne f-je
**              3) P02_krug.c –
**                 sadrzi implementaciju operacija koje su specificirane u *.h datoteci
** DATUM    : 26.02.2016.
** AUTOR    : S.D.L.
** PROMENE  :
**  xx.xx.xxxx. - <opis promene> (<programer>)
**
** Copyright (C) FON-LSI, 2016.
*/

//#include <stdio.h>
#include "P02_06_krug_spec.h"

double pp[5] = {11.1, 12.2, 13.3, 14.4, 15.5};
char * poruka = "P02_06_krug.c";

int main(void) {
	double pv[5], ob[5];
	Info(poruka);
	IzracunajPovrsinu(pp, pv);
```

```
	IzracunajObim(pp, ob);
	PrikazZaglavlja();
	PrikazRezultata(pp, pv, ob);
	return 0;
}
```

Датотека: *P02_06_krug_spec.h*

```
// P02_06_krug_spec.h  :: Specifikacija operacija

#ifndef _P02_06_KRUG_SPEC_H
#define _P02_06_KRUG_SPEC_H

void Info(char *);
void IzracunajPovrsinu(const double[], double[]);
void IzracunajObim(const double[], double[]);
void PrikazZaglavlja(void);
void PrikazRezultata(const double[], const double[], const double[]);

#endif /* _P02_06_KRUG_SPEC_H */
```

Датотека: *P02_06_krug_impl.c*

```
// P02_krug.c  ::  Implementacija operacija

#include <stdio.h>
#include "P02_06_krug_spec.h"

#define PI  3.1415926536

//== Specifikacija internih funkcija ================================================

double PovKruga(double pp);
double ObKruga(double pp);

//================================================================================ Kraj ==

//== Implementacija eksternih funkcija ==============================================

inline void Info(char * txt) {
	printf("\n>>>  %s  <<<\n\n", txt);
}

void IzracunajPovrsinu(const double pp[], double pv[]) {
	unsigned short int i;
	// pp[0] = 10;            // Prijavljuje se greska, jer je: const double pp[]
	for (i = 0; i <= 4; i = i + 1)
		pv[i] = PovKruga(pp[i]);
}

void IzracunajObim(const double pp[], double ob[]) {
	for (size_t i = 0; i <= 4; i++)
		ob[i] = ObKruga(pp[i]);
}

void PrikazZaglavlja(void) {
	printf(" POVRSINE I OBIMI KRUGOVA\n\n");
	printf(" Poluprecnik     Povrsina     Obim\n"
		"==================================\n");
}
```

```
void PrikazRezultata(const double pp[], const double pv[], const double ob[]) {
	for (size_t i = 0; i <= 4; i++)
	printf("%12.1lf %12.3lf %8.3lf\n", pp[i], pv[i], ob[i]);
}
//======================================================================== Kraj ==

//== Implementacija internih funkcija ============================================

inline double PovKruga(double pp) {
	double pov;
	pov = pp * pp * PI;
	return pov;
}

inline double ObKruga(double pp) {
	return 2 * pp * PI;
}
//======================================================================== Kraj ==
```

5.5. Процес израде програма – израда корисничког менија

Верзија 1

Датотека: *meni_01.c*

```
/*
** PROJEKAT : Osnove programiranja u programskom jeziku C
** DATOTEKA : meni_01.c
** OPIS     : Napisati program u programskom jeziku C koji
**            implementira konzolni korisnički interfejs
**            upotrebom menija.
**            V.1
** DATUM    : 26.02.2016.
** AUTOR    : S.D.L.
** PROMENE  :
**  xx.xx.xxxx. - <opis promene> (<programer>)
**
** Copyright (C) FON-LSI, 2016.
*/

