XI – Rad sa datotekama

- U C-jeziku implementirano je niz funkcija koje na jedinstven način tretiraju sve ulazno/izlazne operacije: unos sa tastature, ispisivanje na ekran računara kao i čitanje i pisanje informacija na hard disku.
- Komuniciranje sa uređajima koji obavljaju ove operacije vrši se sekvencijalno bajt po bajt, a programski mehanizam kojim se vrši ovakav prenos informacija naziva se **tok** (***stream***).
- U jednom programu može se raditi sa više tokova.
- Svakom toku se dodeljuje jedna struktura podataka imena **FILE**, koja je definisana u biblioteci **<stdio.h>**.
- Osnovna namena te strukture je da služi kao memorijski ulazno/izlazni bufer (***I/O buffer***) pri prenosu podataka.
- Važno je znati da se pri izlaznim operacijama podaci ne šalju direktno spoljnim uređajima, već se najpre upisuju u ovaj bafer.
- Kada se on ispuni, tada se sadržaj celog bafera šalje spoljnom uređaju.
- Tako se smanjuje broj pristupa disku i ubrzava rad sa datotekama.
- Sličnu namenu bafer ima i pri ulaznim operacijama.

XI – Rad sa datotekama

- Sve tokove možemo podeliti u četiri grupe:
  1. **standardni ulaz** (vrši prijem podataka sa tastature)
  2. **standardni izlaz** (vrši ispis na ekran računara)
  3. **standardno javljanje greške** (obično na ekranu računara)
  4. **datotečni tok** (vrši čitanje ili pisanje podataka u datoteku)
- Standardni ulaz, standardni izlaz i standardni tok dojave greške se samoinicijaliziraju pri pokretanju programa, a njihov pokazivač na strukturu **FILE** nalazi se u globalnim promenljivama:
  1. **FILE *stdin**; /* pokazivač toka standardnog ulaza */
  2. **FILE *stdout**; /* pokazivač toka standardnog izlaza */
  3. **FILE *stderr**; /* pokazivač toka dojave greške */
- Ovi pokazivači su deklarisani u datoteci **<stdio.h>**.
- Ako želimo da koristimo **spoljašnju datoteku** u C programu, moramo samostalno da joj pridružimo neki **tok**
- Inicijalizaciju pokazivača tokova na datoteke mora da uradi programer
- Programer ne mora da vodi računa o detaljima kako se stvarno izvršava ulazno/izlazni prenos podataka u i iz datoteka.

XI – Rad sa datotekama

- Ono što on mora da zna je pokazivač toka, sa kojim se komunicira, i funkcije pomoću kojih se ta komunikacija realizuje.
- Ovo podrazmeva deklarisanje promenjlive zvane **pokazivač na datoteku,** i davanje vrednosti toj promenljivoj
- Vrednost koja se dodeljuje dobija se pozivom funkcije koja pokušava da otvori specificiranu datoteku
- Gledano sa strane korisnika, datoteka ima **ime** i verovatno neki smisleni **sadržaj**
- Sa strane programa, datoteka je **tok bajtova** kome se pristupa preko **pokazivača na fajl**
- Svim operacijama koje pristupaju sadržaju datoteke ili koje obavljaju određene operacije nad datotekama pristupa se preko **pokazivača** na tu datoteku.
- Ako želimo da radimo se datotekama **obavezno** je da se na početku programa uključi biblioteka **#include <stdio.h>**
- Rad sa datotekama (čitanje ili upis) zahteva osnovna **tri koraka**: **Otvaranje** datoteke (pristup), **Čitanje/Upis** i **Zatvaranje** datoteke.

