

Programiranje (C)

13. predavanje

Saša Singer

`singer@math.hr`

`web.math.hr/~singer`

`www.math.hr/~singer`

PMF – Matematički odjel, Zagreb

Sadržaj predavanja

- Datoteke (zadnji dio):
 - Otvaranje i zatvaranje datoteke (ponavljanje).
 - Standardne datoteke — `stdin`, `stdout`, `stderr`.
 - Funkcije za čitanje i pisanje — znak po znak, liniju po liniju, formatirani ulaz i izlaz.
 - Binarni ulaz i izlaz.
 - Direktan pristup podacima.

Datoteke (nastavak)

Otvaranje i zatvaranje datoteke (ponavljanje)

- Komunikacija s datotekama vrši se preko **spremnika** (engl. buffer) u koji se privremeno pohranjuju informacije koje se šalju u datoteku.
- Svrha je spremnika **smanjiti komunikaciju** s vanjskom memorijom (diskom) i povećati efikasnost ulazno–izlaznih funkcija.
- Spremnik i ostali podaci potrebni za otvaranje **datoteke** deklarirani su u **strukturi FILE**.
- Prvi korak u **otvaranju** datoteke je deklaracija **pokazivača** na **FILE**.

Primjer:

```
FILE *ptvar;
```

Otvaranje i zatvaranje datoteke (ponavljanje)

Prije prve operacije pisanja ili čitanja datoteka mora biti otvorena korištenjem funkcije `fopen`.

```
FILE *fopen(const char *ime, const char *tip);
```

gdje je:

- `ime` pravo ime datoteke koja se otvara,
- a `tip` string koji kaže kako treba otvoriti tu datoteku ("`r`", "`w`", "`a`", ...).

Funkcija `fopen` vraća:

- pokazivač na strukturu `FILE` povezanu s datotekom, ako je datoteka uspješno otvorena,
- `NULL`, ako datoteka nije mogla biti otvorena (greška).

Otvaranje i zatvaranje datoteke (ponavljanje)

Na kraju programa datoteka treba biti **zatvorena** funkcijom **fclose** koja uzima kao argument **pokazivač na FILE**.

```
int fclose(FILE *fp);
```

Funkcija **fclose** vraća:

- nulu, ako je datoteka **uspješno** zatvorena,
- EOF u slučaju **greske**.

Primjer:

```
fclose(ptvar);
```

Otvaranje i zatvaranje datoteke (ponavljanje)

Primjer. Otvaranje (za pisanje) i zatvaranje datoteke **primjer.dat** izgledalo bi ovako:

```
#include <stdio.h>
...
FILE *fpt;
if ((fpt = fopen("primjer.dat", "w")) == NULL) {
    printf("\nGRESKA pri otvaranju datoteke.\n");
    exit(-1);
}
/* Rad s datotekom (pisanje u nju) */
...
fclose(fpt);
```

Automatski otvorene datoteke

- C svakom programu automatski otvara tri datoteke. To su standardni ulaz, standardni izlaz i standardni izlaz za greške.
- Standardni ulaz je tipkovnica računala, standarni izlaz i izlaz za greške ekran računala.
- Neki operacijski sustavi (Unix, DOS,...) mogu standardne, automatski otvorene datoteke preusmjeriti pri pozivu programa, tako da učitavanje/ispis vrše iz datoteke (“pipes”).
- U datoteci `<stdio.h>` definirani su konstantni pokazivači na FILE strukturu povezanu sa standardnim ulazom, izlazom i izlazom za greške. Ti pokazivači imaju imena `stdin` (std. ulaz), `stdout` (std. izlaz) i `stderr` (std. izlaz za greške).

Funkcije za čitanje i pisanje

Funkcije:

standardni ulaz/izlaz	koje rade s datotekom
getchar	fgetc, getc
putchar	fputc, putc
gets	fgets
puts	fputs
printf	fprintf
scanf	fscanf

Sve funkcije u desnom stupcu kao argument imaju pokazivač na FILE.

To je zadnji argument u prva četiri reda, a prvi argument za zadnje dvije funkcije.

Čitanje i pisanje znak po znak

Funkcije:

```
int fgetc(FILE *fp);  
int getc(FILE *fp);
```

vraćaju sljedeći znak iz datoteke na koju pokazuje **fp**.

- Razlika između **getc** i **fgetc** je da **getc** može biti implementirana kao makro naredba dok **fgetc** ne smije.
- U slučaju greške ili kraja datoteke vraća se EOF.
- Funkcija **getchar** implementira se kao **getc(stdin)**.

