Algoritmi i programiranje

Programski jezik C

Ulaz, izlaz i datoteke/fajlovi

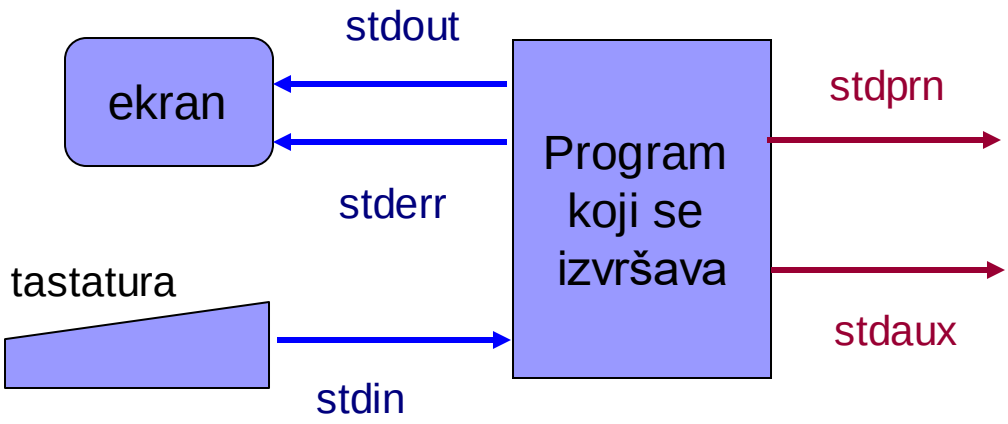
- Programi čitaju podatke sa spoljašnjih uređaja i ispisuju podatke na druge spoljašnje uređaje
- printf i scanf pišu/čitaju na ekran tj. sa tastature – prolazni ulaz/izlaz (UI).
- Trajni UI omogućavaju **datoteke** (drugi naziv: fajlovi)
- Standardni C UI operacije ne smatra svojim delom
  - Postoje mnoge razlike: operativni sistem, podrška UI, i sl.
  - Postoje biblioteke sa funkcijama i makroima za UI
- Biblioteka C funkcija za UI:
  - UI na nivou toka (viši nivo) –
    - skrivaju se detalji fizičkih uređaja,
    - lakša prenosivost na druga radna okruženja
    - Baferovanje se vrši automatski
  - UI na sistemskom nivou (niži nivo)
    - Blisko vezan sa detaljima konkretne implementacije
    - Za baferovanje ste odgovorni sami

UI na nivou toka

- Standardni tok (*stream*)
  - Model toka je izveden iz UNIX radnog okruženja
  - C standard prepoznaje oba tipa tokova – **tekstualni** i **binarni tok**
  - UNIX-ov model i DOS-ov model standardnih tokova
- UNIX-ov model
  - Tri standardna toka:
    - stdin – standardni ulaz;
    - stdout – standardni izlaz;
    - stderr – standardni tok za greške;
- DOS-ov model
  - Tri standardna toka:
    - stdin, stdout, stderr
  - Još dva dodatna:
    - stdprn – standardni priključak na štampač;
    - stdaux – standardni priključak za serijsku komunikaciju;

UI na nivou toka (2)

- stdin
  - Standardno se vezuje za tastaturu
  - Baferuje u redove, što znači da proces ne prima ništa do pritiska na taster Enter
- stdout
  - Standardno, korisnikov ekran
  - Nebaferovan, da bi se izbeglo zadržavanje poruka o greškama u baferu



