Programiranje 2

Laboratorijske vježbe

LV8
Rad s datotekama, binarne datoteke

Fakultet elektrotehnike računarstva i informacijskih tehnologija Osijek
Kneza Trpimira 2b
www.ferit.unios.hr

Uvod

Datoteke predstavljaju apstraktnu reprezentaciju podataka i sadržane su od niza bajtova (okteta). Datoteka može sadržavati znak, riječ, liniju podataka, pa čak i strukturu.

U prethodnim predlošcima objašnjen je rad s tekstualnom datotekom, odnosno objašnjene su funkcije s kojima se može dohvatiti neki sadržaj iz datoteke i zapisati određeni sadržaj u datoteku. Na ovim laboratorijskim vježbama nastavlja se rad s datotekama, objasnit će se manipulacija nad datotekom, rad s pogreškama i rad s binarnom datotekom.

Binarne datoteke su datoteke s ekstenzijom *.bin* ili *.dat* i zapis je nečitljiv ljudima jer se sastoji od niza 0 i 1. Takvu datoteku je potrebno interpretirati sa specijaliziranim programom. Najčešće se koriste kada se želi spremiti veliko broj numeričkih vrijednosti i sadržaj tih datoteka je ekvivalentni sadržaj reprezentacije podataka u memoriji računala. To znači da je rad s binarnim datotekama puno brži naspram rada s tekstualnim datotekama, jer se sadržaj iz memorije u izvornom obliku direktno zapisuje u binarnu datoteku i obrnuto, sadržaj iz binarne datoteke se direktno učitava u memoriju računala.

Primjer 1: Zapis cijelog broja 1234 tipa *unsigned short* koji zauzima 2bajta u binarnu datoteku.

$$1234_{10} = 00000100|11010010_2$$

U binarnom načinu zapisivanja podataka na disk, podaci se u datoteku na disku zapisuju jednako kako su predstavljeni u memoriji računala. Tako će broj 1234 zauzeti ukupno 2bajta unutar binarne datoteke.

Funkcije za manipuliranje datotekama

Ako datoteka postoji na disku, moguće joj je promijeniti ime ili ju obrisati. Funkcije koje se za to koriste su *rename()* i *remove()* i njihove deklaracije se nalaze u standardnom zaglavlju *<stdio.h>*, objašnjene su u nastavku tekstu.

Promjena imena datoteci

Ako datoteka postoji na disku, moguće joj je promijeniti ime i to funkcijom *rename()*. Deklaracija funkcije *rename()*:

```
int rename(const char* staro_ime, const char* novo_ime);
```

Funkcija *rename()* vraća cjelobrojnu vrijednost i to 0 u slučaju uspješne promjene imena, a u slučaju neuspjeha vraća -1, također funkcija prima dva parametra. Prvi parametar je pokazivač na *char* tip podatka i može primiti *string* koji predstavlja staro ime datoteke s ekstenzijom, a drugi parametar je također pokazivač na *char* tip podatka koji isto može primiti *string* koji predstavlja novo ime datoteke s ekstenzijom. Prije korištenja funkcije *rename()*, korisnički kreirani tok podataka prema datoteci treba biti zatvoren.

Primjer 2: Preimenovanje datoteke *Programiranje 1.txt* u *Programiranje 2.txt*.

```
#include <stdio.h>

int main(void) {

	FILE* fp = NULL;
	char* staro_ime = "Programiranje 1.txt";
	char* novo_ime = "Programiranje 2.txt";
	int status = 0;

	fp = fopen(staro_ime, "r");
	if (fp == NULL) {
		printf("Datoteka %s ne postoji na disku.\n", staro_ime);
		return -1;
	} else {
		fclose(fp);
	}
	status = rename(staro_ime, novo_ime);

	if (status == 0) {
		printf("Uspjesno promijenjeno ime datoteci!\n");
	} else {
		printf("Nemogucnost promijene imena datoteci!\n");
	}

	return 0;
}
```

Brisanje datoteke

Ako datoteka postoji na disku, moguće ju je obrisati s diska i to funkcijom *remove()*. Deklaracija funkcije *remove()*:

```
int remove(const char* ime_datoteke);
```

Funkcija *remove()* vraća cjelobrojnu vrijednost i to 0 u slučaju uspješnog brisanja datoteke s diska, a u slučaju neuspjeha vraća -1, također funkcija prima jedan parametar, pokazivač na *char* tip podatka i može primiti *string* koji predstavlja ime datoteke s ekstenzijom koju se želi obrisati s diska. Prije korištenja funkcije *remove()*, korisnički kreirani tok podataka prema datoteci treba biti zatvoren.

