Programiranje I

Fajlovi

Enumeracija

Strukture - Uvodno

Rad sa fajlovima

- Programski jezik C poznaje tip podatka "pokazivač na fajl".
- Da bi se radilo sa ovim tipom u C-u mora biti uključena programska biblioteka stdio.h.
- Pokazivači na fajl se deklarišu kao:
  FILE *a;
- Do trenutka inicijalizacije ove promjenljive (inicijalizacija se obavlja pomoću naredbe fopen) pokazivač je zapravo neupotrebljiv.

Naredba fopen

- Naredba **fopen** ima sintaksu:
  a=fopen("fajl koji se otvara","način otvaranja");
- Ako se "fajl koji se otvara" nalazi u tekućem direktorijumu navodi se samo sa imenom, npr. "Test.txt".
- Ako je fajl koji se želi otvoriti nalazi negdje u nekom od foldera na disku mora se navesti putanja do njega: "C:\\TEMP\\PROBA.TXT". Zbog čega se koristi \\, a ne samo \?
- Ako se fajl želi otvoriti za čitanje drugi argument fopen-a je **"r"**. Ako je otvaranje obavljeno uspješno pokazivač **a** se pozicionira na početak postojećeg fajla.

Naredba fopen

- Ako se fajl otvara za upis drugi argument je **"w"**. Ako fajl ranije nije postojao biće kreiran, a ako je postojao biće izbrisan. U oba slučaja će pokazivač biti pozicioniran na početak otvorenog fajla.
- Nadovezivanje na kraj fajla (bez brisanja postojećeg sadržaja) se obavlja sa **"a"** (pokazivač se sada pozicionira na kraj fajla).
- Opcija **"+"** omogućava otvaranje i za čitanje i za upisivanje.
- Svaki od ovih karaktera može biti praćen slovom **"t"** ili slovom **"b"**.

Naredba fopen

- Kombinacija **"rt"** označava da se fajl otvara za čitanje u tekstualnom modu, dok, recimo, **"wb"** znači da se fajl otvara za upis u binarnom modu.
- Naš kompajler po pravilu otvara u tekstualnom modu tako da se **t** može izostaviti.
- Koje su razlike između binarnog i tekstualnog režima?
- Kod binarnog režima podaci se u fajl smještaju u zavisnosti od binarne reprezentacije tipa podatka, dok se u tekstualnom režimu svi podaci smještaju u formi ASCII koda (jedan karakter-jedan bajt).

Naredba fopen

- Na primjer, broj 1234567 se u binarnom režimu zapisuje kao 4 bajta (pod uslovom da int zauzima 4 bajta), dok u tekstualnom režimu zauzima 7 bajta, za svaki karakter po jedan ('1', '2', '3', '4', '5', '6' i '7').
- Ako se fajl ne može otvoriti iz bilo kog razloga (ne postoji za čitanje, nema prostora za upis, zabranjen pristup) naredba fopen vraća NULL pokazivač.
- Svaka naredba fopen mora biti praćena provjerom da li je fajl otvoren i preduzimanjem eventualnih korektivnih akcija.

Naredba fopen

- Primjer otvaranja fajla sa provjerom
```
a=fopen("abc.txt","r");
      if(a==NULL) exit(1);
```
- Postoji više funkcija za upis i čitanje iz fajla. Uglavnom liče na već poznate naredbe iz stdio.h biblioteke (a i ove funkcije pripadaju toj biblioteci).
- Razlikuju se po dodatnom argumentu koji predstavlja pokazivač na fajl.
- Na primjer, ako je fajl otvoren za upis, funkcija
`fprintf(a,"%d \n",2);`
upisuje predmetni broj i znak za novi red u fajl na koji pokazuje **a** (pozicionira se na početak narednog reda fajla).