Fajlovi

- Fajl ili datoteka je skup bajtova kome je dodeljeno neko ime
- Čuvaju se na nekom spoljnjem uređaju (disk, traka i sl)
- Trajnije čuvanje,
- Brz i lak pristup podacima
- Tekstualni i binarni fajlovi
  - Razlika kod DOSa i UNIXa kod označavanja oznake kraja tekstualnog fajla: DOS ("\r\n") UNIX ("\n") - (carriage return, line feed)
- Preusmeravanje ulaza i izlaza (DOS, UNIX): **>** **<**
  - DOS/UNIX
    - Iz fajla na standardni izlaz

      more< imefajla
    - U fajl sa standardnog ulaza

      UNIX:

      cat> imefajla

      DOS

      copy con: imefajla

UI orijentisan na fajlove u C-u

- Da bi ste koristili spoljašnji fajl u C programu, morate mu pridružiti neki **tok**
- Ovo podrazmeva
  - deklarisanje promenjlive zvane **pokazivač na fajl**, i
  - davanje vrednosti toj promenljivoj
  - Vrednost koja se dodeljuje dobija se pozivom funkcije koja pokušava da otvori specificirani fajl
- Gledano sa strane korisnika, fajl ima **ime** i verovatno neki **sadržaj**
- Sa strane programa, fajl je **tok bajtova** kome se pristupa preko **pokazivača na fajl**
- Svim operacijama koje pristupaju sadržaju fajla ili koje obavljaju određene operacije nad fajlovima pristupa se preko pokazivača na fajl.

Standardna biblioteka funkcija za UI: stdio.h

- Sadrži tipove i funkcije za UI sa fajlovima
- Ako treba da ih koristite, obavezno uključite

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

- Tipovi
  - Pointer na fajl

```
FILE *fp;
```

Šta je tip "FILE *"? Praktično, ne morate da znate – dovoljno je da razmišljate o njemu kao apstraktnom tipu podataka kome pristupate preko C funkcija iz biblioteke.

Napomena: FILE predstavlja strukturu koja sadrži informacije o fajlu. Pri tome se mora koristiti pointer FILE *, pošto neke funkcije menjaju sadržaj nekih elemenata strukture

- Funkcije
  - Rad sa fajlovima (čitanje ili upis) zahteva tri koraka:
    - **Otvaranje datoteke (pristup).**
    - **Čitanje ili upis.**
    - **Zatvaranje datoteke.**

  Nadalje su opisane funkcije za sva tri navedena koraka.

stdio.h Predefinisani tokovi, tipovi i vrednosti

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

stdio.h Predefinisani tokovi, tipovi i vrednosti (2)

Primer: Korišćenje standarnih tokova

- *Standardni ulazni tok* koristite kada navodite scanf(), odnosno

  scanf("%d", &val);

  je ekvivalentno sa sledećim korišćenjem fscanf():

  fscanf(stdin, "%d", &val);

- *Standardni izlazni tok* je onaj koji koristite kod printf(), odnosno:

  printf("Vrednost = %d\n", val):

  je ekvivalentno sa sledećim korišenjem fprintf():

  fprintf(stdout, "Vrednost = %d\n", val):

  Napomena: standardni ulazni tok je povezan sa tastaturom, a standardni izlazni tok sa ekranom, osim ako nije urađena redirekcija!

- *Standardni tok za greške* omogućava da na njega šaljete greške:

  fprintf(stderr, "Ne mogu da otvorim ulazni fajl in.list!\n");

  Ovaj tok je asociran sa standardnim izlaznim tokom. Međutim, ako ste uradili redirekciju za izlazni tok, to ne važi za tok za greške!

stdio.h Otvaranje fajla (pristup)

- Za otvaranje fajlova mogu se koristiti dve funkcije
  - **fopen**
  - **freopen**

- Korišćenje fopen

  fp = fopen(*imefajla*, *mod*);

Gde je:

- *imefajla* je string koji sadrži ime fajla na disku (uključujući i putanju).
- *mod* je string koji predstavlja način otvaranja/pristupa fajlu. Uglavnom ćete koristiti fajlove za čitanje "r" ili upis "w".
- fopen() vraća **FILE *** koji se koristi za pristup fajlu.
- Ako fajl ne može da se otvori iz nekog razloga (privilegije, fajl ne postoji i sl), fopen() vraća **NULL**.
- Postoje i drugi modovi za pristup, napr.
  - za nadovezivanje (*append)* ("a") na kraj fajla bez gubitka prethodno sačuvanih informacija u fajlu
  - Za pristup i za čitanje i za pisanje i sl.