Primjer 3: Brisanje datoteke *Programiranje 1.txt* s diska računala.

```
#include <stdio.h>

int main(void) {

	FILE* fp = NULL;
	char* ime_datoteke = "Programiranje 1.txt";
	int status = 0;

	fp = fopen(ime_datoteke, "r");
	if (fp == NULL) {
		printf("Datoteka %s ne postoji na disku\n", ime_datoteke);
		return -1;
	} else {
		fclose(fp);
	}
	status = remove(ime_datoteke);

	if (status == 0) {
		printf("Uspjesno obrisana datoteka!\n");
	}
	else {
		printf("Nemogucnost brisanja datoteke!\n");
	}

	return 0;
}
```

Upravljanje pogreškama

U standardnom zaglavlju *<errno.h>* definirana je cjelobrojna makro varijabla *errno*, koju se u slučaju pojave pogreške postavlja na određenu vrijednost od strane funkcija iz standardne biblioteke. Ovaj makro je proširen tako da može bit promjenjiv. Prilikom pokretanja programa, varijabla *errno* je postavljena na vrijednost nula. Određene funkcije standardne biblioteke u slučaju pogreške mogu promijeniti vrijednost u *errno* varijabli s brojem većim od nule. Taj broj predstavlja specifičnu pogrešku koju se može ispisati na standardni izlaz za pogrešku funkcijama *perror()* i *strerror()*. Te su funkcije detaljnije objašnjene u nastavku teksta.

Primjer 4: Ispis trenutne vrijednosti varijable *errno* na standardni izlaz za pogrešku.

```c
#include <stdio.h>

int main(void) {

	FILE* fp = NULL;

	fp = fopen("datoteka.txt", "r");
	if (fp == NULL) {
		fprintf(stderr, "Vrijednost pogreske je: %d.\n", errno);
		return -1;
	}
	else {
		fclose(fp);
	}

	return 0;
}
```

Funkcija za ispis pogreške na temelju *errno* varijable

U slučaju pojavljivanja pogreške tijekom rada s datotekama, varijabla *errno* će biti postavljena na određenu vrijednost. Kako bi se znalo kakva je pogreška uslijedila, koristi se funkcije *strerror()* čija se deklaracija nalazi se u standardnom zaglavlju *<string.h>*. Deklaracija funkcije *strerror()*:

```c
char* strerror(int error_num);
```

Funkcija *strerror()* pretražuje predefinirano polje s numeriranim pogreškama na temelju predane vrijednosti i vraća pokazivač na poruku o numeriranoj pogrešci, a kao parametar prima vrijednost *errno*. Sadržaj poruke koju funkcija *strerror()* vraća ovisi o platformi na kojoj se koristi i o prevoditelju.

Primjer 5: Ispis trenutne vrijednosti varijable *errno* i poruke pogreške pomoću funkcije *strerror()* definiranu varijablom *errno* na standardni izlaz za pogrešku.