Upis i čitanje iz fajla

- Sve naredbe za upis i čitanje iz fajla vrše pomjeranje pozicije za čitanje i upis tako da je naredna pozicija tamo gdje se stalo sa prethodnom naredbom.
- Funkcije fputc(c,a) i putc(c,a) upisuju karakter c u fajl a.
- Funkcija fputs(s,a) upisuje string s u fajl a i prelazi u novi red.
- Funkcije fgetc(a) i getc(a) vraćaju tekući karakter iz fajla.

Upis i čitanje iz fajla

- Funkcija fgets(s,n,a) učitava string s iz fajla na koji pokazuje pokazivač a, ali ne više od n karaktera. Ovo je mala razlika u odnosu na naredbu gets(s) koja učitava string do znaka Enter, jer se u fajlovima može dogoditi da je sve štampano u jednom redu, tako da bi praktično čitav fajl bio učitan odjednom.
- Napominjemo da se na kraju fajla nalazi poseban simbol koji je definisan preko simboličke konstante EOF (End-Of-File). Prilikom upisa na kraj fajla vrši se pomjeranje ovog karaktera, dok se čitanje ne može vršiti nakon njega.

Upis i čitanje iz fajla

- Dakle, prilikom čitanja treba uvijek vršiti provjeru da li je dostignut kraj fajla. Ilustrujmo rad sa fajlovima na primjeru 2.9.10 iz zbirke.
- **Primjer:** Pretpostavlja se da postoji fajl u čijem je svakom redu upisan cijeli broj. Na osnovu ovog fajla treba formirati novi koji se razlikuje od prethodnog po tome što su brojevi u neparnim redovima udvostručeni.

```
#include <stdio.h>
int main(){
int broj, i;
FILE *a,*b;
```

```
if((a=fopen("ciobroj.txt","r"))==NULL) {
        puts("Greska pri otvaranju fajla");
        exit(1);}
if((b=fopen("novicio.txt","w"))==NULL) {
        puts("Greska pri otvaranju fajla");
        exit(1);}
```

Primjer

```
i=1;
while(fscanf(a,"%d",&broj)!=EOF){
        fprintf(b,"%d\n",(i%2==1) ? 2*broj : broj);
        i++;
}
fclose(a);
fclose(b);
}
```

- Pojasnimo ukratko ovaj program. Inicijalizovali smo dvije cjelobrojne promjenljive od kojih i služi za brojanje redova fajla, a broj je promjenljiva u koju učitavamo brojeve koji se nalaze u fajlu.
- Inicijalizovali smo dva pokazivača na fajlove.

- Prvi fajl smo otvorili za čitanje, a drugi za upis. U istoj naredbi vršimo provjeru da li je otvaranje uspjelo poređenjem sa NULL. U slučaju da nije uspjelo izlazimo iz programa.
- U petlji, zatim, učitavamo broj po broj iz fajla na koji pokazuje a, sve do kraja fajla.

Primjer - pojašnjenje

- U slučaju da je broj i paran **(ovaj broj se inkrementira nakon svakog učitavanja)** u fajl na koji pokazuje pokazivač b se vrši njegov upis, a ako je neparan **(ako je ostatak dijeljenja sa 2 različit od 0)** upisuje se njegova duplirana vrijednost.
- Na kraju se fajlovi zatvaraju (ove će naredbe biti kasnije objašnjene).
- Još jedan par funkcija se može koristiti za upis i čitanje iz fajla. Za upis podataka iz memorije u fajl može poslužiti funkcija čiji je prototip:

```
int fwrite(void *p, int d, int n, FILE *a);
```

fwrite i fread

Argumenti funkcije fwrite su:

- p pokazivač na zonu u memoriji sa koje počinje prepisivanje u fajl;
- d dužina (u bajtima) objekta koji se prepisuje;
- n broj objekata koji se prepisuju;
- a pokazivač na fajl u koji se vrši prepisivanje.