/*

========
STUDENT
========

0. Kraj rada

1. Ubaci     (Insert)
2. Izbaci    (Delete)
3. Promeni   (Update)
4. Prikazi   (Select)

	4.0. Kraj (povratak u prethodni meni)

	4.1. Prikazi sve
	4.2. Prikazi po broju indeksa (BI)
	4.3. Prikazi po prezimenu
```

```
*/

#include <stdio.h>
#include <conio.h>

#define KRAJ 0

int  GlavniMeni(void);
void Izvrsi(const int);
void Odjava(void);
void UbaciStud(void);
void IzbaciStud(void);
void PromeniStud(void);

int  PrikaziMeni(void);
void IzvrsiPrikaz(const int);
void PrikaziStud(void);
void PrikaziSve(void);
void PrikaziPoBI(void);
void PrikaziPoPrez(void);

int main(void){
	int izbor;

	do {
		izbor = GlavniMeni();
		Izvrsi(izbor);
	} while (izbor != KRAJ);

	return 0;
}

int GlavniMeni(void) {
	clrscr();
	printf("\n=============\n STUDENT\n=============");
	printf("\n GLAVNI MENI\n");
	printf("\n 0. Kraj rada\n");
	printf("\n 1. Ubaci");
	printf("\n 2. Izbaci");
	printf("\n 3. Promeni");
	printf("\n 4. Prikazi");
	int iz;
	do {
		printf("\n\nVas izbor (0 - 4): ");
		scanf("%d", &iz);
	} while ((iz < 0) || (iz > 4));
	return iz;
}

void Izvrsi (const int iz) {
	switch (iz) {
		case 0:	Odjava();	break;
		case 1:	UbaciStud();	break;
		case 2:	IzbaciStud();	break;
		case 3:	PromeniStud();	break;
		case 4:	PrikaziStud();	break;
		default: printf("\nUneli ste neodgovarajuci broj.");
	}
}
```

```
void Odjava(void) {
	printf("\nKRAJ RADA\n\n");
}

void UbaciStud(void) {
	printf("\n\nRadi se... UBACI\n");
	getch();
}

void IzbaciStud(void) {
	printf("\n\nRadi se... IZBACI\n");
	getch();
}

void PromeniStud(void) {
	printf("\n\nRadi se... PROMENI\n");
	getch();
}

void PrikaziStud(void) {
	int izborPrikaza;
	do {
		izborPrikaza = PrikaziMeni();
		IzvrsiPrikaz(izborPrikaza);
	} while (izborPrikaza != KRAJ);
}

int PrikaziMeni(void) {
	clrscr();
	printf("\n======================\nSTUDENT => PRIKAZI\n======================");
	printf("\n M E N I\n");
	printf("\n 0. Kraj\n");
	printf("\n 1. Prikazi sve");
	printf("\n 2. Prikazi po broju indeksa");
	printf("\n 3. Prikazi po prezimenu");
	int iz;
	do {
		printf("\n\nVas izbor (0 - 3): ");
		scanf("%d", &iz);
	} while ((iz < 0) || (iz > 3));
	return iz;
}

void IzvrsiPrikaz (const int iz) {
	switch (iz) {
		case 0:	break;
		case 1:	PrikaziSve();  break;
		case 2:	PrikaziPoBI(); break;
		case 3:	PrikaziPoPrez();break;
		default: printf("\nUneli ste neodgovarajuci broj.");
	}
}

void PrikaziSve(void) {
	printf("\n\nRadi se... PRIKAZI SVE\n");
	getch();
}
```

```
void PrikaziPoBI(void) {
	printf("\n\nRadi se... PRIKAZI PO BROJU INDEKSA\n");
	getch();
}

void PrikaziPoPrez(void) {
	printf("\n\nRadi se... PRIKAZI PO PREZIMENU\n");
	getch();
}
```

Верзија 2

Датотека: *meni_02.c*

