Programiranje 2

Laboratorijske vježbe

LV7
Uvod u rad s datotekama, tekstualne 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.
C programski jezik može raditi s dvije vrste datoteka, a to su:

- Tekstualne datoteke (*.txt*) i
- Binarne datoteke (*.bin*, *.dat*).

Tekstualne datoteke su datoteke s ekstenzijom *.txt* koje se standardno mogu kreirati i uređivati s bilo kojim tekstualnim uređivačem. Sadržaj ovakve datoteke je običan tekst, odnosno niz *ASCII* znakova, gdje svaki red završava znakom za novi red. Ako se u tekstualnu datoteku zapisuju numerička vrijednosti, tada se radi pretvorba iz numeričke vrijednosti u *ASCII* znak i obrnuto, kada se dohvaća numerička vrijednost iz tekstualne datoteke koja je pretvorena u *ASCII* znak, vrši se pretvaranje iz *ASCII* znaka u numeričku vrijednost. Tekstualne datoteke zauzimaju više prostora naspram binarnih datotekama i nešto je sporije zapisivanje i čitanje iz takve vrste datoteka.

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

1234_{10}

'1'(49_{10} = 00110001_2) '2'(50_{10} = 00110010_2) '3'(51_{10} = 00110011_2) '4'(52_{10} = 00110100_2)

Svaka individualna znamenku cijelog broj se zamjenjuje s ASCII vrijednosti. Svaki znak zauzima 1bajt, što će ukupno zauzeti 4bajta unutar tekstualne datoteke.

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 2: 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.

Datoteke u programskom jeziku C

Programski jezik C postupa sa svim uređajima kao da su datoteke, npr. tipkovnica koja predstavlja standardni ulaz u program, ekran koji predstavlja standardni izlaz iz programa. Informacije koje dolaze sa standardnog ulaza ili se šalju na standardni izlaz predstavljaju tok podataka (engl. *stream*) i taj se tok podataka sastoji od niza bajtova. Ulaz u program može biti osim standardnog ulaza i sadržaj nekakve datoteke na disku, ili izlaz iz programa može biti osim standardnog izlaza, također datoteka na disku. Znači, programski jezik C postupa s izvorom ulaznog toka i odredištem izlaznog toka kao s datotekom.

Također programski jezik C automatski predefinira tri toka podataka za rad s tipkovnicom i ekranom računala.

Standardne datoteke	FILE pokazivač	Uređaji
Standardni ulaz	*stdin*	Tipkovnica
Standardni izlaz	*stdout*	Ekran
Standardni izlaz za pogrešku	*stderr*	Ekran

Sve tri toka podataka, *stdin*, *stdou* i *stderr* predstavljaju konstantne pokazivače na *FILE* tip podatka koji su povezani sa standardnim ulazom i izlazom. Standardni ulaz *stdin* povezan je s tipkovnicom, a standardni izlaz *stdout* povezan je s ekranom i oba toka podataka imaju spremnik podataka, dok je standardni izlaz za pogrešku *stderr* povezan s ekranom i koristi se za dijagnostiku, te nema spremnik podataka.

FILE tip podatka predstavlja strukturu i njezina se definicija nalazi u standardnom zaglavlju *<stdio.h>*. Unutar *FILE* strukture nalaze se sve potrebne informacije o toku podataka, odnosno datoteci nad kojom će se vršiti ulazne i izlazne operacije.

Rad s tekstualnim datotekama u programskom jeziku C

Funkcijom *fopen()* kreira se tok podataka, odnosno omogućava se kreiranje ili otvaranje postojeća datoteka na disku, bila ona tekstualna ili binarna. Deklaracija funkcije *fopen()*:

```
FILE* fopen(const char* ime_datoteke_ext, const char* nacin_rada);
```

