Programski jezici i strukture podataka

7

ULAZ, IZLAZ I DATOTEKE

Ulazno izlazne operacije

- Programi moraju imati mogućnost da upisuju podatke u datoteke ili da ih ispisuju na izlazni uređaj poput ekrana ili štampača, te đa učitavaju podatke iz datoteka ili sa ulaznih uređaja (npr. tastature).
- Deo standardne biblioteke je posvećen učitavanju i ispisivanju podataka, to jest ulazno-izlaznim operacijama (engl. I/O library).
- Osim funkcija iz standardne biblioteke, jezik C nema drugu podršku za ulazno-izlazne operacije.
- Sve osnovne funkcije, makroi i tipovi za ulazne i izlazne tokove podataka definisani su u datoteci zaglavlja *stdio.h.*
- Odgovarajuće deklaracije potrebne funkcijama za upis i ispis širokih znakova - tj. znakova tipa **wchar_t** - nalaze se u datoteci zaglavlja *wchar.h.*

Tokovi podataka

- Sa aspekta programa na jeziku **C,** datoteke i uređaji svih vrsta za ulazno-izlazne operacije predstavljaju se na isti način - kao **logički *tokovi podataka*** (engl. *data streams).*
- Tokovi podataka na jeziku **C** mogu biti ***tekstualni*** ili ***binarni,*** na nekim sistemima ta razlika ne postoji.
- Kada se datoteka otvori pomoću funkcije **fopen()** ili **tmpfile()** pravi se novi tok podataka, koji postoji dok ga ne zatvori funkcija **fclose()**.

Tokovi podataka

- Jezik C prepušta upravljanje datotekom izvršnom okruženju - sistemu na kome se program izvršava.
- Tok podataka predstavlja kanal kojim podaci mogu da se prenose od izvršnog okruženja do programa ili u obrnutom smeru.
- Uređajima, poput konzola, pristupa se na isti način kao datotekama.

Tekstualni tokovi podataka

- Tekstualnim tokovima podataka prenose se znakovi teksta koji je podeljen u redove.
- Red teksta sastoji se od sekvence znakova koja se završava znakom za novi red.
- Red teksta može biti prazan - tada sadrži samo znak za novi red.
- Poslednji preneti red ne mora imati znak za novi red na kraju, to zavisi od implementacije.

Tekstualni tokovi podataka

- Interno predstavljanje teksta je isto na svim sistemima na kojima se program izvršava.
- Učitavanje i ispisivanje teksta na određenom sistemu može podrazumevati uklanjanje, dodavanje ili menjanje pojedinih znakova.
- Kada se na sistemima koji se ne zasnivaju na Unixu učitavaju tekstualne datoteke, indikatori kraja reda obično se moraju konvertovati u znakove za novi red: primer je Windows, u kome je indikator kraja reda sekvenca dva upravljačka znaka, \r (znak za prelazak na početak novog reda) i \n (znak za novi red).
- Upravljački znak ^Z (znakovni kod 26) u tekstualnom toku podataka na Windowsu označava kraj toka.

Binarni tokovi podataka

- Binarni tok podataka je sekvenca bajtova koji se prenose bez izmena.
- Kada ulazno-izlazne funkcije rade s binarnim tokovima podataka, uopšte ne prepoznaju upravljačke znakove.
- Podatak upisan u datoteku pomoću binarnog toka podataka može se uvek neizmenjen učitati na istom sistemu.
- Ako program učitava sadržaj tekstualne datoteke pomoću binarnog toka, tekst se u programu pojavljuje u obliku u kome je sačuvan, sa svim upravljačkim znacima koji se koriste na datom sistemu.

Datoteke

- Datoteka predstavlja sekvencu bajtova.
- Funkcija fopen() povezuje datoteku s tokom podataka i inicijalizuje objekat tipa FILE, koji sadrži sve informacije neophodne za upravljanje tokom podataka.
- U takve informacije spadaju **pokazivač na korišćeni bafer**, **indikator pozicije** u datoteci koji određuje mesto pristupa u datoteci, i indikatori (engl. flags) **greške** i **kraja datoteke**.