XI – Rad sa datotekama

- **Pokazivač toka (FILE *)** se mora navesti kao argument svake funkcije sa kojom se vrše ulazno/izlazne operacije.
  Primer: *za formatirani ispis podataka koristi se funkcija* ***fprintf*** *koja se poziva na sledeći način*:

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

  gde je **pTok** pokazivač toka, a tri tačke označavaju da se funkcija može koristiti sa promjenljivim brojem argumenata.
- Formatirani ispis se vrši prema obrascu koji se zapisuje u stringu format, na isti način kako se zapisuje format ispisa u printf() funkciji.
  Primer: *za ispis stringa "**Hello World**" na standardnom izlazu može se koristiti sledeća naredba*:

  fprintf(stdout, "Hello World");

  koja ima isto dejstvo tj. potpuno je identična kao i naredba:

  printf("Hello World");
- Zapravo, funkcija **printf()** je interno realizovana kao **fprintf()** funkcija kod koje je definisan pokazivač toka na **stdout**.

XI – Pojam datoteke

- Datoteka predstavlja imenovano područje u sekundarnoj memoriji, najčešće na hard disku, koje služi za smeštanje podataka.
- Sa stanovišta operativnog sistema datoteka je složen objekat koji se obično sastoji od više povezanih fragmenata smeštenih na različitim lokacijama u sekundarnoj memoriji.
- Kako operativni sistem brine o svim detaljima rada sa datotekama, C posmatra datoteku sasvim funkcionalno kao celinu u koju je moguće upisati niz znakova ili ga iz nje pročitati.
- Preciznije, u C-u je datoteka kontinuirani niz okteta (***byte***-ova) kome se može pristupati potpuno individualno.
- Prilikom čitanja iz datoteke niz okteta je usmeren od datoteke prema programu; program može čitati ulazni niz znak-po-znak.
- Kod pisanja u datoteku niz znakova ide od programa prema datoteci.
- Najmanja jedinica koja može biti pročitana/upisana je **1 znak** (oktet).
- Ovakva apstraktna definicija datoteke omogućava da se tastatura i ekran računara tretiraju kao datoteke.
- Sa tastature je moguće samo čitati, dok je na ekran moguće samo pisati

XI – Pojam datoteke

- Informacije se u datotekama zapisuju u kodiranom obliku.
- **Dva su načina** kodiranog zapisa: **binarni/tekstualni** ili **formatirani**.
- Zapis je upisan u binarnu datoteku kada se informacije na hard disk upisuju u istom binarnom obliku kako su kodirane u memoriji računara
- U tekstualnim datotekama upis se vrši formatirano pomoću **ASCII** koda, na isti način kako se vrši tekstualni ispis na video monitoru.
- Sadržaj tekstualnih datoteka se može pregledati bilo kojim editorom teksta, dok sadržaj binarnih datoteka obično može razumeti samo program koji ih je formirao.
- Treba imati na umu da kada se u tekstualnu datoteku formatirano upisuje C-string, tada se ne zapisuje završni znak '**\0**'.
- Uobičajeno se zapis u tekstualnim datotekama unosi u redovima teksta, na način da se za oznaku kraja reda koristi znak '**\n**'.
- Takve zapise zovemo ***linija.***
- Poželjno je da se ne unose linije koje sadrže više od **256 znakova**, jer se time osigurava da će tekstualna datoteka biti ispravno očitana sa gotovo svim programima koji operišu sa tekstualnim zapisima.

XI – Pristup datotekama

- Svaka datoteka ima svoje jedinstveno ime.
- Ime datoteke je upamćeno na disku kao tekstualni zapis u posebnoj sekciji kataloga diska.
- Uz ime su zabeleženi i podaci o datoteci: vreme kada je upamćena, broj bajtova koje datoteka zauzima, mesto na disku gde je upamćen sadržaj datoteke i atributi pristupa datoteci (*read*, *write*, *hidden*).
- Da bi mogli da koristimo neku datoteku potrebno je od operativnog sistema zatražiti dozvolu za pristup toj datoteci.
- Taj proces se zove otvaranje datoteke.
- Isto tako se i za kreiranje nove datoteke mora zatražiti dozvola od OS
- Otvaranje datoteke obavlja standardna funkcija **fopen()**.
- Pored komunikacije sa OS ona formira **datotečni tok** koji sadrži memorijski bafer za efikasno čitanje ili pamćenje podataka
- Funkcija **fopen()** se deklariše na sledeći način:

FILE *fopen(const char *ime, *const char *mod);*

ime je string koji sadrži ime datoteke, sa potpunom putanjom gde se ta datoteka nalazi, a **mod** predstavlja dozvoljeni način rada sa datotekom

XI – Pristup datotekama

Primer: `char *filename = "c:\\data\\list.txt";`

ovaj string bi koristili na Windows računarima za otvaranje datoteke koja ima ime **list.txt** i koja se nalazi na disku c: u direktorijumu pod imenom **data**.

- Zabranjeno je da u imenu datoteke koristimo znakove: /, \, :, *, ?, “, <, > i |.
- Ako se zapiše samo ime datoteke, podrazumeva se da se datoteka nalazi u tekućem direktorijumu.
- **mod** je string kojim se opisuje način otvaranja datoteke.
- Zapisuje se sa jednim ili više znakova: **r**, **w**, **a** i **+**.

Mod	Opis
r	Otvara trenutni tekstualni fajl za čitanje podataka
w	Otvara tekstualni fajl kako bi se izvršio upis. Ako fajl ne postoji, novi fajl se kreira i program će izvršiti upis na početak fajla.
a	Otvara tekstualni fajl za upis. Ako fajl ne postoji kreira se novi fajl. Upis će izvršiti dodavanje sadržaja na već postojeći sadržaj
r+	Otvara tekstualni fajl za čitanje i pisanje.
w+	Otvara tekstualni fajl za čitanje i pisanje. Sav sadržaj fajla se briše. Ukoliko fajl ne postoji kreira se novi fajl.
a+	Otvara tekstualni fajl za čitanje i pisanje. Kreira fajl ukoliko ne postoji. Čitanje polazi od početka fajla, a pisanje će izvršiti dodavanje sadržaja na kraju fajla.

XI – Pristup datotekama

- F-ja **fopen** treba da uspostaviti komunikaciju sa OS-om i njegovim rutinama za rad sa datotekama, i da vratiti pokazivač datoteke **fp** koji će se koristiti za zapisivanje ili čitanje podataka iz datoteke.
- Pokazivač datoteke pokazuje na strukturu (tipa FILE) koja sadrži informacije o datoteci (deklaracija FILE nalazi se u **stdio.h**)
- U slučaju da datoteka ne može da se otvori vraća se NULL
- Najčešći uzroci greške pri otvaranju datoteke su:
  - ✓ Neispravan zapis imena datoteke.
  - ✓ Neispravan zapis direktorijuma ili oznake diska.
  - ✓ Ne postoji direktorijum koji smo dali u imenu
  - ✓ Zahtev da se otvori nepostojeća datoteka u modu čitanja – "r".
- Preporučuje se da se uvek proveri da li je datoteka otvorena bez greške
- ANSI standard pravi razliku između tekstualnih i binarnih datoteka, pa uz oznaku **mod** treba dodati '**b**' ukoliko se radi sa binarnom datotekom
- Datotekama se može pristupiti na jedan od dva načina:
  1. **Sekvencijalno**
  2. **Direktno ili slučajno** (*random access*)

XI-Otvaranje i zatvaranje datoteka

Primer: *Napisati kod za upisivanje podataka u datoteku "vtš.txt"*

```
FILE * fp;
fp = fopen("vtš.txt", "w");
if( fp == NULL)
printf("Greška pri otvaranju datoteke");
```

- Kada je datoteka otvorena, koristi se kao tok.
- Tako ako prethodno napisanom kodu programa dodamo naredbe:

```
fprintf( fp, "SRT\n");
fprintf( fp, "%s\n" "SRT drugi put!");
```

```
SRT
SRT drugi put
```

- Kada se ne pristupa datoteci,treba je zatvoriti sa **int fclose (FILE *fp)**; čime se prekida veza između pokazivača datoteke i njenog imena
- Dva su osnovna razloga za to:

1. Ušteda resursa jer OS često ograničava broj istovremeno otvorenih datoteka
2. Kako bi svi podaci iz računara, koji se jednim delom nalaze u baferu strukture FILE, bili smešteni na disk.