Funkcija *fopen()* kao povratnu vrijednost vraća pokazivač na *FILE* tip podatka, odnosno funkcija vraća memorijsku adresu početka memorijskog bloka koji je zauzet za *FILE* strukturu čiji se članovi inicijaliziraju s određenim vrijednostima za ispravnu komunikaciju s datotekom. U slučaju neuspješnog kreiranja ili otvaranja datoteke, funkcija će vratiti *NULL* pokazivač. Prvi parametar predstavlja ime datoteke s ekstenzijom. Ako se želi kreirati ili otvoriti datoteka koja se nalazi unutar mape gdje se nalazi izvorni kôd, dovoljno je navesti ime datoteke s ekstenzijom unutar dvostrukih navodnika. U slučaju da se datoteka želi kreirati ili otvoriti unutar druge mape, potrebno je navesti cijelu putanju do datoteke, uključujući i ime datoteke s ekstenzijom, također unutar dvostrukih navodnika. Način pisanje putanje ovisi od platforme do platforme (Windows, Unix/Linux/BSD/Mac). Drugi parametar predstavlja način rada s datotekom i može predstavljati jedan od navedenih:

Način rada	Opis
r	Otvaranje postojeće datoteke u svrhu čitanja, datoteka mora postojati.
w	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.
a	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+	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+	Otvara datoteku u svrhu zapisivanja i čitanja. Ako datoteka ne postoji biti će kreirana. Ako postoji datoteka istog imena, prilikom pisanja će se obrisati prethodni sadržaj.
a+	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.

Prije korištenja funkcije *fopen()* potrebno je deklarirati pokazivač na *FILE* tip podatka u koji će se spremiti povratna vrijednost funkcije *fopen()*. Taj deklarirani pokazivač će predstavljati tok podataka prema datoteci koju se kreiralo ili otvorilo pomoću funkcije *fopen()*. Također nakon završenog zapisivanja ili čitanja datoteke, potrebno je zatvoriti korisnički kreirani tok podataka datoteke, očistiti spremnik podataka i osloboditi memoriju od *FILE* strukture koja je bila zadužena za rad s datotekom. To se radi pomoću funkcije *fclose()* koja ima sljedeću deklaraciju:

```
int fclose(FILE* pok);
```

Funkcija *fclose()* kao povratnu vrijednost vraća cjelobrojnu vrijednost, a za parametar prima pokazivač na *FILE* tip podatka, odnosno memorijsku adresu korisnički kreiranog toka podataka prema datoteci. U slučaju da funkcija *fclose()* uspješno zatvori tok podataka datoteke, vratit će vrijednost 0, a u slučaju neuspjeha vratiti će *EOF*. *EOF* označava kraj datoteke (engl. *end-of-file*) i predstavlja makro konstantu negativne cjelobrojne vrijednosti, najčešće -1, a njezina definicija se nalazi u standardnom zaglavlju *<stdio.h>*.

Primjer 3: Provjera da li datoteka postoji u projektnoj mapi na disku.

```c
#include <stdio.h>

int main(void) {

	FILE *fp = NULL;
	const char *imeDatoteke = "moja_datoteka.txt";
	//const char imeDatoteke[] = "moja_datoteka.txt";

	fp = fopen(imeDatoteke, "r");
	//fp = fopen("moja_datoteka.txt", "r");

	if (fp == NULL) {
		printf("Datoteka %s ne postoji na disku.\n", imeDatoteke);
	}
	else {
		printf("Datoteka %s postoji na disku.\n", imeDatoteke);
		fclose(fp);
	}

	return 0;
}
```

Primjer 4: Provjera da li datoteka postoji u određenoj mapi na disku.

```c
#include <stdio.h>

int main(void) {

	FILE *fp = NULL;

	fp = fopen("C:\\Users\\Public\\Documents\\moja_datoteka.txt", "r");

	if (fp == NULL) {
		printf("Datoteka ne postoji na disku.\n");
	}
	else {
		printf("Datoteka postoji na disku.\n");
		fclose(fp);
	}

	return 0;
}
```

Funkcije za čitanje iz datoteke

Dosada se za ulaz u program koristio standardni ulaz, odnosno tipkovnica. Sadržaj sa standardnog ulaza se dohvaćao s funkcijama iz standardne biblioteke kao što je funkcija *getchar()*, ~~*gets()*~~, *scanf()*, koje imaju predefinirani rad s tokom podataka *stdin*, odnosno standardnim ulazom. Ulaz u program može biti i sadržaj datoteke koji je potrebno dohvati funkcijama čiji se tok podataka može preusmjeriti. Zato su razvijene funkcije *fgetc()*, *fgets()* i *fscanf()* koje su dio standardne biblioteke, i čije se deklaracije nalaze u standardnom zaglavlju *<stdio.h>*.