Datoteke

- Prenošenje tokova podataka u odnosu na bafer može se odvijati na tri načina:
  - **Po punjenju bafera** (engl. fully buffered)
    Znakovi u baferu se normalno prenose samo ako je bafer pun.
  - **Posle znaka za novi red** (engl. line buffered)
    Znaci u baferu se normalno prenose samo ako se u bafer upiše znak za novi red ili ako je pun. Sadržaj bafera toka podataka se upisuje u datoteku i ako program zahteva upis preko toka van bafera (engl. unbuffered stream) ili kada se zbog zahteva za upis toka podataka na ovakav način učitavaju znakovi iz sistemskog okruženja.
  - **Izvan bafera** (engl. unbuffered.)
    Znakovi se prenose što je brže moguće.

Sfandardni tokovi podataka

Pokazivač	Naziv	Upis
stdin	Standardni ulaznitok	Posle znaka za novi red
stdout	Standardni izlaznitok	Posle znaka za novi red
stderr	Standardni izlazni tok greške	Van bafera

- Tok **stdin** obično je pridružen tastaturi, a tokovi **stdout** i **stderr** ekranu konzole.
- Te veze mogu se izmeniti takozvanim reusmeravanjem (engl. *redirection):* to može uraditi funkcija **freopen()** kada je program pozove ili okruženje u kome se program izvršava.

Otvaranje i zatvaranje datoteka

- Da biste upisali sadržaj u novu datoteku ili izmenili sadržinu postojeće, morate je prvo otvoriti.
- Prilikom otvaranja datoteke, morate navesti *režim pristupa* (engl. *access mode)* kako biste ukazali na to da li planirate da čitate njen sadržaj i/ili da upisujete u nju.
- Kada završite rad s datotekom, zatvorite je da biste oslobodili resurse.

Otvaranje datoteke

- Standardna datoteka sadrži funkciju **fopen()** za otvaranje datoteke. U posebnim slučajevima, datoteka se otvara pomoću funkcija **freopen(**) i **tmpfile()**.

```
FILE *fopen( const char * restrict imedat, const char * restrict rezim );
```

- Ova funkcija otvara datoteku čije ime predstavlja znakovni niz ***imedat***. Ime datoteke možeda sadrži i ime direktorijuma.
- Drugi argument, ***rezim,*** takođe je znakovni niz i označava režim pristupa.
- Funkcija fopen() povezuje datoteku s novim tokom podataka imedat.

Otvaranje datoteke

```
FILE *freopen( const char * restrict imedat,
        const char * restrict rezim,
                FILE * restrict tok );
```

- Ova funkcija preusmerava tok podataka.
- Poput funkcije fopen(), otvara određenu datoteku u zadatom režimu.
- Funkcija freopen() povezuje datoteku s postojećim tokom podataka određenim trećim argumentom.
- Datoteka koja je prethodno bila povezana sa tim tokom, zatvara se.
- Funkcija freopen() najčešće se koristi za preusmeravanje standardnih tokova stdin, stdout i stderr.

Otvaranje datoteke

```
FILE *tmpfile( void );
```

- Funkcija tmpfile() pravi novu privremenu datoteku, imena različitog od svih postojećih datoteka, i otvara je zbog binarnog upisa i čitanja (kao kada bi se funkcija fopen() pozvala pomoću znakovnog niza režima pristupa "wb+").
- Ako se program normalno završi, datoteka se automatski briše.

Otvaranje datoteke

- Sve tri funkcije za otvaranje datoteke vraćaju pokazivač na otvoreni tok u slučaju uspeha, ili pokazivač na vrednost NULL, u suprotnom.

Režim pristupa

- Režim pristupa određen drugim argumentom funkcije fopen() ili freopen() označava koje su ulazne i izlazne operacije dozvoljene nad novim tokom podataka.
- Dozvoljene vrednosti znakovnog niza za režim pristupa jasno su definisane: prvi znak znakovnog niza koji određuje režim pristupa je uvek:
  - **r** za čitanje („read“),
  - **w** za pisanje („write“), ili
  - **a** za dopisivanje („append“);

  u najjednostavnijem slučaju, znakovni niz sadrži samo jedan znak.