XI-Otvaranje i zatvaranje datoteka

- Funkcija **fclose()** prima argument koji je pokazivač toka prethodno otvorene datoteke, a vraća vrednost nula ako je proces zatvaranja uspešan ili EOF ako pri zatvaranju dođe do greške.
- Ukoliko se ne zatvori datoteka pomoću ove funkcije, ona će biti prisilno zatvorena po završetku programa.
- Preporuka je da se uvek zatvori datoteka čim se sa njom završi rad, jer se time štede resursi OS i osigurava od mogućeg gubitka podataka (pri resetovanju računara, nestanku el.napajanja,...).
- Ako je potrebno da datoteke budu otvorene sve vreme tada se koristiti funkcija **fflush()** kojom se forsira pražnjenje bafera datoteke i ažurira stanje datoteke na disku, bez zatvaranja datoteke.
- Prototip funkcije ***fflush()*** je **int fflush(FILE *fp);**
  funkcija prima argument koji je pokazivač toka prethodno otvorene datoteke, a vraća vrednost nula ako je proces pražnjenja bafera uspešan ili EOF ako pri zapisu podataka iz bafera nastane greška.
- Funkcija **fflush(stdin)** se takođe često koristi za odstranjivanje viška znakova iz standardnog ulaza.

XI-Otvaranje i zatvaranje datoteka

Primer: *U datoteku "podaci.txt" potrebno je upisati prvih 10 prirodnih brojeva, a zatim iz iste datoteke očitati brojeve dok se ne stigne do kraja i ispisati ih na standardni izlaz.*

```
#include <stdio.h>
#include <stdlib.h> /* funkcija exit */
main()
{
int i;
int br;
FILE* f = fopen("podaci.txt", "w");
if (f == NULL)
{
printf("Greska prilikom otvaranja
datoteke podaci.txt za pisanje\n");
exit(1);
}
/*Učitavanje brojeva*/
for (i = 0; i<10; i++)
   fprintf(f, "%d\n", i);
fclose(f);
/*Otvaranje datoteke za čitanje*/
f = fopen("podaci.txt", "r");
if (f == NULL)
   { printf("Greska prilikom otvaranja
   datoteke podaci.txt za citanje\n");
   exit(1);
   }
/*Čitanje brojeva i njihovo ispisivanje*/
while(fscanf(f, "%d", &br) == 1)
         printf("Procitano : %d\n", br);
   fclose(f);
}
```

XI – Čitanje i upisivanje podataka

- **Funkcija** za čitanje jednog znaka (bajta) iz datoteke je:

  int fgetc(FILE *pok_dat)

  - Ako je pre učitavanja znaka došlo do kraja datoteke rezultat funkcije je EOF (simbolička konstanta koja označava kraj datoteke (-1)).
  - Ova konstanta je, takođe, definisana u fajlu **stdio.h**.

- **Funkcija** za upis jednog znaka (bajta) u datoteku je:

  int fputc(char c,FILE *pok_dat)

  - Rezultat funkcije je upisani znak.

- **Funkcija** za čitanje jednog reda iz tekstualne datoteke je:

  char *fgets(char *s, int maxbr, FILE *pok_dat)

  - Učitava jedan red (niz znakova do znaka ‘\n’) iz datoteke na koju ukazuje **pok_dat**. **s** je niz u koji se smešta učitani red, a **maxb-1** je maksimalni broj znakova koji se mogu učitati.