stdio.h Otvaranje fajla (pristup) (2)

- FILE *fopen(const char *path, const char *mode)
  - fopen otvara fajl sa navedenim imenom (string *path koji sadrži pored imena i putanju do fajla), i asocira ga sa odgovarajućim tokom (stream).
  - mode je string koji može da ima sledeće vrednosti:
    - **r** – otvara postojeći fajl za čitanje, počinje od početka fajla.
    - **w** – kreira novi fajl (ako ne postoji) ili briše sadržaj postojećeg i otvara ga za upis, od početka fajla.
    - **a** – otvara ili kreira fajl za dodavanje na kraj tekstualnog fajla.
    - **r+** - otvara fajl za čitanje i upis, počev od startne pozicije (ažuriranje).
    - **w+** - isto kao w, s tim da se fajl otvara za čitanje i upis (ažuriranje).
    - **a+** - isto kao a, s tim da se fajl otvara za čitanje i upis (ažuriranje).
  - String mode može da sadrži i b kao drugi ili treći karakter, da označi da se radi o binarnom fajlu.
  - String mode može da sadrži i druge karaktere koji se mogu koristiti u procesu implementacije.
  - Ako se fajl otvara za ažuriranje (mod +), operacija izlaza (upisa) ne može da sledi iza operacije ulaza (čitanje) bez operacije flushing-a za bafer (fflush())) ili repozicioniranja (fseek(), fsetpos, rewind); takođe, operacija čitanja ne može da sledi iza operacije upisa bez flush-inga bafera ili repozicioniranja osim ako ste dostigli kraj fajla.
- Ako je fopen uspešno otvorio fajl, vraća pointer na FILE. Inače vraća NULL i setuje se errno.

stdio.h Otvaranje fajla (pristup) (3)

- FILE *freopen(const char *pathname, const char *mode, FILE *stream)
  - freopen radi kao fopen s tim da asocira fajl sa nekim tokom, umesto da kreira novi tok.
  - freopen se koristi primarno da se fajl asocira sa nekim standardnim tokom za tekstualne podatke (stdin, stdout, ili stderr).

Primer:

```
FILE *ifp, *ofp;
char *mode = "r";
char outputFilename[] = "out.list";
ifp = fopen("in.list", mode);
if (ifp == NULL)
{
        fprintf(stderr, "Ne mogu da otvorim ulazni fajl in.list!\n");
        exit(1);
}
ofp = fopen(outputFilename, "w");
if (ofp == NULL) {
        fprintf(stderr, " Ne mogu da otvorim izlazni fajl %s!\n", outputFilename);
         exit(1);    }
```

Napomena:

Fajl koji otvarate za čitanje ("r") mora da postoji.

Kod upisa ("w"), ako pokušate da otvorite fajl koji ne postoji, kreiraće se novi sa zadatim imenom. Ako postoji takav fajl, njegov kompletan sadržaj će biti obrisan!!

stdio.h Čitanje i upis

- Nakon uspešnog otvaranja fajla možete da:
  - čitate sadržaj korišćenjem
    fscanf()
  - upisujete vrednosti korišćenjem
    fprintf().
    Ove funkcije rade na isti način kao scanf() i printf(), osim što zahtevaju dodatni parametar FILE * koji specificira fajl u koji pišete ili iz koga čitate podatke.
- Postoje druge funkcije u stdio.h za upis i čitanje!

Dostizanje kraja fajla:

- Funkcija **fscanf()**, kao i scanf(), normalno vraća broj vrednosti koje može da pročita.
- Ako dođe do kraja fajla, vraća specijalnu vrednost EOF. Obrada celog fajla može da ide u petlji, gde je uslov za izlazak da li ste došli do EOF
- Nedostatak: šta ako dođe do greške u formatu podataka, pa se umesto očekivanog slova pročita broj. Tada fscanf() ne može da pročita tu liniju i samim tim ne prelazi na sledeću – u ovom slučaju fscanf() neće vratiti EOF....
- Ovakve greške prouzrokuju probleme tipa kako pročitati ostatak fajla ili mogu da prouzrokuju beskonačne petlje.