- Funkcija za prepisivanje iz fajla u memoriju je fread i ima prototip:
  int fread(void *p,int d,int n,FILE *a);

  Značenje argumenata je kao i kod fwrite, ali se sada prepisuju podaci iz fajla u memoriju.
- Upis i čitanje se po pravilu uvijek vrše pomoću parova funkcija (fscanf, fprintf), (fgets, fputs) i (fread, fwrite).

Pomjeranje u fajlu

- Pored standardnog načina za pomjeranje pokazivača na fajl naredbama za upis i čitanje, postoje i druge specijalizovane funkcije koje samo vrše pomjeranje po fajlu bez upisa i čitanja.
- Prva od njih je

  rewind(a)

  koja "premotava" fajl a na početak, odnosno pokazivač a nakon ove naredbe pokazuje na početak fajla. Podsjeća na komandu za premotavanje trake na kasetofonu, a ima i slično porijeklo jer je nekad najznačajniji memorijski medij bila baš magnetna traka koja se s vremena na vrijeme morala premotati.

Pomjeranje u fajlu

- Druga značajna funkcija za pomjeranje u fajlu je fseek(a,n,k).

  Argumenti funkcije su:
  - a pokazivač na fajl
  - n udaljenost od referentne pozicije (u bajtima)
  - k referentna pozicija
- Pretpostavljamo da su prva dva argumenta jasna. Treći argument omogućava predefinisanje pozicije u odnosu na koju se pomjeramo. Obično se zadaje preko simboličke konstante koja može uzeti jednu od sljedeće tri vrijednosti:

  SEEK_SET (ili numerički 0 → početak fajla)

  SEEK_CUR (ili numerički 1 → tekuća pozicija)

  SEEK_END (ili numerički 2 → kraj fajla).

Pomjeranje u fajlu

- Jasno je da se od kraja fajla može ići samo unazad (n je negativno), a od početka unaprijed, dok se iz tekuće pozicije, podrazumjevajući da je negdje u sredini fajla, možemo kretati i unaprijed i unazad.
- Funkcija ftell(a) kazuje tekuću poziciju u fajlu na koju pokazuje a.
- Funkcija feof(a) provjerava da li smo u fajlu stigli do EOF karaktera (naredba vraća 1 ako jesmo i 0 ako nijesmo).

Zatvaranje fajlova

- Da bi promjene koje su se vršile nad fajlovima bile pravilno ažurirane, fajlovi se moraju, prije završetka programa, **obavezno** zatvoriti.
- Fajl koji je otvoren preko pokazivača a može se zatvoriti funkcijom:
  fclose(a);
- Svi fajlovi koji su otvoreni u programu jednovremeno se zatvaraju funkcijom:
  _fcloseall();

Redirekcija - kratko

- Postoje situacije kada jedan dio rada sa fajlovima umjesto programa može vršiti operativni sistem. To se obavlja preko tzv. **redirekcije**.
- Posmatrajmo sljedeći elementarni program:

```
#include <stdio.h>
int main(){
        int a,b;
        scanf("%d %d",&a,&b);
        printf("%d %d",a,b);
}
```

Redirekcija - kratko

- Program prima dva cjelobrojna argumenta i onda ih štampa. Pretpostavimo da se izvršna verzija programa zove pr1.exe.
- Izvršavanje se obavlja sa komande linije naredbom:
  pr1
- Program sačeka da unesemo dva broja koja zatim odštampa na ekranu.
- Međutim, verzija naredbe:
  pr1 > ex.txt
  će sa komandne linije primiti dva broja, koja neće odštampati na ekranu.

Redirekcija - kratko

- Umjesto na ekran rezultat je preusmjeren (redirektovan) u fajl ex.txt, a znak **>** obavještava jezgro operativnog sistema da rezultate treba preusmjeriti sa ekrana (tzv. standardnog izlaza) u fajl.
- Slično, naredba:
  pr1 < ul.txt
  će iz fajla ul.txt učitati podatke (neće ih tražiti sa tastature – standardnog ulaza) i rezultat će odštampati na ekranu.
- Dozvoljene su i kombinacije sa preusmjeravanjem i ulaza i izlaza.