- **Funkcija** za upis jednog reda u tekstualnu datoteku je:

  char *fputs(char *s, FILE *pok_dat)

XI – Čitanje i upisivanje podataka

➢ **Funkcija** za formatirano čitanje podataka iz tekstualne datoteke

int fscanf(FILE *pok_dat,char *format [,arg]…)

- Funkcija služi za čitanje podataka iz fajla na koji ukazuje **pok_dat**. Povratna vrednost predstavlja broj podataka koje je funkcija učitala u argumente. Ako se javi greška pri učitavanju povratna vrednost je -1
- Parametri funkcije:
  1. **pok_dat** – pokazivač na fajl iz koga treba učitavati podatke.
  2. **format** – predstavlja niz karaktera koji označava način konverzije (niza karaktera u podatak odgovarajućeg tipa) koji treba primeniti pri učitavanju podataka iz fajla u promenljive zadate kao parametri **argument** (npr. “%d”). Ovaj parametar odgovara prvom parametru funkcije **scanf()**.
  3. **argument** – predstavlja promenljivu u koju treba učitati vrednost. Tip ovog argumenta mora odgovarati tipu konverziji naznačenom u parametru **format**.
- Može postojati više parametara **argument**, sa tim da broj argumenata ovog tipa treba da odgovara broju specifikacija konverzija u formatu

XI – Rad sa datotekom

Primer:

```
char znak, rec[20];
int povratna, brojCeo;
double brojRealan;
FILE *pokFajla;
pokFajla = fopen( “Dokument.txt”, “r” );
...
povratna = fscanf( pokFajla, “%c %s %d %f”, znak, rec,
brojCeo, brojRealan );
...
povratna = fclose( pokFajla );
```

XI – Rad sa datotekom

➢ **Funkcija** za formatirani upis podataka u tekstualne datoteke

int fprintf(FILE *pok_dat,char *format [,arg]…)

- Funkcija služi za upisivanje podataka u fajl na koji ukazuje **pok_dat**. Povratna vrednost predstavlja broj podataka koje je funkcija upisala u fajl. Ako se pojavi greška pri učitavanju povratna vrednost je -1.
- **Parametri:**

1. **pok_dat** – pokazivač na fajl iz koga treba učitavati podatke.
2. **format** – predstavlja niz karaktera koji označavaju način konverzije (podatka odgovarajućeg tipa u niz karaktera) koji treba primeniti pri upisu podataka u fajl iz promenljivih koje su navedene kao parametri **argument** (npr. “%d”). Ovaj parametar odgovara prvom parametru funkcije **printf()**.
3. **argument** – predstavlja promenljivu čiju vrednost treba upisati u fajl. Tip ovog argumenta mora odgovarati tipu konverziji naznačenom u parametru **format**. Može postojati više parametara **argument**, s tim da broj argumenata ovog tipa treba da odgovara broju specifikacija konverzija u formatu da ne bi došlo do greške.

XI – Rad sa datotekom

Primer:

```
char znak, rec[20];
int povratna, brojCeo;
double brojRealan;
FILE *pokFajla1, *pokFajla2;
pokFajla1 = fopen( “Dokument1.txt”, “r” );
pokFajla2 = fopen( “Dokument2.txt”, “r” );
...
povratna = fscanf( pokFajla1, “%c %s %d %f”, znak, rec, brojCeo, brojRealan );
...
povratna = fprintf( pokFajla2, “%c %s %d %f”, znak, rec, brojCeo, brojRealan );
...
povratna = fclose( pokFajla1 );
povratna = fclose( pokFajla2 );
```

XI - Funkcije za direktan pristup

- Da bi se direktno pristupilo nekom podatku u okviru datoteke, mora se znati njegovo rastojanje u bajtovima od početka datoteke, kraja datoteke ili od tekuće pozicije.
- **Standardna funkcija** za pozicioniranje u datoteci

int fseek(FILE *pok_dat, long int n, int modus)

pok_dat - ukazuje na datoteku u kojoj se vrši pozicioniranje,

n - broj bajtova za koliko se pomera pokazivač na datoteku u odnosu na zadatu lokaciju,

modus - definiše u odnosu na koju lokaciju se vrši pomeranje:

modus=0, pomeranje se vrši u odnosu na početak datoteke,

modus=1, pomeranje se vrši u odnosu na tekuću poziciju,

modus=2, pomeranje se vrši u odnosu na kraj datoteke.