stdio.h Čitanje i upis (2)

Primer: Čitanje iz fajla

Pročitati iz datoteke imena studenata (max. dužine 8) i rezultat sa ispita i povećati rezultat svakog za 10 poena

Datoteka:

```
in.list
------
Pera 70
Mlka 98
Aca  A+
...
```

Rešenje (deo koda):

```
char ime[9]; /* Ime plus oznaka kraja stringa */
int rezultat;
...
while (fscanf(ifp, "%s %d", ime, &rezultat) != EOF)        {
      fprintf(ofp, "%s %d\n", ime, rezultat+10);     } ...
```

- Jedno rešenje problema kod čitanja podataka pogrešnog formata je testiranje broja vrednosti koje treba da pročitamo sa fscanf(). Pošto je naveden format (u primeru) "%s %d", očekuje se čitanje dve vrednosti pa bi uslov provere bio:

```
while (fscanf(ifp, "%s %d", ime, &rezultat) == 2)
```

stdio.h Čitanje i upis (2)

- Drugi način:
  - korišćenje funkcije **feof()** iz biblioteke stdio.h.
- **feof()** kao argument ima pointer na fajl a vraća true/false u zavisnosti od toga da li smo dostigli kraj fajla.

Primer korišćenja:

```
while (!feof(ifp))          {
        if (fscanf(ifp, "%s %d", ime, &rezultat) != 2)
                break;
        fprintf(ofp, "%s %d", ime, rezultat+10); }
```

Napomena:

Kao i kod testiranja **!= EOF**, ova funkcija može prouzrokovati beskonačnu petlju ako je format ulaznog podatka iz fajla neodgovarajući.

Može se dodati i kod za proveru da li su pročitane dve vrednosti.

stdio.h Spisak funkcija za čitanje i upis (1)

Čitanje karaktera iz fajla - getc, fgetc i getchar

- int fgetc(FILE *stream)
  - Čita sledeći karakter iz ulaznog toka i vraća ga kao int (kod karaktera)
- int getc(FILE *stream)
  - Radi kao fgetc s tim da je obično implementiran kao makro
- int getchar(void)
  - Čita sledeći karakter sa stdin; obično implementiran kao getc(stdin) (što znači kao getc, tj. kao makro)

Upis karaktera u fajl - putc, fputc i putchar

- int fputc(int c, FILE *stream)
  - Upisuje c na izlazni tok (stream) kao unsigned char i vraća karakter kao int. Ako dođe do greške, vraća se EOF i setuje se errno.
- int putc(int c, FILE *stream)
  - Identična kao fputc ali implemetirana kao makro
- int putchar(int c)
  - Upisuje c na stdout, implementirana kao putc(stdout) (makro)

stdio.h Spisak funkcija za čitanje i upis (2)

Čitanje stringa iz fajla - fgets i gets

- char *fgets(char *s, int n, FILE *stream)
  - Čita karaktere iz toka stream i smešta ih u string na koji ukazuje s.
  - Čitanje se prekida kod *newline* karaktera, dostizanja kraja fajla (end-of-file) ili je pročitano n-1 karaktera, i oznaka kraja stringa '\0' je dodata stringu s (nakon svakog *newline* karaktera)
  - Ako je dostignut kraj fajla, bez pročitanog karaktera, fgets vraća NULL i sadržaj stringa s ostaje nepromenjen.
  - Ako dođe do greške tokom čitanja, vraća NULL i sadržaj stringa s je nedefinisan.
  - Inače, vraća s (pointer na pročitani string)
- char *gets(char *s, FILE *stream)
  - Slično kao fgets, ali mnogo opasniji !!
    - Ne pamti *newline* karakter
    - Važno: gets podrazumeva da je s neodređeno velik string, što može da izazove dosta problema (svesno ili nesvesno).

stdio.h Spisak funkcija za čitanje i upis (3)

Upis stringa u fajl - fputs i puts

- int fputs(const char *s, FILE *stream)
  - Upisuje null-terminated string s na izlazni tok stream.
  - Ako dođe do greške, vraća EOF.
  - Inače vraća ne-negativni ceo broj.
- int puts(const char *s)
  - Upisuje null-terminated string s, iza koga sledi newline karakter, na standardni izlazni tok stdout.