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

int main(void) {

	FILE* fp = NULL;

	fp = fopen("datoteka.txt", "r");
	if (fp == NULL) {
		fprintf(stderr, "Vrijednost pogreske je: %d.\n", errno);
		fprintf(stderr, "Pogreska: %s.\n", strerror(errno));
		return -1;
	}
	else {
		fclose(fp);
	}

	return 0;
}
```

Funkcija za ispis pogreške na standardni izlaz za pogrešku

U slučaju kada se želi ispisati poruka na standardni izlaz za pogrešku , može se koristiti funkcija *perror()* kojom se može ispisati korisnički definirana poruka, a nakon nje slijedi dvotočka s razmakom i predefinirana poruka o pogrešci. Deklaracija funkcije *perror()* nalazi se u standardnom zaglavlju *<stdio.h>*. Deklaracija funkcije *perror()*:

```c
void perror(const char* poruka);
```

Funkcija *perror()* ništa ne vraća, a za parametar prima korisnički definiranu poruku *string* koju će funkcija ispisati na *stderr* popraćeno s dvotočkom i razmakom, te porukom o pogrešci koja opisuje broj pogreške koji se nalazi u varijabli *errno*.

Primjer 6: Ispis poruke pogreške na standardni izlaz za pogrešku pomoću funkcije *perror()* definirano varijablom *errno*.

```c
#include <stdio.h>

int main(void) {
	FILE* fp = NULL;
	fp = fopen("datoteka.txt", "r");
	if (fp == NULL) {
		perror("Otvaranje");
		return -1;
	}
	else {
		fclose(fp);
	}
	return 0;
}
```

Funkcija za prekid programa *exit()*

Funkcija *exit()* služi za trenutno terminiranje pozvanog procesa, a njezina deklaracija se nalazi u <*stdlib.h*> zaglavlju.

```c
void exit(int status);
```

Funkcija *exit()* ne vraća vrijednost, a kao parametar prima cjelobrojnu vrijednost koja predstavlja status koji se predaje roditeljskom procesu. Za cjelobrojni parametar funkciji *exit()* može se proslijediti makro konstante *EXIT_FAILURE* i *EXIT_SUCCESS* koje se također nalaze u <*stdlib.h*> zaglavlju. Kada se funkciji *exit()* kao parametar preda makro *EXIT_SUCCESS*, ekvivalentno je kao *return 0* i predstavlja izlazak u slučaju uspjeha, a ako se kao parametar preda makro *EXIT_FAILURE*, ekvivalentno je kao *return -1* i predstavlja izlazak iz programa u slučaju neuspjeha.

Nije uvijek prikladno iznenadno prekinuti program funkcijom *exit()*. Potrebno je pripaziti na spremanje svih potrebnih postavki programa prije njegovog izlaska uz poruku upozorenja i korisničkim odobrenjem.

Primjer 7: Terminiranje programa funkcijom *exit()* ovisno o ispunjenom uvjetu.

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

int main(void) {
	FILE* fp = NULL;

	fp = fopen("datoteka.txt", "r");

	if (fp == NULL) {
		perror("Otvaranje");
		exit(EXIT_FAILURE);
	}
	else {
		fclose(fp);
		exit(EXIT_SUCCESS);
	}

	return 0;
}
```

Funkcije za pozicioniranje unutar datoteke

U slučaju da se tijekom zapisivanja ili čitanja podataka unutar datoteke javi potreba da se vrati na početak sadržaja datoteke ili je potrebno direktno pristupiti određenom sadržaju unutar datoteke razvijene su funkcije kojima se to može postići. Neke od tih funkcija su funkcije *rewind()*, *ftell()*, *fseek()* čije se deklaracije nalaze u standardnom zaglavlju <*stdio.h*>. Detaljnije su objašnjene u nastavku teksta.

Vraćanje na početak sadržaja datoteke

U slučaju da je potrebno vratiti se na početak sadržaja datoteke, potrebno je koristiti funkciju *rewind()*. Deklaracija funkcije *rewind()*:

```
void rewind(FILE* fp);
```

Funkcija *rewind()* ništa ne vraća, a za parametar prima korisnički kreirani tok podataka prema datoteci. U slučaju da je trenutačni položaj u sredini datoteke ili se došlo do kraja datoteke i ako je potrebno ponovno dohvatiti sadržaj s početka datoteke, potrebno je koristiti funkciju *rewind()* pomoću koje se vraća na početak toka podataka. Prije korištenja funkcije *rewind()*, tok podataka prema datoteci treba biti otvoren.