Režim pristupa

- Znakovni niz režima pristupa može da sadrži znak + ili b (ili oba, u proizvoljnom redosledu: sekvence +b i b+ imaju isti efekat).
- Znak plus (+) u znakovnom nizu režima pristupa kazuje da je dozvoljeno i čitanje i upis.
- Ipak, program ne može posle jedne od njih odmah izvršiti drugu.
- Nakon operacije upisivanja, prvo morate pozvati funkciju fflush() ili funkcije za pozicioniranje fseek(), fsetpos() ili rewind() pre operacije čitanja.
- Slično tome, posle operacije čitanja, pre upisivanja morate pozvati funkciju za pozicioniranje.

Režim pristupa

- **Znak b** u znakovnom nizu režima pristupa uslovljava otvaranje datoteke u binarnom režimu — to jest, novi tok koji se povezuje sa datotekom **binaran** je.
- Ako u znakovnom nizu režima pristupa **nema znaka b**, novi tok je **tekstualni**.

Režim pristupa

- Ukoliko znakovni niz režima pristupa počinje znakom r, neophodno je da **datoteka već postoji** u sistemu datoteka.
- Ako je na početku znakovnog niza slovo w i datoteka ne postoji, mora se napraviti.
- Ukoliko datoteka već postoji, njen sadržaj biće **izgubljen**, pošto funkcija fopen() veličinu datoteka u režimu upisivanja svodi na nulu.
- Ako znakovni niz režima upisa počinje znakom a („append“), a datoteka ne postoji - onda se mora napraviti.
- Ukoliko datoteka postoji, njen sadržaj će se očuvati, jer se sve operacije upisa automatski obavljaju na kraju datoteke.

```
ifinclude <stdio.h> ifinclude <stdbool.h>
_Bool isReadWriteable( const char *filename )
{

FILE *fp = fopen( filename, "r+" );// Otvara datoteku za upis i ispis.
if ( fp != NULL )      // Da li je fopen() uspešna?
{
       fclose(fp);     // Da: zatvara datoteku; bez obrade greške.
       return true;
}
else   // Ne.
       return false;
}
```

Datoteke

- Tekstualne
- Binarne

- Tok rada:
  - Otvoriti datoteku – `fopen`
  - Čitamo i/ili upisujemo podatke u datoteku
  - Zatvorimo datoteku – `fclose`

`FILE *fopen(const char *path, const char *mode);`

- **`path`** je naziv datoteke koja će biti otvorena
- **`mode`** je režim (**uvek** mora biti string) u kome će datoteka biti otvorena i može biti jedan od sledećih režima:
  - `"r"`: otvori postojeću tekstualnu datoteku u režimu čitanja.
  - `"w"`: otvori tekstualnu datoteku u režimu pisanja. Ako datoteka ne postoji biće kreirana, ili ako već postoji njen sadržaj će biti uništen.
  - `"a"`: otvori tekstualnu datoteku u režimu dodavanja novih podataka (pisanje nakon postojećeg kraja datoteke). Datoteka će biti kreirana ako ne postoji.
  - "r+" – otvori postojeću tekstualnu datoteku u režimu čitanja i pisanja.
  - "w+" – otvori tekstualnu datoteku u režimu pisanja i čitanja. Ako datoteka ne postoji biće kreirana, ili ako već postoji njen sadržaj će biti uništen.
  - "a+" - otvori tekstualnu datoteku u režimu dodavanja novih podataka (pisanje nakon postojećeg kraja datoteke) i čitanja. Ako datoteka ne postoji biće kreirana.

```
FILE *fopen(const char *path, const char
*mode);
```

- Ako je datoteka uspešno otvorena u zadatom režimu povratna vrednost će biti pokazivač na konkretni `FILE`. U suprotnom povratna vrednost će biti `NULL` i kod uzroka greške će biti postavljen u globalnu promenljivu `errno`.

```c
// Primer obrade grešaka pri otvaranju datoteke
#include<errno.h> // Upravljanje greskama ('errno')
#include<stdio.h>
#include<stdlib.h> // Potreban za 'exit()'

int main()
{
    FILE *fp = fopen("test.txt", "w");
    if(fp == NULL) { // Fatalna greska
        // Ispisi opis i uzrok greske
        perror("Can't open file test.txt");
        // Izadji iz programa, vrati OS-u kod uzroka greske
        exit(errno);
    }

    fputs("Zdravo svete\n", fp);

    fclose(fp);
    return 0;
}
```

```c
// Primer rada sa tekstualnom datotekom – čitanje
#include<stdio.h>
#define MAXL 80

int main()
{
    FILE *pf;
    char str[MAXL];

    pf=fopen("test.txt", "r");

    if(pf!=NULL){
        while(fgets(str, MAXL, pf)!=NULL)
            puts(str);

        fclose(pf);
    } else {
        printf("Nije moguce otvoriti datoteku ili ona ne postoji.");
    }

    return 0;
}
```