Čitanje iz binarne datoteke - fread

- size_t fread(void *ptr, size_t siz, size_t num, FILE *stream)
  - Čita do najviše num objekata, gde je svaki veličine siz bajtova, sa ulaznog toka stream, i smešta ih u memorijsku lokaciju na koju ukazuje ptr.
  - Vraća broj pročitanih objekata.
  - Ako dođe do greške vraća nulu.
  - Ako dođe do kraja fajla (end-of-file), vrednost koju vrati biće manja od num (može biti i nula), pri čemu se mogu koristiti feof ili ferror da bi se video razlog greške)

stdio.h Spisak funkcija za čitanje i upis (4)

Upis u binarni fajl - fwrite

- size_t fwrite(const void *ptr, size_t siz, size_t num, FILE *stream)
  - Upisuje do num objekata, gde je svaki veličine siz bajtova, iz memorijske lokacije na koju ukazuje ptr na izlazni tok stream.
  - Vraća broj upisanih objekata.
  - Ako dođe do greške, vraća nulu.

Upis formatiranog izlaza - printf, fprintf, sprintf

- int printf(const char *format, ...)
  - Piše na standardni izlazni tok stdout.
- int fprintf(FILE *stream, const char *format, ...)
  - Piše na izlazni tok stream.
- int sprintf(const char *str, const char *format, ...)
  - sprintf "piše" svoj izlaz u string str (završen oznakom kraja '\0'.)

Sve tri funkcije vraćaju broj upisanih karaktera (bez '\0' za sprintf)

Čitanje formatiranog ulaza - scanf, fscanf, sscanf

- int scanf(const char *format, ...)
- int fscanf(FILE *stream, const char *format, ...)
- int sscanf(const char *str, const char *format, ...)

stdio.h Spisak funkcija za čitanje i upis (5)

Provera statusa fajla - feof, ferror i clearerr

- int feof(FILE *stream)
  - Proverava indikator kraja fajla (end-of-file) za navedeni tok stream i vraća broj različit od nule ako je setovan.
  - Napomena: oznaka kraja fajla se nalazi iza poslednjeg karaktera u fajlu.
- int ferror(FILE *stream)
  - Proverava indikator greške za tok stream i vraća broj različit od nula ako je setovan.
- void clearerr(FILE *stream)
  - Briše vrednosti indikatora za kraj fajla i indikatora za greške za navedeni tok stream.
  - Napomena: kada se jednom setuju ova dva indikatora, oni se ne resetuju sve do poziva clearerr (osim ako se repozicioniranjem ne obriše oznaka kraja fajla tj. vrednost odgovarajućeg indikatora.)

stdio.h Spisak funkcija za čitanje i upis (6)

Određivanje pozicije u fajlu - fgetpos, fsetpos, rewind, fseek, i ftell

- int fgetpos(FILE *stream, fpos_t *pos);
  - Smešta vrednost tekuće pozicije indikatora za tok stream u pos.
  - pos je *implementation-defined* tip koji može da ima i kompleksnu strukturu.
  - Ako dođe do greške, vraća broj veći od nula i setuje se errno.
- int fsetpos(FILE *stream, fpos_t *pos)
  - Postavlja indikator pozicije u fajlu za stream na poziciju definisanu u pos.
  - Ako dođe do greške, vraća broj veći od nula i setuje se errno.
  - Kod uspešnog izvršenja, indikator end-of-file se briše.
- void rewind(FILE *stream)
  - Postavlja indikator pozicije na početak fajla.
- int fseek(FILE *stream, long offset, int whence)
  - Postavlja indikator pozicije za stream. Nova pozicija se određuje dodavanjem vrednosti pomeraja offset u odnosu na poziciju zadatu u whence:
    - Ako je whence postavljeno na **SEEK_CUR**, pomeraj se računa u odnosu na tekuću poziciju u fajlu.
    - Ako je whence postavljeno na **SEEK_SET**, pomeraj se računa u odnosu na početak fajla.
    - Ako je whence postavljeno na **SEEK_END**, pomeraj se računa u odnosu na kraj fajla.