Funkcija za čitanje individualnih znakova

Kako bi se dohvaćali individualni znakovi iz datoteke ili standardnog ulaza koristi se funkcija *fgetc()* koja ima sljedeću deklaraciju:

```
int fgetc(FILE* pok);
```

Funkcija *fgetc()* vraća cjelobrojnu vrijednost koja predstavlja ASCII znak dohvaćen iz određenog toka podataka, te povećava za jedan poziciju unutar datoteke. U slučaju neuspješnog dohvaćanja znaka funkcija će vratiti *EOF*. Kao parametar prima pokazivač na *FILE* tip podatka koji može biti korisnički kreirani tok podataka ili *stdin* pokazivač.

Primjer 5: Dohvaćanje individualnih znakova iz datoteke pomoću funkcije *fgetc()*.

```
#include <stdio.h>

int main(void) {

	int brojac = 0;
	int znak = -1;
	int status = 1;
	FILE* fp = NULL;
	const char *imeDatoteke = "moja_datoteka.txt";
	fp = fopen("moja_datoteka.txt", "r");

	if (fp == NULL) {
		printf("Datoteka %s ne postoji na disku.\n", imeDatoteke);
		status = 0;
	}
	else {
		while ((znak = fgetc(fp)) != EOF) {
			brojac++;
		}
		fclose(fp);
	}

	if (status) {
		printf("Datoteka %s ima ukupno %d znakova.\n", imeDatoteke, brojac);
	}

	return 0;
}
```

Unutar *while()* uvjeta za ispitivanje istinitosti dohvaća se znak po znak iz datoteke i znakovi se spremaju u varijablu *znak* čiji se sadržaj ispituje da li je različit od *EOF*. Sve dok je sadržaj varijable *znak* različit od *EOF*, tijelo *while()* petlje će se izvršavati, tek kada se dođe do kraja datoteke, funkcija *fgetc()* vraća *EOF* i time se prekida izvršavanje *while()* petlje.

Funkcija za čitanje niza znakova (*string*)

Kako bi se dohvatio *string* iz datoteke ili standardnog ulaza koristiti se funkcija *fgets()* koja ima sljedeću deklaraciju:

```
char* fgets(char* spremnik, int n, FILE* pok);
```

Funkcija *fgets()* vraća memorijsku adresu prvog elementa polja *spremnik*, ako je uspješno dohvaćen *string* iz datoteke ili standardnog ulaza. U slučaju kada se došlo do kraja datoteke i nije se dohvatio nikakav sadržaj, sadržaj polja *buff* ostat će prazan i funkcija će vratiti *NULL* pokazivač. Prvi parametar predstavlja pokazivač na *char* tip podatka, odnosno prima ime polja znakova unutar kojeg će se spremiti *string* dohvaćen iz datoteke ili standardnog ulaza, drugi parametar predstavlja duljinu polja, a treći parametar predstavlja tok podataka iz kojega će se dohvatiti sadržaj. Tok podataka može biti korisnički kreiran ili *stdin* pokazivač. Čitanje znakova iz toka podataka će se izvršavati dok se ne učita *n – 1* znak, dok se ne učita znak za novi red *'\n'* ili dok se ne dođe do kraja datoteke *EOF*, ovisno što se prije dogodi.
Funkcija *fgets*() će uključiti znak za novi red *'\n'* unutar string-a, u slučaju da je linija znakova kraća od duljine polja znakova u koji se spremaju znakovi, te će ga zatvoriti unutar *string-a* sa znakom *'\0'* te je preporučeno nakon svakog dohvaćanja *string-a* funkcijom *fgets()* provjeriti da li postoji znak *'\n'* unutar *stringa* i prepisati ga s znakom *'\0'*.

