Programiranje 1

Milena Vujošević Janičić

www.matf.bg.ac.rs/~milena

Ulaz i izlaz programa

Pregled

Pregled

Argumenti komandne linije

- Jedan način da se određeni podaci proslede programu je da se navedu u komandnoj liniji prilikom njegovog pokretanja.

```
./a.out argument1 argument2 argument3
```

- Argumenti koji su tako navedeni prenose se programu kao argumenti funkcije `main`.
- Da bi `main` prihvatila argumente, potrebno je da se definiše na sledeći način

```
int main(int argc, char* argv[]) {
...
}
```

Argumenti komandne linije

- Prvi argument (koji se obično naziva `argc`, od engleskog *argument count*) je broj argumenata komandne linije (uključujući i sâm naziv programa) navedenih prilikom pokretanja programa.
- Drugi argument (koji se obično naziva `argv`, od engleskog *argument vector*) je niz niski karaktera koje sadrže argumente — svaka niska direktno odgovara jednom argumentu.
- Identifikatori `argc` i `argv` su proizvoljni i funkcija `main` može biti deklarisana i na sledeći način:

```
int main (int br_argumenata, char* argumenti[]);
```

Argumenti komandne linije

- Nazivu programa odgovara niska `argv[0]`.
- `argv[1]` do `argv[argc-1]` su tekstovi argumenata programa, a element `argv[argc]` sadrži vrednost `NULL`.
- Ako je `argc` tačno 1, onda nisu navedeni dodatni argumenti nakon imena programa.

Primer

```
#include <stdio.h>
int main(int argc, char* argv[]) {
   int i;
   printf("argc = %d\n", argc);

   /* Nulti argument uvek je ime programa
      (na primer, a.out) */
   for (i = 0; i < argc; i++)
       printf("argv[%d] = %s\n", i, argv[i]);
   return 0;
}
```

Argumenti komandne linije

- Pre upotrebe argumenata komandne linije važno je proveriti da li je program ispravno pokrenut, tj da li postoji argument komandne linije koji želimo da koristimo. Provera da li je program ispravno pokrenut, tj sa dovoljnim brojem argumenata komandne linije, se vrši proverom veličine argc

```
...
if(argc<3) {
   printf("Neispravno pokretanje programa\n");
   return 1;
}
printf("argv[1]= %s argv[2]= %s\n", argv[1], argv[2]);
```

Ulaz i izlaz

- Jezik C je dizajniran kao mali jezik i ulazno/izlazne operacije nisu direktno podržane samim jezikom, već specijalizovanim funkcijama iz standardne biblioteke jezika
- Standardna biblioteka ima garantovanu prenosivost
- Potrebno je uključiti zaglavlje `<stdio.h>`

Standardni tokovi

- Standardna biblioteka implementira jednostavan model tekstualnog ulaza i izlaza.
- Ulaz i izlaz se modeluju tzv. *tokovima* (engl. stream) podataka (obično pojedinačnih bajtova ili karaktera).
- *Standardni ulaz* obično čine podaci koji se unose sa tastature.
- Podaci koji se upućuju na *standardni izlaz* se obično prikazuju na ekranu.
- Postoji i *standardni izlaz za greške* na koji se obično upućuju poruke o greškama nastalim tokom rada programa i koji se, takođe, obično prikazuje na ekranu.

Redirekcija

- U mnogim okruženjima, moguće je izvršiti preusmeravanje (redirekciju) standardnog ulaza tako da se, umesto sa tastature, karakteri čitaju iz neke datoteke.
- Na primer, ukoliko se program pokrene sa

  `./prog < infile`

  onda program `prog` čita karaktere iz datoteke `infile`, umesto sa tastature.

Redirekcija

- Takođe, mnoga okruženja omogućavaju da se izvrši preusmeravanje (redirekcija) standardnog izlaza u neku datoteku.
- Na primer, ukoliko se program pokrene sa

  `./prog > outfile`

  onda program `prog` upisuje karaktere u datoteku `outfile`, umesto na ekran.