```
/*
** PROJEKAT : Osnove programiranja u programskom jeziku C
** DATOTEKA : meni_02.c
** OPIS     : Napisati program u programskom jeziku C koji implementira
**            konzolni korisnički interfejs upotrebom menija.
**            V.2
** DATUM    : 11.03.2019.
** AUTOR    : S.D.L.
** PROMENE  :
**  12.03.2019. - dodat podmeni PRIKAZI (SDL)
**  xx.xx.xxxx. - <opis promene> (<programer>)
**
** Copyright (C) FON-LSI, 2019.
*/

/*

Implementirati sledeći meni:

===============
 S T U D E N T
===============

1. Ubaci     (Insert)
2. Izbaci    (Delete)
3. Promeni   (Update)
4. Prikazi   (Select)
	1. Prikazi sve
	2. Prikazi po broju indeksa
	3. Prikazi po prezimenu
	4. Kraj (tj. povratak u nadmeni)
5. Kraj rada

*/

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

#define KRAJ  0
#define RADI  1

void glavni_meni(char* stavke, int br_stavki);
void prikazi_meni(char* stavke, int br_stavki);
int  get_int(char* prompt, int donja_granica, int gornja_granica);
int  izvrsi_gm(int izbor);
int  izvrsi_pr(int izbor);
```

```
char* stavke_gl =                       // Stavke glavnog menija
   "\n===============\n "
   "S T U D E N T"
   "\n===============\n\n"
   " GLAVNI MENI\n\n"
   " 1. Ubaci\n"
   " 2. Izbaci\n"
   " 3. Promeni\n"
   " 4. Prikazi...\n"
   " 5. Kraj rada\n";

char* stavke_pr =                       // Stavke menija Prikazi
   "\n===============\n "
   "S T U D E N T"
   "\n===============\n\n"
   " PRIKAZI\n\n"
   " 1. Prikazi sve\n"
   " 2. Prikazi po broju indeksa\n"
   " 3. Prikazi po prezimenu\n"
   " 4. Kraj\n";

int main(void) {
       glavni_meni(stavke_gl, 5);
       return 0;
}

void glavni_meni(char* stavke, int br_stavki) {
       int izbor = -1;
       char prompt[20];
       sprintf(prompt, "\nVas izbor [1..%d]: ", br_stavki);
       do {
              fputs(stavke, stdout);
              izbor = get_int(prompt, 1, br_stavki);
       } while (izvrsi_gm(izbor) != KRAJ);
}
void prikazi_meni(char* stavke, int br_stavki) {
       int izbor = -1;
       char prompt[20];
       sprintf(prompt, "\nVas izbor [1..%d]: ", br_stavki);
       do {
              fputs(stavke, stdout);
              izbor = get_int(prompt, 1, br_stavki);  //
       } while (izvrsi_pr(izbor) != KRAJ);
}

int izvrsi_gm(int izbor) {
       int signal = RADI;
       system("cls");
       switch (izbor) {
              case 1:  puts("\nIzvrsavam UBACI"); system("pause"); break;
              case 2:  puts("\nIzvrsavam IZBACI"); system("pause"); break;
              case 3:  puts("\nIzvrsavam PROMENI"); system("pause"); break;
              case 4:  prikazi_meni(stavke_pr, 4); break;
              case 5:  puts("\n\nKraj rada"); signal = KRAJ; break;
              default: puts("\n\nGRESKA");
       }
       return signal;
}
```

```
int izvrsi_pr(int izbor) {
	int signal = RADI;
	system("cls");
	switch (izbor) {
		case 1:  puts("\nIzvrsavam PRIKAZI SVE"); system("pause"); break;
		case 2:  puts("\nIzvrsavam PRIKAZI PO BROJU INDEKSA"); system("pause");
break;
		case 3:  puts("\nIzvrsavam PRIKAZI PO PREZIMENU"); system("pause"); break;
		case 4:  signal = KRAJ; break;
		default: puts("\n\nGRESKA");
	}
	return signal;
}

int get_int(char* prompt, int dg, int gg) {
	char string[99];
	int number;
	do {
		do {
			fputs(prompt, stdout);
			fgets(string, 99, stdin);
		} while(!(number = atoi(string)));
	} while (number < dg || number > gg);
	return number;
}
```

Верзија 3

Датотека: *meni_03.c*