Ovo je samo ilustracija bez potrebe da joj bude posvećena značajna pažnja. U operativnim sistemima UNIX/Linux postoji mnogo složeniji sistem koji dozvoljava operacije tipa: sjutra u 6 pokreni program i rezultat smjesti u fajl koji ćeš nakon dva sata poslati na e-mail.

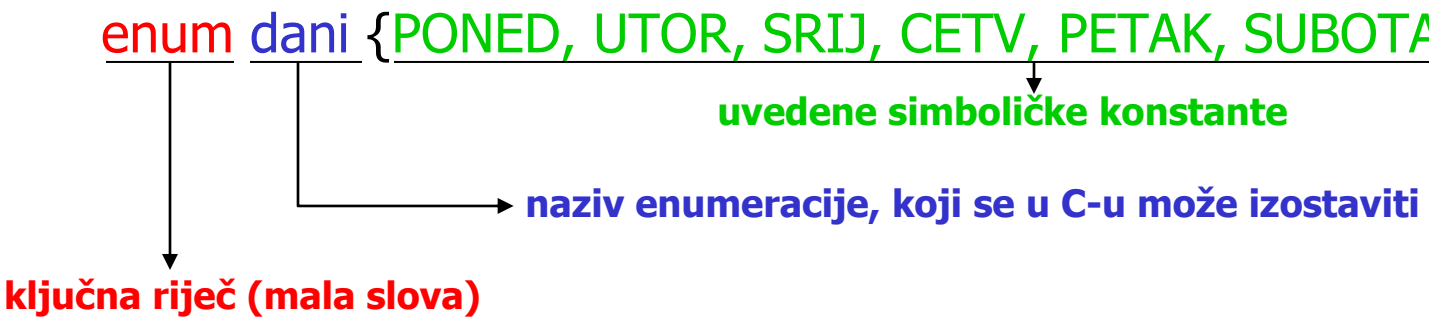
Enumeracija

- Enumeracija (ili nabrajanje) je specijalni tip podatka u C-u koji predstavlja zamjenu za veći broj cjelobrojnih konstanti.
- Konstantama se dodjeljuju imena (tzv. simboličke konstante).
- Prije nego objasnimo detaljnije enumeraciju pojasnimo njenu primjenu.
- Dani u nedjelji se u programu mogu pamtiti kao stringovi, ali ih je često mnogo bolje zapamtiti kao cijele brojeve. To onda znači da programer treba da zapamti preslikavanje dana u odgovarajući cijeli broj i da kroz program koristi cijeli broj gdje bi se inače pozvao na dan.

Enumeracija

- Čak i da programer ima takav kapacitet da pamti sva preslikavanja koja uvede u program to ne znači da će to umijeti da prate drugi programeri koji budu nastavljali njegov rad. Stoga je mnogo bolje definisati odgovarajuće nabrajanje i uvesti simboličke konstante koje već po nazivu sugerišu o kom se danu (ili bilo kom drugom podatku) radi.
- Enumeracija se definiše najčešće globalno (van svih funkcija u programu). Primjer:

```
enum dani {PONED, UTOR, SRIJ, CETV, PETAK, SUBOTA, NEDJ};
```



Enumeracija

- Simboličke konstante uzimaju vrijednosti PONED=0, UTOR=1, SRIJ=2, CETV=3, PETAK=4, SUBOTA=5, NEDJ=6.
- Pokušaj promjene vrijednosti simboličke konstante dovodi do greške u programu i prekida izvršavanja.
- Nije potrebno simboličke konstante pisati velikim slovima, ali je to konvencija.
- Primjer:
  enum {AAA=6,BBB=-1,CCC,DDD=3,EEE};

  Konstante kojima su dodijeljene vrijednosti uzimaju te vrijednosti, dok one koje nijesu specificirane uzimaju vrijednost za jedan veću od prethodne. Ako prva nije definisana njena vrijednost je 0.