Primjer 6: Dohvaćanje linije znakova, odnosno string-a iz datoteke pomoću funkcije *fgets()*.

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

void provjeraStringa(char*);

int main(void) {

	char stringPolje[100] = { '\0' };
	FILE* fp = NULL;

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

	if (fp == NULL) {

		printf("Datoteka ne postoji na disku.\n");
	}
	else {

		while (fgets(stringPolje, 100, fp) != NULL) {

			puts(stringPolje);
			provjeraStringa(stringPolje);
			puts(stringPolje);
		}

		fclose(fp);
	}

	return 0;
}

void provjeraStringa(char *polje) {

	int n = strlen(polje);

	if (polje[n - 1] == '\n') {
		polje[n - 1] = '\0';
	}
}
```

Unutar *while()* uvjeta za ispitivanje istinitosti dohvaća se linija po linija znakova iz datoteke i sprema se u polje znakova *stringPolje*, te se provjerava povratna vrijednost funkcije *fgets()* da li je različita od *NULL* vrijednosti. Sve dok je povratna vrijednost funkcije *fgets()* različita od *NULL* vrijednosti tijelo *while()* petlje će se izvršavati, tek kada se dođe do kraja datoteke, funkcija *fgets()* će vratiti *NULL* vrijednost i time se prekida izvršavanje *while()* petlje. Kako se izvršava tijelo *while()* petlja, svaki puta će se pozvati funkcija *puts()* koja na standardni izlaz ispisuje sadržaj predanog joj polja *stringPolje*, te ispisom *string-a* uključuje prelazak pozicije u novi red. Ako se u polju znakova *stringPolje* kao zadnji znak nalazi znak *'\n'*, funkcija *puts()* će ispisati znak za novi red i dodatno će uključiti prelazak pozicije u novi red. Funkcija *provjeraStringa()* povjerava zadnji znak predanog joj polja, ako je zadnji znak *'\n'*, funkcija će prepisati zadnji znak sa znakom *'\0'*. Funkcijom *provjeraString()* uklonit će se znak za novi red, te drugim pozivom funkcije *puts()* regularno će se ispisati sadržaj predanog joj polja *stringPolje* i uključit će prelazak pozicije u novi red.

Funkcija za čitanje formatiranog sadržaja s ulaza

Kako bi se dohvatio formatirani sadržaj iz datoteke ili standardnog ulaza koristiti se funkcija *fscanf()* koja ima sljedeću deklaraciju:

```
int fscanf(FILE* pok, const char* format, ...);
```

Funkcija *fscanf()* vraća cjelobrojnu vrijednost koja predstavlja broj uspješno dohvaćenih vrijednosti iz datoteke, odnosno broj uspješno sparenih vrijednosti s memorijskim lokacijama gdje se te dohvaćene vrijednosti trebaju spremiti, a taj broj može biti manji od željenog broja dohvaćenih vrijednosti ili čak nula u slučaju nemogućnosti sparivanja vrijednosti. Prvi parametar predstavlja tok podataka koji može biti korisnički kreiran ili *stdin* pokazivač, drugi parametar predstavlja format, odnosno *string* koji sadrži poseban format očekivanog tipa podatka. Treći parametar predstavlja niza parametara koji trebaju primiti očekivanu vrijednost iz drugog parametra. Drugi i treći parametar trebaju odgovarati po broju specifikatora pretvorbi i memorijskim adresama gdje će se dohvaćeni sadržaj spremiti. Također funkcija *fscanf()* u slučaju pogreške ili ako je došla do kraja datoteke vraća *EOF* vrijednost.

Primjer 7: Dohvaćanje sadržaja iz datoteke u formatiranom obliku pomoću funkcije *fscanf()*.