Rad sa tekstualnim datotekama

- Funkcijama za prenos znakova vrši se čitanje ili pisanje pojedinačnih znakova ili nizova znakova bez konverzije.
- int fgetc (FILE *dat) ;
- int getc   (FILE *dat) ;
- int getchar (void);

Rad sa tekstualnim datotekama - čitanje

- Ove funkcije čitaju jedan znak iz datoteke **dat** (fgetc i getc) odnosno preko glavnog ulaza (getchar).
- Dok je **fgetc** funkcija, dotle **getc** se ostvaruje kao makro.
- Zato, može da se desi da se u slučaju getc, izraz dat izračunava više puta, što može da predstavlja problem ako taj izraz proizvodi i bočne efekte.
- S druge strane, izvršavanje getc je efikasnije od pozivanja funkcije fgetc. U logičkom smislu, fgetc i getc su istovetne.

Rad sa tekstualnim datotekama - čitanje

- Vređnost sve tri funkcije je kod pročitanog znaka, odnosno simbolička konstanta EOF u slučaju nailaska na kraj datoteke ili u slučaju otkrivanja greške u toku čitanja.

Rad sa tekstualnim datotekama - pisanje

- fputc **(int *zn,*** FILE ***dat)***;
- **int** putc **(int*zn,*** FILE ***dat) ;***
- **int** putchar **(int** zn) ;
- Ove funkcije uspisuju jedan znak zn u datoteku dat (fputc i putc) odnosno ispisuju preko glavnog izlaza (putchar).
- Dok je **fputc** funkcija, dotle **putc** se ostvaruje kao makro. Zato, može da se desi da se u slučaju putc izraz dat izračunava više puta, što može da predstavlja problem ako taj izraz proizvodi i bočne efekte.
- S druge strane, izvršavanje putc je efikasnije odpozivanja funkcije fputc.
- U logičkom smislu, fputc i putc su istovetne.

Rad sa tekstualnim datotekama - ispis

- Vrednost sve tri funkcije je kod ispisanog znaka, odnosno simbolička konstanta EOF u slučaju otkrivanja greške u toku pisanja.
- Evo primera kojim se piše jedan znak u datoteku podaci:

```
greska = fputc ('A', podaci) == EOF;
```

Rad sa tekstualnim datotekama - čitanje stringa

```
char *fgets (char *tekst, int n, FILE *dat) ;
char * gets (char *tekst);
```

- Ove funkcije čitaju jedan red teksta u znakovni niz tekst iz datoteke dat (fgets) odnosno preko glavnog ulaza (gets).
- Funkcija fgets čitanje završava kada pročita n-1 znakova ili kad pročita znak za kraj reda \n i iza pročitanih znakova dodaje završni znak \0.
- Funkcija gets uvek čita sve znakove do znaka \n, koga ne stavlja u niz *tekst* već ga zamenjuje znakom \0.

Rad sa tekstualnim datotekama - čitanje stringa

- Vrednost obe funkcije je tekst (adresa!), odnosno NULL u slučaju nailaska na kraj datoteke ili u slučaju otkrivanja greške u toku čitanja.
- Primer kojim se čita jedan red teksta u znakovni niz tekst (tip *string*) iz datoteke podaci:

```
greska = fgets (recenica, 55, podaci) == NULL;
```

Rad sa tekstualnim datotekama - pisanje stringa

```
int fputs   (const char *tekst, FILE *dat);
int puts    (const char *tekst);
```

- Ove funkcije uspisuju znakovni niz tekst kao jedan red teksta u datoteku dat (fputs) odnosno ispisuju preko glavnog izlaza (puts).