Primjer 8: Primjena funkcije *rewind()*.

```
#include <stdio.h>
void ispis(FILE*);
int main(void) {
	FILE* fp = NULL;
	const char* tekst = "Ovo je Programiranje 2!";
	fp = fopen("datoteka.txt", "w");
	if (fp == NULL) {
		perror("Otvaranje");
	}else {
		fputs(tekst, fp);
		fclose(fp);
	}
	fp = fopen("datoteka.txt", "r");
	if (fp == NULL) {
		perror("Otvaranje");
	}else {
		ispis(fp);
		rewind(fp);
		ispis(fp);
		fclose(fp);
	}
	return 0;
}
void ispis(FILE* fp) {
	int c = 0;
	while ((c = fgetc(fp)) != EOF) {
		putchar(c);
	}putchar('\n');

}
```

Direktni pristup sadržaju datoteke

Do sada se sadržaj datoteke dohvaća sekvencijalno, znak po znak ili liniju po liniju. Pozicija se nije moglo direktno promijeniti i tako dohvatiti određeni sadržaj unutar datoteke. Pomoću funkcije *fseek()* moguće je od određene pozicije promijeniti položaj unutar datoteke, a funkcijom *ftell()* može se ustanoviti na kojoj se trenutnoj poziciji nalazimo unutar datoteke. Deklaracije funkcija *fseek()* i *ftell()* nalaze se unutar standardnog zaglavlja *<stdio.h>*. Deklaracija funkcije *fseek()*:

```
int fseek(FILE* fp, long int pomak, int od_pozicije);
```

Funkcija *fseek()* vraća cjelobrojnu vrijednost i to 0 ako je promjena pozicije uspješno izvršena, u suprotno broj različit od nule ako se promjena pozicije nije uspjela izvršiti. Funkcija prima tri parametra. Prvi parametar predstavlja korisnički kreirani tok podataka unutar kojeg se želi promijeniti pozicija, drugi parametar predstavlja pomak koji se broji od trećeg parametra koji predstavlja poziciju od koje će se pomak napraviti. Treći parametar je definiran s jednim od tri makro konstante (*SEEK_SET*, *SEEK_CUR*, *SEEK_END*) koje se nalaze u standardom zaglavlju *<stdio.h>.*

Makro konstante	Opis
SEEK_SET	Početak datoteke
SEEK_CUR	Trenutna pozicija unutar datoteke
SEEK_END	Kraj datoteke

Deklaracija funkcije *ftell()*:

```
long int ftell(FILE* fp);
```

Funkcija *ftell()* vraća dugački cjelobrojni broj koji predstavlja trenutnu poziciju unutar datoteke, tj. broj bajtova. U slučaju neuspješnog izvršavanja funkcija će vratiti -1L i postavit će se odgovarajuća vrijednost u *errno* varijablu. Funkcija *ftell()* prima tok podataka za koji treba vratiti trenutnu poziciju unutar datoteke.

Primjer 9: Primjena funkcije *fseek()* za promjenu pozicije unutar datoteke.

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

void ispis(FILE*);

int main(void) {

	FILE* fp = NULL;

	fp = fopen("datoteka.txt", "w+");
	if (fp == NULL) {
		perror("Otvaranje");
		exit(EXIT_FAILURE);
	}
	else {
		fputs("Ovo je zanimljivo\n", fp);
		rewind(fp);
		ispis(fp);
		fseek(fp, 7, SEEK_SET);
		fputs("Programiranje 2!\n", fp);
		rewind(fp);
		ispis(fp);
		printf("Datoteka ima velicinu od %ld bajtova\n", ftell(fp));
		fclose(fp);
	}

	return 0;
}
void ispis(FILE* fp) {
	int c = 0;
	while ((c = fgetc(fp)) != EOF) {
		putchar(c);
	}putchar('\n');