Redirekcija

- Ukoliko bi se poruke o greškama štampale na standardni izlaz, zajedno sa ostalim rezultatima rada programa, onda, u slučaju preusmeravanja standardnog izlaza u datoteku, poruke o greškama korisniku ne bi bile prikazane na ekranu i korisnik ih ne bi video.
- Ovo je glavni razlog uvođenja standardnog izlaza za greške koji se obično prikazuje na ekranu.
- Moguće je izvršiti redirekciju i standardnog izlaza za greške u datoteku, na primer:

```
./prog 2> errorfile
```

Funkcije za ulaz/izlaz jednog karaktera

- Funkcija `int getchar(void);` vraća sledeći karakter sa ulaza, svaki put kada se pozove, ili `EOF` kada dođe do kraja toka.
- Funkcija `int putchar(int c);` štampa karakter `c` na standardni izlaz, a vraća karakter koji je ispisala ili `EOF` ukoliko je došlo do greške.

Linijski ulaz/izlaz

- Bibliotečka funkcija `char* gets(char* s);` čita karaktere sa standardnog ulaza do kraja tekuće linije ili do kraja datoteke i karaktere smešta u nisku s.
- Oznaka kraja reda se ne smešta u nisku, a niska se automatski završava nulom.
- Ne vrši se nikakva provera da li niska `s` sadrži dovoljno prostora da prihvati pročitani sadržaj i ovo funkciju `gets` čini veoma opasnom za korišćenje.
- U slučaju da je ulaz uspešno pročitan, `gets` vraća `s`, a inače vraća `NULL` pokazivač.

Linijski ulaz/izlaz

- Bibliotečka funkcija `int puts(const char* s);` ispisuje nisku na koju ukazuje `s` na standardni izlaz, dodajući pri tom oznaku za kraj reda.
- Završna nula se ne ispisuje.
- Funkcija `puts` vraća `EOF` u slučaju da je došlo do greške, a nenegativnu vrednost inače.

Formatiran ulaz/izlaz

- Funkcije `printf` i `scanf`
- Funkcija `printf` prevodi vrednosti osnovnih tipova podataka u serije karaktera i usmerava ih na standardni izlaz. Funkcija vraća broj odštampanih karaktera.
- Format niska sadrži dve vrste objekata: obične karaktere, koji se doslovno prepisuju na standardni izlaz i specifikacije konverzija od kojih svaka uzrokuje konverziju i štampanje sledećeg uzastopnog argumenta funkcije `printf`.
- Potrebno je poznavati pravila formiranja format niske i značenja konverzionih karaktera

Formatiran ulaz/izlaz

- Funkcija `scanf` čita karaktere sa standardnog ulaza, interpretira ih na osnovu specifikacije navedene format niskom i smešta rezultat na mesta određena ostalim argumentima
- Funkcija `scanf` prestaje sa radom kada iscrpi svoju format nisku ili kada neki deo ulaza ne može da se uklopi u shemu navedenu format niskom.
- Funkcija vraća broj uspešno uklopljenih i dodeljenih izlaznih podataka.
- U slučaju kraja datoteke, funkcija vraća `EOF`.
- Vrednost 0 se vraća u slučaju da tekući karakter sa ulaza ne može da se uklopi u prvu specifikaciju zadatu format niskom.

Formatiran ulaz/izlaz

- Ulaz iz niske i izlaz u nisku: funkcije `sprintf` i `sscanf`
- Standardna biblioteka jezika C definiše funkciju `sprintf` koja je veoma slična funkciji `printf`, ali rezultat njenog rada je popunjavanje niske karaktera formatiranim tekstom:

```
int sprintf(char *string, const char *format, arg1, arg2
```

- Standardna biblioteka jezika C definiše i funkciju `sscanf` koja je analogna funkciji `scanf`, osim što ulaz čita iz date niske karaktera, umesto sa standardnog ulaza.

```
int sscanf(const char* input, char* format, ...)
```

Datoteke

- Šta je datoteka?

Datoteke

- Šta je datoteka? Niz bajtova.
- Prednosti u odnosu na preusmeravanje
- Koje vrste datoteka postoje?

Datoteke

- Šta je datoteka? Niz bajtova.
- Prednosti u odnosu na preusmeravanje
- Koje vrste datoteka postoje?
- Binarne i tekstualne datoteke
- U zavisnosti od sadržaja datoteke (binarna ili tekstulna) razlikujemo i dva načina pristupa datotekama.