```
#include<stdio.h>

int main(void) {

	char ime[20] = { '\0' };
	char prezime[30] = { '\0' };
	int ocjena;
	int bodovi;
	int status;

	FILE* fp = NULL;

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

	if (fp == NULL) {
		printf("Datoteka se ne moze otvoriti.\n");
	}
	else {
		while ((status = fscanf(fp, "%s %s %d %d",
ime, prezime, &ocjena, &bodovi)) != EOF) {
			printf("Status je %d.\n", status);
			printf("Ime: %s Prezime: %s\tOcjena: %d Bodovi: %d.\n",
ime, prezime, ocjena, bodovi);
		}
		printf("Status je %d.\n", status);

		fclose(fp);
	}

	return 0;
}
```

Unutar *while()* uvjeta za ispitivanje istinitosti dohvaća se sadržaj iz datoteke u formatiranom obliku i provjerava se povratna vrijednost funkcije *fscanf()* da li je različita od *EOL* vrijednosti. Sve dok je povratna vrijednost funkcije *fscanf()* različita od *EOF* vrijednosti tijelo *while()* petlje će se izvršavati, tek kada se dođe do kraja datoteke, funkcija *fscanf()* će vratiti *EOF* vrijednost i time se prekida izvršavanje *while()* petlje.

Funkcije za zapisivanje u datoteku

Dosada se za izlaz iz programa koristio standardni izlaz, odnosno ekran. Sadržaj iz programa se ispisivao na standardni izlaz pomoću funkcija iz standardne biblioteke kao što su funkcije *putchar()*, *puts()*, *printf()*, koje imaju predefinirani rad s tokom podataka *stdout*, odnosno standardnim izlazom. Izlaz iz program može biti i datoteke u koju je potrebno zapisati sadržaj s funkcijama čiji se tok podataka može preusmjeriti. Zato su razvijene funkcije *fputc()*, *fputs()* i *fprintf()* koje su dio standardne biblioteke, i čije se deklaracije nalaze u standardnom zaglavlju *<stdio.h>*.

Funkcija za zapisivanje individualnih znakova

Kako bi se zapisali individualni znakovi u datoteku ili na standardni izlaz može se koristiti funkcija *fputc()* koja ima sljedeću deklaraciju:

```
int fputc(int c, FILE* pok);
```

Funkcija *fputc()* vraća cjelobrojnu vrijednost koja predstavlja *ASCII* znak koji se zapisuje u tok podataka, te povećava za jedan poziciju unutar datoteke. U slučaju neuspješnog zapisivanja znaka, funkcija će vratiti *EOF*. Prvi parametar funkcije predstavlja znak koji se treba zapisati, a drugi parametar predstavlja tok podataka u koji će se upisati znak, a može biti korisnički kreiran ili *stdout* pokazivač.

Primjer 8: Zapisivanje individualnih znakova u datoteke pomoću funkcije *fputc()*.

```
#include<stdio.h>

int main(void) {

	int znak;
	FILE *fp = NULL;
	int br = 0;
	fp = fopen("moja_datoteka.txt", "w");

	if (fp == NULL)
	{
		printf("Datoteka se ne moze kreirati.\n");
	}
	else {
		while ((znak = getchar()) != EOF)
		{
			printf("Upisani znak u datoteku %c\n", fputc(znak, fp));

			br++;
			if (br == 5) {
				printf("Pritisnite Ctrl+Z u DOS-u ili Ctrl+D u Linux-u\
kako bi se onemogucilo daljnje ucitavanje znakova\n\n");
				br = 0;
			}
		}
		fclose(fp);
	}
	return 0;
}
```

Unutar *while()* uvjeta za ispitivanje istinitosti pomoću funkcije *getchar()* dohvaća se znak po znak sa standardnog ulaza. Znakovi se spremaju unutar privremenog spremnika i tek kada se pritisne tipka *ENTER*, znak po znak se iz privremenog spremnika spremaju u varijablu *znak*, te dok je god sadržaj varijable *znak* različit od *EOF* tijelo *while()* petlje se izvršava. Funkcija *fputc()* zapisuje znakove iz varijable *znak* u datoteku, te svakim zapisom funkcija *fputc()* vraća cjelobrojnu vrijednost zapisanog znaka koji se ispisuje u standardni izlaz pomoću funkcije *printf()*. Sve dok je sadržaj varijable *znak* različit od *EOF*, tijelo *while()* petlje će se izvršavati, tek kada se pritisne *CTRL + Z* u Windows-a ili *CTRL + D* u Linux-u prekida se izvršavanje *while()* petlje.

Funkcija za zapisivanje niza znakova (*string*)

Kako bi se zapisao *string* u datoteku ili standardni izlaz koristiti se funkcija *fputs()* koja ima sljedeću deklaraciju:

```
int fputs(const char* spremnik, FILE* pok);
```

Funkcija *fputs()* vraća cjelobrojnu ne negativnu vrijednost u slučaju uspješnog zapisivanja stringa u tok podataka, a u slučaju neuspješnog zapisa vraća *EOF*. Prvi parametar predstavlja pokazivač na *char* tip podatka čiji se sadržaj ne može promijeniti, odnosno prima ime polja znakova unutar kojega se nalazi *string* koji se treba zapisati. Drugi parametar predstavlja tok podataka u koji će se sadržaj prvog parametra zapisati, a tok podataka može biti korisnički kreiran ili *stdout* pokazivač. Funkcija zapisuje *string* u tok podataka do *null* znaka *'\0'*.

Primjer 9: Zapisivanje linije znakova, odnosno string-a u datoteku pomoću funkcije *fputs()*.

```
#include<stdio.h>

int main(){
	char stringPolje[100] = { '\0' };
	FILE *fp = NULL;
	int br = 0;
	fp = fopen("moja_datoteka.txt", "w");
	if (fp == NULL)
	{
		printf("Datoteka se ne moze kreirati.\n");
	}
	else {
		while (fgets(stringPolje, 100, stdin) != NULL)
		{
			fputs(stringPolje, fp);
			br++;
			if (br == 5) {
				printf("Pritisnite Ctrl+Z u DOS-u ili Ctrl+D u Linux-u\
kako bi se onemogucilo daljnje ucitavanje znakova\n\n");
				br = 0;
			}
		}
		fclose(fp);
	}
	return 0;
}
```

Unutar *while()* uvjeta za ispitivanje istinitosti dohvaća se linija po linija znakova iz standardnog ulaza i sprema se u polje znakova *stringPolje*, te se provjerava povratna vrijednost funkcije *fgets()* da li je različita od *NULL* vrijednosti. Sve dok je povratna vrijednost funkcije *fgets()* različita od *NULL* vrijednosti tijelo *while()* petlje će se izvršavati, tek kada se pritisne *CTRL + Z* u Windows-a ili *CTRL + D* u Linux-u prekida se izvršavanje *while()* petlje. Kako se izvršava tijelo *while()* petlja, svaki puta će se pozvati funkcija *fputs()* pomoću koje se zapisuje linija znakova u datoteku iz polja znakova *stringPolje*. Ako se u polju znakova *stringPolje* kao zadnji znak nalazi znak *'\n'*, funkcija *fputs()* će zapisati znak za novi red u datoteku.

Funkcija za zapisivanje formatiranog sadržaja na izlaz

Kako bi se zapisao formatirani sadržaj u datoteku ili standardni izlaz, koristiti se funkcija *fprintf()* koja ima sljedeću deklaraciju:

```
int fprintf(FILE* pok, const char* format, ...);
```

Funkcija *fprintf()* vraća cjelobrojnu vrijednost koja predstavlja broj uspješno zapisanih znakova iz string-a u datoteku. U slučaju neuspješnog zapisa funkcija vraća negativni broj, odnosno *EOF* vrijednost. Prvi parametar predstavlja tok podataka u koji će se upisati formatirani sadržaj, a tok podataka može biti korisnički kreiran ili *stdout* pokazivač. Drugi parametar predstavlja format, odnosno *string* koji će biti zapisan u tok podataka, također može sadržavati oznake koje mogu biti zamijenjene vrijednošću koje se nalaze u dodatnim parametrima.

Primjer 10: Zapisivanje sadržaja u datoteke u formatiranom obliku pomoću funkcije *fprintf()*.