}
```

Rad s binarnim datotekama u C programskom jeziku

Rad s binarnim datotekama je puno brži od rada s tekstualnim datotekama, jer se podaci iz memorije direktno zapisuju u binarnu datoteke i obrnuto, dok rad s tekstualni datotekama zahtjeva pretvorbu podataka prilikom zapisivanja iz memorije u datoteku i obrnuto.
Kao i kod tekstualnih datoteka, tok podataka za binarne datoteke se može kreirati pomoću funkcije *fopen()*. Potrebno je obratiti pozornost na drugi parametar funkcije *fopen()* koji predstavlja način rada i može biti jedan od sljedećih:

Način rada	Opis
rb	Otvaranje postojeće datoteke u svrhu čitanja, datoteka mora postojati.
wb	Kreiranje datoteke u svrhu zapisivanja. Ako datoteka ne postoji biti će kreirana. Ako postoji datoteka istog imena, sadržaj će biti prepisana. Ovim načinom rada sadržaj se zapisuje od početka datoteke.
ab	Otvaranje datoteke u svrhu naknadnog dodavanja sadržaja. Ako datoteka ne postoji biti će kreirana. Ovim načinom rada sadržaj će biti zapisan nakon postojećeg sadržaja u datoteci.
r+b / rb+	Otvara postojeću datoteku u svrhu čitanja i zapisivanja, datoteka mora postojati. Prilikom zapisivanja u datoteku prethodni sadržaj se neće obrisati, već će se ažurirati sadržaj ovisno o poziciji unutar datoteke.
w+b / wb+	Otvara datoteku u svrhu zapisivanja i čitanja. Ako datoteka ne postoji biti će kreirana. Ako postoji datoteka istog imena prilikom pisanja obrisat će se prethodni sadržaj.
a+b / ab+	Otvaranje datoteke u svrhu naknadnog dodavanja sadržaja i čitanja. Ako datoteka ne postoji biti će kreirana. Ovim načinom rada čitanje će započeti od početka datoteke, ali zapisivanje će se nastaviti nakon postojećeg sadržaja u datoteci.

Također nakon što se završi rad s binarnom datotekom, potrebno je zatvoriti korisnički kreirani tok podataka s funkcijom *fclose()*.

Primjer 9: Provjera postoji li datoteka u projektnoj mapi na disku.

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

int main(void) {

	FILE* fp = NULL;
	fp = fopen("binarna.bin", "rb");

	if (fp == NULL) {
		fprintf(stderr, "Vrijednost pogreske je: %d.\n", errno);
		fprintf(stderr, "Pogreska: %s.\n", strerror(errno));

	}else {
		fprintf(stdout, "Datoteka postoji na disku.\n");
		fclose(fp);
	}
	return 0;
}
```

Binarne datoteke omogućuju zapisivanje i čitanje složenih tipova podataka kao što su strukture i polja i za to se koriste funkcije *fwrite()* i *fread()* čije se deklaracije nalaze u standardnom zaglavlju *<stdio.h>*.

Zapisivanje podataka u binarnu datoteku

Zapisivanje podataka u binarnu datoteku omogućeno je funkcijom *fwrite()*. Deklaracija funkcije *fwrite()*:

```
size_t fwrite(const void* pok, size_t vel, size_t br, FILE* fp);
```

Funkcija *fwrite()* vraća cjelobrojnu vrijednost koja predstavlja ukupni broj uspješno zapisanih elemenata, a prima četiri parametra. Prvi parametar predstavlja pokazivač na element koji će se zapisati u binarnu datoteku (predaje se memorijska adresa), drugi parametar predstavlja veličinu elementa u bajtovima, treći parametar predstavlja broj elemenata i četvrti parametar predstavlja korisnički kreirani tok podataka prema binarnoj datoteci.

Primjer 10: Zapisivanje jednog podatka u binarnu datoteku.

```
#include<stdio.h>

int main(void) {

	int m = 5;
	FILE* fp = NULL;

	fp = fopen("binarna.bin", "wb");

	if (fp == NULL) {
		perror("Otvaranje");
	}else{
		fwrite(&m, sizeof(int), 1, fp);
		fclose(fp);
	}

	return 0;
}
```