```
/*
** PROJEKAT : Osnove programiranja u programskom jeziku C
** DATOTEKA : meni_03.c
** OPIS     : Napisati program u programskom jeziku C koji implementira
**            konzolni korisnički interfejs upotrebom menija.
**            V.3
** DATUM    : 17.03.2019.
** AUTOR    : S.D.L.
** PROMENE  :
**  18.03.2019. - xxx (SDL)
**  xx.xx.xxxx. - <opis promene> (<programer>)
**
** Copyright (C) FON-LSI, 2019.
*/

/*****************************************

Имплементирати следећи мени:

===============
 S T U D E N T
===============

1. Ubaci     (Insert)
2. Izbaci    (Delete)
3. Promeni   (Update)
4. Prikazi   (Select)
	1. Prikazi sve
	2. Prikazi po broju indeksa
```

```
       3. Prikazi po prezimenu
       4. Kraj (tj. povratak u nadmeni)
5. Kraj rada

*******************************************/

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

#define MBS 5// максималан број ставки у менију

typedef enum { ERR = -2, WRN, KRAJ, OK } signal_t;  // сигнал успешности извршења
наредбе
typedef signal_t func_t(void);         // функција која извршава наредбу; без
параметара, враћа сигнал
typedef func_t* naredba_t;
typedef struct {
       int       br_stavki;      // број ставки у менију
       char *    naslov;         // наслов менија
       char *    tekst;          // текст који садржи ставке менија
       naredba_t naredba[MBS];   // наредбе
       } meni_t;

// Функције за реализацију корисничког интерфејса
signal_t izvrsi(meni_t meni);
int izaberi_naredbu(meni_t meni);
int get_int(char* prompt, int dg, int gg);  // dg = доња граница, gg = горња граница

// Функција за обраду сигнал успешности извршења наредбе
void obradi_signal(signal_t signal);

// Тип и функција за обраду упозорења и грешака
typedef enum { GR_IZZ, GR_NULL, PRAZNO_PREZ, NEMA_PREZ, PRAZAN_BI, NEMA_BI }
izuzetak_t;
void obradi_izuzetak(izuzetak_t izuzetak);

// Наредбе Главног менија
signal_t ubaci(void);
signal_t izbaci(void);
signal_t promeni(void);
signal_t prikazi(void);
signal_t kraj(void);

// Наредбе менија Прикази
signal_t prikazi_sve(void);
signal_t prikazi_po_broju_indeksa(void);
signal_t prikazi_po_prezimenu(void);
signal_t kraj_prikazi(void);

meni_t glavni_meni = {
       .br_stavki = 5,
       .naslov = "\n===============\n S T U D E N T\n===============\n\n G L A V N I   M
E N I\n",
       .tekst = " 1. Ubaci\n 2. Izbaci\n 3. Promeni\n 4. Prikazi...\n 5. Kraj rada\n",
       .naredba = { ubaci, izbaci, promeni, prikazi, kraj }
};

meni_t prikazi_meni = {
```

```
	.br_stavki = 4,
	.naslov = "\n===============\n S T U D E N T\n===============\n\n P R I K A Z
I\n",
	.tekst = " 1. Prikazi sve\n 2. Prikazi po broju indeksa\n 3. Prikazi po
prezimenu\n 4. Kraj\n",
	.naredba = { prikazi_sve, prikazi_po_broju_indeksa, prikazi_po_prezimenu,
kraj_prikazi }
};

int main(void) {
	while (izvrsi(glavni_meni) != KRAJ);
	return EXIT_SUCCESS;
}

signal_t izvrsi(meni_t meni) {
	int izbor = izaberi_naredbu(meni);
	signal_t sig = meni.naredba[izbor]();  // изврши наредбу
	obradi_signal(sig);
	return sig;
}