```
#include<stdio.h>

int main(void) {

	char ime[20] = { '\0' };
	char prezime[30] = { '\0' };
	int ocjena = 0;
	int bodovi = 0;
	int status = 0;
	int n = 0;
	FILE* fp = NULL;
	fp = fopen("moja_datoteka.txt", "w");

	if (fp == NULL) {
		printf("Datoteka se ne moze otvoriti.\n");
	}
	else {
		printf("Unesite broj zapisa koje zelite zapisati.\n");
		scanf("%d", &n);
		for (int i = 0; i < n; i++)
		{
			printf("Unesite ime za %d. studenta\n", i + 1);
			scanf("%19s", ime);
			printf("Unesite prezime za %d. studenta\n", i + 1);
			scanf("%29s", prezime);
			printf("Unesite ocjenu za %d. studenta\n", i + 1);
			scanf("%d", &ocjena);
			printf("Unesite bodove za %d. studenta\n", i + 1);
			scanf("%d", &bodovi);
			status = fprintf(fp, "Student br: %d\tdIme: %s Prezime:\
%s\tOcjena: %d Bodovi: %d.\n", i + 1, ime, prezime, ocjena, bodovi);
			printf("Status je %d.\n", status);
		}
		fclose(fp);
	}
	return 0;
}
```

Zadaci

1. Napisati C program koji će pročitati cijeli broj iz datoteke *"dat.txt"* i toliko puta tražiti unos imena i prezimena studenta sa standardnog ulaza. Sva učitana imena i prezimena zapisati u datoteku *"studenti.txt"* koju je potrebno kreirati iz programa i to tako da zapis bude u formatu: *"Student broj: %d.\tIme: %s\t Prezime: %s\n"*, svaki zapis u novi red. Datoteku *"dat.txt"* kreirati na računalu pomoću *Notepad-a*, te upisati cijeli broj i spremiti sadržaj datoteke. Prilikom otvaranja datoteke *"dat.txt"* navesti apsolutnu putanju do sam datoteke.

   Pojedine dijelove programa razdijeliti u funkcije,
   Organizirati kôd u različite datoteke.

2. Napisati C program koji će iz datoteke *"in.txt"* učitati dva cijela broja. Popuniti realnu matricu *A* pseudo-slučajnim vrijednostima iz intervala [-125.5, 65.8]. U datoteku *"out.txt"* i na ekran ispisati dio matrice , u matričnom obliku, koja se dobije tako da se izostavi prvi stupac i zadnji redak. Datoteku *"out.txt"* treba kreirati iz programa. Datoteku *"in.txt"* kreirati na računalu pomoću *Notepad-a*, te upisati dva cijela broja svaki u svoj red i spremiti sadržaj datoteke. Prilikom otvaranja datoteke *"in.txt"* navesti apsolutnu putanju do sam datoteke.

   Pojedine dijelove programa razdijeliti u funkcije,
   Koristiti dinamičko zauzimanje memorije (u potpunosti rukovati memorijom),
   Primijeniti isključivo pokazivačku notaciju,
   Organizirati kôd u različite datoteke.

3. Napisati C program koji kopira sadržaj jedne tekstualne datoteku u drugu pri čemu se mijenjaju mala slova u velika. Datoteka iz koje se kopira sadržaj zove se *"prva.txt"* i pripremljena je samo za čitanje. Datoteku pod imenom *"druga.txt"* potrebno je kreirati iz programa i u nju se zapisuje sadržaj iz datoteke *"prva.txt"*. Datoteku *"prva.txt"* kreirati na računalu pomoću *Notepad-a*, te upisati generirani sadržaj sa stranice https://baconipsum.com/, te spremiti sadržaj datoteke. Prilikom otvaranja datoteke *"prva.txt"* navesti apsolutnu putanju do sam datoteke.

   Pojedine dijelove programa razdijeliti u funkcije,
   Organizirati kôd u različite datoteke.