- **Funkcija** za pozicioniranje na početak datoteke

void rewind(FILE *pok_dat)

- **Funkcija** za nalaženje trenutne vrednosti pokazivača datoteke

long int ftell (pok_dat)

Daje trenutnu poziciju pokazivača datoteke (broj bajtova od početka)

XI – Rad sa datotekom

- **Funkcija** za ispitivanje da li je pokazivač datoteke pozicioniran na kraj datoteke

  int feof(FILE *dat_pok);

  Vraća vrednost različitu od nule ukoliko je pokazivač datoteke pozicioniran na kraj datoteke.

- **Funkcija** za binarno čitanje

  int fread(void *ulaz, long vel, long br, FILE *uldat)

  Iz binarne datoteke na koju ukazuje **pok_dat** učitava se **br** podataka veličine **vel** bajtova i to smešta počev od lokacije na koju ukazuje pokazivač **ulaz**.

- **Funkcija** za binarni upis

  int fwrite(void *izlaz, long vel, long br, FILE *izdat)

  U binarnu datoteku na koju ukazuje **pok_dat** upisuje se **br** podataka veličine **vel** bajtova. **Izlaz** je početna adresa od koje su smešteni podaci koji se prenose u datoteku.

XI – Osnovne funkcije za rad sa datotekama

1. **FILE fopen(char filename, char mode); -** otvara datoteku čije je ime zapisano u stringu filename gde mode određuje način pristupa Vraća FILE pokazivač kojim se ta datoteka predstavlja u memoriji raˇc
2. **int fclose (FILE fp);**-zatvara datoteku predstavljenu pokazivačem fp
3. **int getc(FILE fp);**
4. **int fgetc (FILE fp); -** čita znak iz datoteke predstavljene pokazivačem fp, te vraća pročitani znak.
5. **int fputc(int c, FILE fp);**
6. **int putc(int c, FILE fp);** - zapisuje znak c u datoteku
7. **int fscanf (FILE fp, char format, ...);** - formatirano čitanje podataka iz datoteke prema formatu specificiranom stringom **format**.
8. **int fprintf (FILE fp, char format, ...);** - formatiran ispis podataka u datoteku prema formatu specificiranom stringom **format**.
9. **int feof (FILE fp);**-vraća **true** ako se čitanjem datoteke predstavljene pokazivačem **fp** stiglo do kraja te datoteke, odnosno **0** inače.
10. **void rewind(FILE fp);** - repozicionira pokazivač **fp** tako da "pokazuje" na početak datoteke.

XI – Biblioteka stdio.h

1. FILE - tip podataka (struktura) koja čuva sve informacije o fajlu, koje se koriste napr kod otvaranja fajla..
2. FILE *stdin-stdin je povezan sa tokom za standardni ulaz za podatke
3. FILE *stdout - stdout je povezan sa standardnim tokom koji se koristi za izlaz iz programa.
4. FILE *stderr - stderr je povezan sa tokom za greške.
5. EOF - vrednost koja označava kraj fajla (end-of-file); Za ANSI C to je negativni celobrojna konstanta, čija je vrednost tradicionalno -1.
6. NULL - vrednost nultog pointera (konstanta 0)
7. BUFSIZ - celobrojna konstanta (int) koja specificira “odgovarajuću” veličinu bafera preko kojih ide UI za fajlove.
8. size_t - *unsigned* tip podataka čija je veličina takva da može da sačuva bilo koju vrednost koju može da vrati sizeof

Hvala na pažnji !!!

Pitanja

? ? ?