Strukture

- Do sada smo se srijetali sa kolekcijama podataka istog tipa i te kolekcije smo nazivali nizovima (sa specijalnim slučajevima - matricama i stringovima).
- Modelovanje podatka iz realnog svijeta je često teško ostvarivo preko do sada učenih tipova podataka, uključujući i nizove.
- Na primjer, RADNIK, MAŠINA, STUDENT, PROZOR se teško modeluju ijednim do sada uvedenim tipom podataka.
- Stoga su definisane strukture (alternativni nazivi su zapis, slog, record), koje mogu da sadrže više podataka raznorodnog tipa.

Strukture

- Strukture se u programskom jeziku C uvode pomoću ključne riječi struct:

```
struct naziv_strukture {
        deklaracija promjenljivih članica strukture
} promjenljive_strukturnog_tipa;
```

- Primjer:

```
struct str {
        char ch;
        int i;
} s1, s2;
```

Sada su promjenljive s1 i s2 strukturnog tipa str (tipa struktura str) i svaka od ovih promjenljivih sadrži (zauzima memoriju) za cijeli broj i i karakter ch.

Memorija se za jedan podatak strukturnog tipa zauzima redom (karakter, pa cijeli broj).

Strukture

- Struktura je vidljiva tamo gdje je definisana.
- Po pravilu se strukture definišu van svih funkcija (globalno), ali se istovremeno ne deklarišu (ako nema potrebe za globalnim strukturama) promjenljive odgovarajućeg tipa.
- Promjenljive strukturnog tipa se mogu deklarisati unutar funkcija (mogu biti i argumenti funkcija):

```
struct str {
        char ch;
        int i;
};
```

Definicija strukture

```
struct str s1,s2;
```

Deklaracija promjenljivih strukturnog tipa

Strukture

- Sama struktura predstavlja mustru (šemu, pravilo) kako će se promjenljive strukturnog tipa smjestiti u memoriju računara. Stoga, struktura ne zauzima memoriju!
- Promjenljive strukturnog tipa zauzimaju memoriju koja je jednaka zbiru memorija potrebnih za smještanje promjenljivih članica strukture.
- Podaci strukturnog tipa se mogu inicijalizovati navođenjem odgovarajućih promjenljivih unutar vitičastih zagrada. Za strukturu iz prethodnog primjera, to se radi sa:
  `struct str s1={'a',4}, s2;`

Strukture

- Promjenljivim unutar strukturnog tipa se može pristupiti preko operatora tačka.

```
s1.ch='A';
s1.i=123;
```

- Unutar strukture se može nalaziti niz kao podatak član i tom nizu trebaju biti poznate sve dimenzije kako bi se mogla obaviti alokacija memorije:

```
struct Radnik{
        char ImePrezime[40];
        int GodineStaza;
} rad1={"Mitar Pavlović", 4};
```

Strukture

- Sada se elementima ovog niza može pristupiti kao:
  rad1.ImePrezime[7]='a';
  Učitavanje imena i prezimena bi bilo:
  scanf("%d%s", &rad1.GodineStaza, rad1.ImePrezime);
- Moguće je deklarisati niz struktura (tačnije je reći niz podataka strukturnog tipa):
  struct Radnik Pogon[40];
- Elementima niza se pristupa na standardni način:

  Pogon[7].GodineStaza=11;
  strcpy(Pogon[3].ImePrezime,"Petar Pavlovic");

Strukture i typedef

- Neki programeri ne vole da "vuku" ključnu riječ struct kroz program, već umjesto nje uvode "sopstveni tip podatka" preko typedef. Ovo se, po pravilu, vrši globalno. Na primjer:

```
typedef struct {
        char ImPrez[40];
        int godUpis, godStud;
} Student;
```

- Deklaracija i upotreba promjenljivih ovog tipa se obavlja navođenjem samo novodefinisanog tipa, npr.:
  Student DrugaGod[90];