Tekstualne datoteke

- Sadržaj tekstualne datoteke čine vidljivi karakteri, sa dodatkom oznake kraja reda i horizontalnog tabulatora.
- Tekstualne datoteke se obično obrađuju liniju po liniji, pa je oznaka za kraj linije veoma relevantna i značajna
- Na različitim sistemima tekst se kodira na različite načine i na nekim sistemima kraj reda u tekstualnoj datoteci se zapisuje sa dva karaktera (na primer, sa `\r\n` na sistemima Windows) a na nekim sa samo jednim karakterom (na primer, sa `\n` na sistemu Linux).

Tekstualne datoteke

- Kako programer ne bi morao da se stara o ovakvim specifičnostima, C jezik nudi *tekstualni mod* rada sa datotekama: prilikom svakog čitanja i upisa u datoteku vrši se konverzija iz podrazumevanog formata označavanja kraja reda u jedinstven karakter `\n`.
- Tako će na Windows sistemima dva karaktera `\r\n` na kraju reda biti pročitana samo kao `\n` i, obratno, kada se u datoteku upisuje `\n` biće upisana dva karaktera `\r\n`.

Binarne datoteke

- Drugi način pristupa datotekama prilagođen je datotekama čiji sadržaj ne predstavlja tekst i u kojima se mogu naći bajtovi svih vrednosti od 0 do 255 (na primer, `jpg` ili `zip` datoteke).
- Za obradu ovakvih datoteka, jezik C nudi *binarni mod* rada sa datotekama.
- U ovom slučaju nema nikakve konverzije i interpretiranja karaktera prilikom upisa i čitanja i svaki bajt koji postoji u datoteci se čita doslovno.

fopen

- Sve potrebne deklaracije i za rad sa datotekama nalaze u zaglavlju `<stdio.h>`.
- Da bi se pristupilo datoteci, bilo za čitanje, bilo za pisanje, potrebno je izvršiti određenu vrstu povezivanja datoteke i programa. Za ovo se koristi bibliotečka funkcija `fopen`:
  `FILE* fopen(const char* filename, const char* mode);`
- Funkcija `fopen` dobija nisku koja sadrži ime datoteke i uz pomoć usluga operativnog sistema vraća pokazivač na strukturu koja predstavlja sponu između lokalne datoteke i programa i koja sadrži sve potrebne informacije o datoteci.
- Programer treba da čuva pokazivač na strukturu `FILE` i da ga prosleđuju svim funkcijama koje treba da pristupaju datoteci.

fopen

- Drugi argument je niska karaktera koja predstavlja način (mod) otvaranja datoteke i koja ukazuje na to kako će se datoteka koristiti.
- Dozvoljeni modovi uključuju čitanje (read, `"r"`), pisanje (write, `"w"`) i dopisivanje (append, `"a"`).
- Ako se datoteka koja ne postoji na sistemu otvara za pisanje ili dopisivanje, onda se ona kreira
- U slučaju da se postojeća datoteka otvara za pisanje, njen stari sadržaj se briše, dok se u slučaju otvaranja za dopisivanje stari sadržaj zadržava, a novi sadržaj upisuje nakon njega.

fopen

- Ukoliko se pokušava čitanje datoteke koja ne postoji dobija se greška. Takođe, greška se javlja i u slučaju da se pokušava pristup datoteci za koju program nema odgovarajuće dozvole.
- U slučaju greške funkcija `fopen` vraća `NULL`.
- Modovi `r+`, `w+` i `a+` ukazuju da će rad sa datotekom podrazumevati i čitanje i pisanje (ili dopisivanje).
- U slučaju da se želi otvaranje binarne datoteke, na mod se dopisuje `"b"` (na primer, `rb`, `wb`, `a+b`, ...).

fclose

- Kada se C program pokrene, operativni sistem otvara tri toka podataka (standardni ulaz, stadnardni izlaz i standardni izlaz za greške), kao da su datoteke i obezbeđuje pokazivače kojima im se može pristupati. Ti pokazivači se nazivaju:

```
FILE* stdin;
FILE* stdout;
FILE* stderr;
```

- Funkcija `int fclose(FILE *fp)` prekida vezu između programa i datoteke koju je funkcija `fopen` ostvarila.
- S obzirom na to da većina operativnih sistema ograničava broj datoteka koje mogu biti istovremeno otvorene, dobra je praksa zatvarati datoteke čim prestanu da budu potrebne.