    SEEK_CUR, **SEEK_SET**, i **SEEK_END** su definisani u <stdio.h>.
  - fseek se obično primenjuje na binarne fajlove.
- long ftell(FILE *stream)
  - Vraća tekuću poziciju u fajlu za stream.
  - Za binarne fajlove, vrednost koju vrati je broj bajtova od početka fajla do tekuće pozicije.
  - Za tekstualne fajlove, vrednost je *implementation-defined.*

stdio.h Zatvaranje fajla

- Funkcija za zatvaranje: fclose()
- int fclose(FILE *stream)
  - Baferovani sadržaj se upisuje na disk i zatvara se tok stream.
  - Naknadno korišćenje istog toka stream bez prethodnog freopen prouzrokuje grešku.
  - Ako je operacija zatvaranja uspešna, fclose vraća 0. Inače, vraća EOF i postavlja se kod greške u errno.

Primer (dodatak koda za preth. primer): zatvaranje ulaznog i izlaznog fajla:

```
fclose(ifp);
fclose(ofp);
```

Napomena:

- **Fajl mora biti zatvoren nakon korišćenja!**

Zatvaranje fajla je veoma važno, naročito za izlazne fajlove !!

Razlog je što je izlaz baferovan. Ako hoćete da u C-u nešto upišete u fajl, npr:

```
fprintf(ofp, “Nesto pisem!\n");
```

ne znači da će se navedeni sadržaj upisati u fajl na disku računara, već da će taj sadržaj završiti u baferu u memoriji koji se koristi za pristup fajlu!

Taj bafer čuva tekst privremeno, do zatvaranja fajla.

Nakon zatvaranja fajla, sadržaj bafera se upisuje na disk.

Primeri: UI sa fajlovima

Čitanje iz fajla i konverzija malih u velika slova

```
#include<stdio.h>
int main(int argc, char** argv)
{
   char* imefajla = "upper.c";
   FILE* fptr;  char c;
   if (argc > 1)
        imefajla = argv[1];
        fptr = fopen(imefajla,"r");
    while((c = getc(fptr)) != EOF)
     {
         if (islower(c))
             c = toupper(c);
        printf("%c",c);
     }
}
```

Dvostruki prored

```
#include<stdio.h>
int main(int argc, char** argv) {
   char* ulazni_fajl = "upper.c";
   char* izlazni_fajl = "double.txt";
   FILE* iptr;
   FILE* optr;
   char c;
   if (argc > 1)
        ulazni_fajl = argv[1];
    if (argc > 2)
        izlazni_fajl = argv[2];
   iptr = fopen(ulazni_fajl,"r");
   optr = fopen(izlazni_fajl,"w");
   while((c = getc(iptr)) != EOF) {
         putc(c,optr);
         if (c == '\n')
         putc(c,optr);
         }
    fclose(iptr);
    fclose(optr); }
```

Obrada podataka iz fajla

```
#include<stdio.h>
int main() {
   FILE* iptr;
   char c;
   int n1, n2;
   iptr = fopen("access.dat","r");
   while((c = getc(iptr)) != EOF) {
   switch(c) {
        case '+':
              fscanf(iptr,"%d %d",&n1,&n2);
              printf("process %d pocinje, size %d\n",n1,n2);
              break;
        case 'r':
              fscanf(iptr,"%d %d",&n1,&n2);
              printf("process %d cita stranu %d\n",n1,n2);
              break;
        case 'w':
              fscanf(iptr,"%d %d",&n1,&n2);
              printf("process %d upisuje %d\n",n1,n2);
              break;
        case '-':
              fscanf(iptr,"%d",&n1);
              printf("process %d se zavrsava\n",n1);      break;     } }}
```