Primjer 11: Zapisivanje cjelobrojnog polja u binarnu datoteku.

```c
#include<stdio.h>

int main(void) {

	int polje[] = { 1, 2, 3, 4, 5 };
	int n = sizeof(polje) / sizeof(int);

	FILE* fp = NULL;

	fp = fopen("binarna.bin", "wb");

	if (fp == NULL) {
		perror("Otvaranje");
	}
	else {
		fwrite(polje, sizeof(int), n, fp);
		fclose(fp);
	}
	//else {
	//	fwrite(polje, sizeof(polje), 1, fp);
	//	fclose(fp);
	//}

	return 0;
}
```

Primjer 12: Zapisivanje polje struktura u binarnu datoteku.

```c
#include<stdio.h>

typedef struct student {
	char ime[20];
	float prosjek;
}STUDENT;

int main(void) {
	STUDENT studenti[] = { "Matej", 3.5, "Marko", 4.0, "Luka", 4.5,
				"Ivan", 5.0 };
	int n = sizeof(studenti) / sizeof(STUDENT);
	FILE* fp = NULL;

	fp = fopen("binarna.bin", "wb");

	if (fp == NULL) {
		perror("Otvaranje");
	}else {
		fwrite(studenti, sizeof(STUDENT), n, fp);
		fclose(fp);
	}
	//else {
	//	fwrite(polje, sizeof(studenti), 1, fp);
	//	fclose(fp);
	//}
	return 0;
}
```

Čitanje podataka iz binarne datoteke

Čitanje podataka iz binarne datoteke omogućeno je funkcijom *fread()*. Deklaracija funkcije *fread()*:

```
size_t fread(void* pok, size_t vel, size_t br, FILE* fp);
```

Funkcija *fread()* vraća cjelobrojnu vrijednost koja predstavlja ukupni broj uspješno pročitanih elemenata, a prima četiri parametra. Prvi parametar predstavlja pokazivač na memorijski prostor u koji će se zapisati sadržaj iz binarne datoteke, drugi parametar predstavlja veličinu u bajtovima pojedinog elementa, treći parametar predstavlja broj elemenata i četvrti parametar predstavlja korisnički kreirani tok podataka prema binarnoj datoteci.

Primjer 13: Čitanje jednog podatka iz binarne datoteke.

```
#include<stdio.h>

int main(void) {

	int n = 0;
	FILE* fp = NULL;

	fp = fopen("binarna.bin", "rb");

	if (fp == NULL) {
		perror("Otvaranje");
	}
	else {
		fprintf(stdout, "Datoteka postoji na disku.\n");
		fread(&n, sizeof(int), 1, fp);
		fprintf(stdout, "n = %d\n", n);
		fclose(fp);
	}

	return 0;
}
```

Primjer 14: Čitanje cjelobrojnog polja iz binarne datoteke.

```c
#include<stdio.h>

int main(void) {

	int polje[5] = { 0 };
	int n = sizeof(polje) / sizeof(int);
	FILE* fp = NULL;

	fp = fopen("binarna.bin", "rb");

	if (fp == NULL) {
		perror("Otvaranje");
	}
	else {
		fprintf(stdout, "Datoteka postoji na disku.\n");
		fread(polje, sizeof(int), n, fp);

		//for (size_t i = 0; i < n; i++)
		//{
		//	fread(polje + i, sizeof(int), 1, fp);
		//}

		for (int i = 0; i < n; i++) {
			if (i == 0) {
				fprintf(stdout, "polje[%d ", polje[i]);
			}
			else if (i > 0 && i < n - 1) {
				fprintf(stdout, "%d ", polje[i]);
			}
			else {
				fprintf(stdout, "%d]\n", polje[i]);
			}
		}
		fclose(fp);
	}
	return 0;
}
```