Rad sa tekstualnim datotekama - pisanje stringa

- Vrednost obe funkcije je ne-negativan broj, odnosno simbolička konstanta EOF u slučaju otkrivanja greške u toku ispisivanja.
- Primer kojim se ispisuje jedna rečenica u datoteku podaci kao zaseban red teksta:

```
greska = fputs ("Dobar dan!", podaci) == EOF;
```

Prenos podataka sa konverzijom

- Funkcije kojima se vrši ulazna ili izlazna konverzija za sva moguća izvorišta i odredišta podataka.
- Bez obzira na izvorište ili odredište podataka, važe sva pravila i tok konverzije ranije spomenuti.

```
int fprintf(FILE*dat, const char *format, argl,
      arg2, ...);
int printf(constchar *format,argl,arg2,...) ;
int sprintf(char*niz, const char *format, argl,
      arg2, ...) ;
```

Prenos podataka sa konverzijom

- Ove funkcije vrše izlaznu konverziju podataka arg1, arg2, ... uz primenu konverzija koje su zadate znakovnim nizom format.
- Pojedini argumenti mogu da budu nuemrički izrazi, odnosno adresni izrazi za znakovne nizove.
- Odredište rezultujućeg niza znakova je datoteka dat (fprintf), glavni izlaz (printf) ili znakovni niz niz (sprintf).
- Argument niz mora da je dovoljno dugačak za prihvatanje svih znakova dobijenih u toku konverzije i za završni znak \0.
- Ovaj poslednji slučaj naziva se internom konverzijom, jer je i rezultat konverzij u operativnoj memoriji.
- Funkcija **sprintf** je korisna za konverziju binarnih oblika numeričkih podataka u nizove cifara koji mogu dalje da se obrađuju kao tekst u raznim programima za obradu teksta.

Prenos podataka sa konverzijom

- Vrednost sve tri funkcije je broj znakova u rezultujućem nizu znakova, odnosno negativna vrednost u slučaju otkrivanja greške u toku konverzije.
- Primer kojim se piše jedna rečenica uz konverziju numeričkog podatka u datoteku podaci:

```
greska = fprintf (podaci, "Broj kuglica = %d\n", kugl) < 0;
```

Prenos podataka sa konverzijom

```
int fscanf ( FILE *dat, const char *format, arg1, arg2, ...) ;
int scanf ( const char *format, arg1, arg2, ...) ;
int sscanf (const char *niz,  const char *format, arg1, arg2,
        ...) ;
```

- Ove funkcije vrše ulaznu konverziju podataka arg1, arg2, ... uz primenu konverzija koje su zadate znakovnim nizom format.
- Pojedini argumenti treba da budu adrese numeričkih podataka ili znakovnih nizova.
- Izvorište niza znakova koji se konvertuje je datoteka dat (fscanf), glavni ulaz (scanf) ili znakovni niz niz (sscanf).
- Ovaj poslednji slučaj naziva se internom konverzijom, jer je i početni niz koji se konvertuje u operativnoj memoriji.
- Funkcija sscanf je korisna, u programima za obradu teksta, za izdvajanje numeričkih podataka iz teksta i njihovo pretvaranje u binarni oblik za potrebe raznih izračunavanja.

- Vrednost sve tri funkcije je broj konvertovanih podataka (ne znakova!), odnosno simbolička konstanta EOF u slučaju nailaska na kraj datoteke ili otkrivanja greške pre konverzije prvog podatka.
- Primer kojim se čita jedan celobrojni podatak uz konverziju iz datoteke podaci:

```
greska = fscanf (podaci, "%d", Skugl) == EOF;
```

```c
// Primer rada sa tekstualnom datotekom – upis
#include <stdio.h>
#include <stdlib.h>
typedef char Tizraz30[31]; // 30 karaktera i ‘\0’
int main()
{
    FILE *IZLdat;
    Tizraz30 ime, prezime, NazivIZLdat;
    printf("Unesite Vase ime: ");
    scanf("%s", ime);
    __fpurge(stdin);
    printf("\nUnesite vase prezime: ");
    scanf("%s", prezime);
    __fpurge(stdin);
    printf("Unesite naziv datoteke u koju zelite da zapisete podatke: ");
    scanf("%s", NazivIZLdat);
    // otvaranje datoteke sa proverom prava na pisanje(w)
    if((IZLdat=fopen(NazivIZLdat,"w")) == NULL) {
        printf("Greska prilikom otvaranja datoteke %s\n",NazivIZLdat);
        // izlaz iz programa u slucaju zabrane pisanja u datoteku
        exit(EXIT_FAILURE);
    }
    fprintf(IZLdat,"%s\n%s",ime, prezime); // upis u izlaznu datoteku
    fclose(IZLdat); // zatvaranje datoteke
    printf("Podaci studenta su upisani u datoteci: %s\n", NazivIZLdat);
    return 0;
}
```