Primer

```
int main()
{
     FILE *ulaz, *izlaz;
     ulaz = fopen("ulazna.txt","r");
     if(ulaz == NULL) {
       printf("Greska pri otvaranju ulazne datoteke\n");
         return -1;
     }

     izlaz = fopen("izlazna.txt","w");
     if(izlaz == NULL) {
         printf("Greska pri otvaranju izlazne datoteke\n");
         return -1;
     }
   ...

     fclose(ulaz);
     fclose(izlaz);
     return 0;
}
```

- Čitanje i pisanje sadržaja u datoteku može da se vrši na različite načine
- Za tekstualne datoteke, to može biti karakter po karakter, liniju po liniju ili ulaz-izlaz može biti formatiran
- Funkcija `int getc(FILE *fp)` vraća naredni karakter iz datoteke određene prosleđenim `FILE` pokazivačem ili `EOF` u slučaju kraja datoteke ili greške.
- Funkcija `int putc(int c, FILE *fp);` upisuje dati karakter u datoteku određenu prosleđenim `FILE` pokazivačem i vraća upisani karakter ili `EOF` u slučaju greške.

feof i ferror

- Funkcija `int feof(FILE *fp)` vraća vrednost *tačno* (ne-nula) ukoliko se došlo do kraja date datoteke.
- Funkcija `int ferror(FILE *fp)` vraća vrednost *tačno* (ne-nule) ukoliko se došlo do greške u radu sa datotekom.

fgets

- Funkcija
  `char *fgets(char *line, int maxline, FILE *fp)` čita jednu liniju iz datoteke (uključujući oznaku kraja reda) iz datoteke `fp` i rezultat smešta u nisku `line`.
- Može da bude pročitano najviše `maxline-1` karaktera. Rezultujuća niska završava se karakterom `'\0'`.
- U normalnom toku izvršavanja, funkcija `fgets` vraća pokazivač na početak linije, a u slučaju kraja datoteke ili greške vraća `NULL`.

fputs

- Funkcija `int fputs(const char *line, FILE *fp)` ispisuje nisku (koja ne mora da sadrži oznaku kraja reda) u datoteku.
- Ona vraća `EOF` u slučaju da dođe do greške, a neki nenegativan broj inače.
- Za razliku od funkcija `fgets` i `fputs`, funkcija `gets` briše završni `'\n'`, a funkcija `puts` ga dodaje.

Formatirani ulaz i izlaz

- Za formatirani ulaz i izlaz mogu se koristiti funkcije `fscanf` i `fprintf`.
- One su identične funkcijama `scanf` i `printf`, osim što je prvi argument `FILE` pokazivač koji ukazuje na datoteku iz koje se čita, odnosno u koju se piše.
- Format niska je u ovom slučaju drugi argument.

```
int fscanf(FILE *fp, const char *format, ...)
int fprintf(FILE *fp, const char *format, ...)
```

Primer

```
fscanf(stdin, "%d", &a);
<=>
scanf("%d", &a);

fprintf(stdout, "%d\n",a);
<=>
printf("%d\n",a);

fprintf(stderr, "Neispravan unos!\n");
```

Primer

```
int main()
{
     FILE *ulaz, *izlaz;
     ulaz = fopen("ulazna.txt","r");
     if(ulaz == NULL) {
       fprintf(stderr, "Greska pri otvaranju ulazne datoteke\n");
         return -1;
     }

     izlaz = fopen("izlazna.txt","w");
     if(izlaz == NULL) {
         fprintf(stderr, "Greska pri otvaranju izlazne datoteke\n");
         return -1;
     }
   ...

     fclose(ulaz);
     fclose(izlaz);
     return 0;
}
```

Binarne datoteke

- Funkcija `fread` se koristi za čitanje niza slogova iz binarne datoteke, a funkcija `fwrite` za pisanje niza slogova u binarnu datoteku.
- Funkcija `fseek` služi za pozicioniranje na mesto u datoteci sa koga će biti pročitan ili na koje će biti upisan sledeći podatak.
- Funkcija `ftell` vraća trenutnu poziciju u datoteci (u obliku pomeraja od početka izraženog u broju bajtova).
- Iako primena funkcija `ftell` i `fseek` nije striktno ograničena na binarne datoteke, one se najčešće koriste sa binarnim datotekama.

Literatura

Slajdovi su pripremljeni na osnovu materijala iz dvaneastog poglavlja knjige:
Filip Marić, Predrag Janičić: Programiranje 1
Za pripremu ispita nisu dovoljni slajdovi, potrebno je koristiti knjigu!