int izaberi_naredbu(meni_t meni) {
	system("cls");
	puts(meni.naslov);
	puts(meni.tekst);
	char prompt[22] = " Vas izbor [1..%d] -> ";
	snprintf(prompt, 22, prompt, meni.br_stavki);  // прилагоди промпт за приказ
	int i = get_int(prompt, 1, meni.br_stavki);    // изабери наредбу из менија
	return (--i);
}

void obradi_signal(signal_t signal) {
	switch (signal) {
	    case ERR:  fputs("\n>>> GRESKA: program ce biti prekinut.\n", stderr);
exit(EXIT_FAILURE);
		case WRN:
		case OK:   system("pause"); break;
		case KRAJ: break;
		default:   fputs("ERROR: izvrsi()", stderr);  // Ово не би требало да се
изврши. Никада.
	}
}

void obradi_izuzetak(izuzetak_t izuzetak) {
	// Изузетак = упозорење или грешка
	// GR_IZZ, GR_NULL, PRAZNO_PREZ, NEMA_PREZ, PRAZAN_BI, NEMA_BI
	char * poruka[] = {
		"\n>>> GRESKA: nepoznat izuzetak. ",
		"\n>>> GRESKA: vracena je NULL vrednost. ",
		"\n::: Nedostaje prezime. ",
		"\n::: Nema studenata sa navedenim prezimenom. ",
		"\n::: Nedostaje broj indeksa. ",
		"\n::: Nema studenta sa navedenim BI. "
	};
	unsigned int broj_poruka = sizeof(poruka) / sizeof(char *);
	if (izuzetak >= (izuzetak_t)broj_poruka) izuzetak = 0;
	fputs(poruka[izuzetak], stdout);
}
```

```
int get_int(char* prompt, int dg, int gg) {
	char string[99];
	int number;
	do {
		do {
			fputs(prompt, stdout);
			fgets(string, 99, stdin);
		} while(!(number = atoi(string)));
	} while (number < dg || number > gg);
	return number;
}

signal_t ubaci(void)   { puts("\nIzvrsavam ubaci()");  return OK; }
signal_t izbaci(void)  { puts("\nIzvrsavam izbaci()"); return OK; }
signal_t promeni(void) { puts("\nIzvrsavam promeni()"); return OK; }
signal_t prikazi(void) { while (izvrsi(prikazi_meni) != KRAJ); return OK; };
signal_t kraj(void)    { fputs("\nKRAJ RADA\nCopyright (C) FON-LSI, 2019.\n", stdout);
return KRAJ; };

signal_t prikazi_sve(void)              { puts("\nIzvrsavam prikazi_sve()"); return OK;
}
signal_t prikazi_po_broju_indeksa(void) { puts("\nIzvrsavam
prikazi_po_broju_indeksa()"); return OK; }
signal_t kraj_prikazi(void) { return KRAJ; }

signal_t prikazi_po_prezimenu(void) {  // пример имплементације наредбе из менија
	signal_t signal = OK;
	char prezime[99];
	system("cls");
	fputs("\nPRIKAZI PO PREZIMENU\n\nUnesite prezime: ", stdout);
	if (gets_s(prezime, 98) == NULL) {
		obradi_izuzetak(GR_NULL);
		signal = ERR;
	}
	else if (strlen(prezime) == 0) {
		obradi_izuzetak(PRAZNO_PREZ);
		signal = WRN;
	}
	else if (strncmp(prezime, "Jovanovic", strlen(prezime)) == 0) {
		fputs("\n\tPREZIME\t\tIME\n\t-----------------------\n", stdout);
		fputs("\tJovanovic\tAna\n\tJovanovic\tEma\n\tJovanovic\tMiodrag\n\n",
stdout);
	}
	else {
		obradi_izuzetak(NEMA_PREZ);
		signal = WRN;
	}
	return signal;
}
```