Rad sa binarnim datotekama - čitanje

```
int fread (void *niz, int vel, int br, FILE *dat) ;
```

- Ova funkcija čita iz datoteke dat najviše **br** podataka veličine **vel bajtova** u memoriju počev od adrese **niz**. Čitanje podataka počinje na poziciji gde je prethodni pristup datoteci završen.
- Po završetku čitanja zaustavlja se neposredno iza poslednjeg pročitanog bajta.

Rad sa binarnim datotekama - čitanje

- Dužnost je programera da vodi računa o logičkoj strukturi datoteke, tj. o veličini pojedinih podataka, njihovom broju i redosledu.
- U slučaju neusaglašenosti čitanja prema ranijim upisivanjima, dobiće se besmisleni podaci bez ikakve opomene.

Rad sa binarnim datotekama - čitanje

- Vrednost funkcije je stvarni broj pročitanih podataka. Za ispitivanje uspeha ili ne- uspeha neophodno je da se koriste ftmkcije feof i ferror. Jedan karakterističan siučaj kada je vrednost funkcije manja od br je kada od trenutne pozicije u datoteci do kraja datoteke nema dovoljan broj podataka.
- Primer kojim se iz datoteke podaci vrši čitanje niza od najviše n_max celobrojnih podatka:

```
int n = fread (vektor, sizeof (int), n_max, podaci);
```

Rad sa binarnim datotekama - pisanje

```
fwrite (const void *niz, int vel, int br, FILE *dat);
```

- Ova funkcija piše **br** podataka veličine **vel** bajtova iz memorije počev od adrese nizu datoteku dat.
- Pisanje podataka počinje na poziciji gde je prethodni pristup datoteci završen.
- Ako trenutna pozicija nije na kraju datoteke, nove vrednosti se prepisuju preko zatečenih vrednosti u datoteci.

Rad sa binarnim datotekama - pisanje

- Ako do kraja datoteke nema dovoljno mesta za smeštanje svih podataka, pisanje se nastavlja iza kraja datoteke.
- Time se povećava veličina datoteke.
- Po završetku pisanja zaustavlja se neposredno iza poslednjeg upisanog bajta.

Rad sa binarnim datotekama - pisanje

- Dužnost je programera da vodi računa o logičkoj strukturi datoteke, tj. o veličini pojedinih podataka, njihovom broju i redosledu.
- U slučaju neusaglašenosti prilikom prepisivanja preko starog sadržaja prema ranijim upisivanjima, oštetiće se logička struktura datoteke bez ikakve opomene.

Rad sa binarnim datotekama - pisanje

- Vrednost funkcije je broj prenetih podataka.
- Ta vrednost je u slučaju greške manja od br.
- Primer kojim se u datoteku podaci piše niz od n celobrojnih podatka:

```
greska = fwrite (vektor, sizeof (int), n, podaci) < n;
```

Pozicioniranje unutar datoteke (direktan pristup)

```
int fseek (FILE *dat, long pomeraj, int reper) ;
```

- Ova funkcija vrši pozicioniranje na mesto u datoteci dat čija je udaljenost pomeraj bajtova od označene reperne tačke.
- Moguće reperne tačke obeležavaju se simboličkim konstantama **SEEK_SET** (početak datoteke), **SEEK_CUR** (trenutna pozicija u datoteci) ili **SEEK_END** (kraj datoteke).
- Sledeće čitanje ili pisanje vršiće se počevši od ovako odabrane pozicije u datoteci.