Primjer 15: Čitanje polja struktura iz binarne datoteke.

```c
#include<stdio.h>

typedef struct student {
	char ime[20];
	float prosjek;
}STUDENT;

int main(void) {

	STUDENT studenti[4] = { "", 0.0 };
	int n = sizeof(studenti) / sizeof(STUDENT);
	FILE* fp = NULL;

	fp = fopen("binarna.bin", "rb");
	if (fp == NULL) {
		perror("Otvaranje");
	}
	else {
		fprintf(stdout, "Datoteka postoji na disku.\n");
		fread(studenti, sizeof(STUDENT), n, fp);

		//for (size_t i = 0; i < n; i++)
		//{
		//	fread(studenti + i, sizeof(STUDENT), 1, fp);
		//}

		for (int i = 0; i < n; i++) {
			if (i == 0) {
				fprintf(stdout, "studenti[%s %.2f ", studenti[i].ime,
				studenti[i].prosjek);
			}
			else if (i > 0 && i < n - 1) {
				fprintf(stdout, "%s %.2f ", studenti[i].ime,
				studenti[i].prosjek);
			}
			else {
				fprintf(stdout, "%s %.2f]\n", studenti[i].ime,
				studenti[i].prosjek);
			}
		}
		fclose(fp);
	}
	return 0;
}
```

Zadaci

1. Napisati C program kojim će se omogućiti vođenje evidencije članova jedne videoteke. Program treba omogućiti korisniku izbornik unutar kojeg će korisnik moći odabrati neku od radnji. Program treba kreirati datoteku clanovi.bin, ukoliko datoteka ne postoji na disku (prvi puta kada se pokrene program). Članovi videoteke predstavljeni su strukturom CLAN s članovima strukture koji su stringovi: ID, ime, prezime, adresa i broj mobitela. Na početku programa ponuditi korisniku izbornik za odabir željene radnje:
    1. Dodavanje novih članova u datoteku clanovi.bin,
    2. Čitanje iz datoteke clanovi.bin,
    3. Završetak programa.

    Ako se želi unijeti novi korisnik, svaki puta je potrebno odabrati opciju br. 1 i s tipkovnice unijeti podatke o korisniku, također je potrebno brojati koliko se novih korisnika unijelo, te svakim novim unosom korisnika potrebno je zapisati korisnika u datoteku. Voditi evidenciju broja unesenih korisnika. Broj korisnika je potrebno zapisati na početku datoteke, a tek poslije cijele vrijednosti je potrebno zapisati strukturu CLAN.
    Ako je potreba za čitanjem korisnika iz datoteke potrebno je odabrati opciju br. 2. Potrebno je prvo pročitati cijelu vrijednost koja predstavlja ukupni broj spremljenih korisnika unutar datoteke, te pronaći korisnika po određenom kriteriju pretraživanja, može biti po bilo kojem članu strukture CLAN.
    Ako je potrebno završiti s programom, potrebno je odabrati opciju br. 3, nakon kojeg će se korisnika upitati s porukom *"Da li ste sigurni kako zelite zavrsiti program?"* popraćeno s opcijom d*a/ne* koju je potrebno utipkati. Ako je odabrana opcija *"da"* zatvoriti program, ako je odabrana opcija *"ne"* ponuditi ponovni izbor radnji.

    Operacije za rad s datotekom izvesti s funkcijama!
    Organizirati kôd u više datoteka!

2. Iskoristiti prethodni zadataka, no dodati jednu više operaciju (posudba filma) u izbornik:
    1. Dodavanje novih članova u datoteku clanovi.bin,
    2. Čitanje iz datoteke clanovi.bin ,
    3. Posudba filma ,
    4. Završetak programa.

Kreirati strukturu FILM koja sadrži sljedeće članove koji su stringovi: ID, ime, datum.

Ukoliko se odabere opcija 3, trebaju se izvršiti sljedeće operacije:

- Pitati koliko se filmova želi posuditi (ne više od 4 – novi član u strukturi CLAN).
- Za svaki film unijeti: ime filma, datum posudbe (broj manji od 365 koji označava dan u godini) i ID člana koji je posudio film.
- Sve podatke o posuđenom filmu spremiti u datoteku posudbe.bin

Ako je korisnik već posuđivao filmove, ima pravo posuditi nove filmove dok god je broj posuđenih filmova manji od 4. Ukoliko ne postoji datoteka posudbe.bin, potrebno ju je kreirati, u suprotnom je otvaramo i dodajemo nove posuđene filmove, također je potrebno pratiti broj unesenih filmova.