Vraćeni znak je **unsigned char** (pretvoren u **int**).

Čitanje i pisanje znak po znak (nastavak)

Primjer. Program koji broji znakove u datoteci koja je navedena kao argument komandne linije.

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

int main(int argc, char *argv[]) {
    int ch, count = 0;
    FILE *fpt;

    if (argc == 1) { /* nema imena datoteke! */
        printf("\nUporaba: %s ime_datoteke\n",
               argv[0]);
        exit(-1);
    }
```

Čitanje i pisanje znak po znak (nastavak)

```
else
    if ((fpt = fopen(argv[1], "r")) == NULL) {
        printf("Ne mogu otvoriti"
               "datoteku %s\n", argv[1]);
        exit(-1);
    }
    while ((ch = fgetc(fpt)) != EOF) count++;
    fclose(fpt);
    printf("\nBroj znakova = %d\n", count);
    return 0;
}
```

Čitanje i pisanje znak po znak (nastavak)

Funkcije:

```
int fputc(int c, FILE *fp);  
int putc(int c, FILE *fp);
```

upisuju znak (pretvoren u **unsigned char**) u datoteku.

- Razlika između **putc** i **fputc** je da **putc** može biti implementirana kao makro naredba dok **fputc** ne smije.
- Vraća se **upisani znak**.
- U slučaju **greške** vraća se **EOF**.
- Funkcija **putchar(c)** implementira se kao **putc(c, stdout)**.

Čitanje i pisanje znak po znak (nastavak)

Primjer. Funkcija koja kopira jednu datoteku u drugu može se napisati ovako:

```
void cpy(FILE *fpulaz, FILE *fpizlaz) {  
    int c;  
    while ((c = getc(fpulaz)) != EOF)  
        putc(c, fpizlaz);  
}
```

Čitanje i pisanje liniju po liniju

Funkcija koja **učitava** podatke iz datoteke **liniju po liniju** je:

```
char *fgets(char *buf, int n, FILE *fp);
```

- **buf** je pokazivač na dio memorije (engl. buffer) u koji će ulazna linija biti spremljena.
- **n** je veličina memorije (stringa) na koju prvi argument pokazuje. Preciznije, funkcija čita najviše **n-1** znak, da ima mesta za nul-znak na kraju.
- **fp** je pokazivač na datoteku iz koje se učitava.

Čitanje i pisanje liniju po liniju (nastavak)

- Funkcija će pročitati liniju **uključujući** i znak za prijelaz u novi red i na kraj **dodati** nul–znak '**\0**'. Pritom će biti učitano najviše **n – 1** znakova, uključivši '**\n**'.
- Ako je ulazna linija **dulja** od toga, ostatak će biti pročitan pri sljedećem pozivu funkcije **fgets**.
- Funkcija vraća **buf** ako je sve u redu ili **NULL** ako se došlo do kraja datoteke ili se pojavila greška.

Čitanje i pisanje liniju po liniju (nastavak)

Funkcija:

```
char *gets(char *buf);
```

uvijek čita sa standardnog ulaza.

- `gets` ne uzima veličinu buffera kao argument i stoga se može dogoditi da je ulazna linija **dulja** od za nju rezervirane memorije.
- Stoga je bolje koristiti `fgetf(buf, n, stdin)` umjesto `gets(buf)`.
- Pritom treba uzeti u obzir da `fgets` učitava i znak '`\n`' (bez zamjene), dok `gets` učita '`\n`' i **zamjenjuje** ga znakom '`\0`'.

Čitanje i pisanje liniju po liniju (nastavak)

Funkcija za **ispis** podataka u datoteku, **liniju po liniju** je:

```
int fputs(const char *str, FILE *fp);
```

- Funkcija vraća nenegativnu vrijednost ako je ispis uspio ili **EOF** ako je došlo do greške.
- **fputs** ispisuje znakovni niz na koji pokazuje **str** u datoteku na koju pokazuje **fp**. Zadnji nul–znak neće biti isписан.
- Ako želimo prijelaz u novi red, onda string mora sadržavati znak '**\n**' (ne piše se automatski).

Čitanje i pisanje liniju po liniju (nastavak)

Funkcija:

```
int puts(const char *str);
```

ispisuje znakovni niz na koji pokazuje **str** na standardni izlaz.

Na kraj niza ona **dodaje** znak za prijelaz u novi red što je razlikuje od **fputs(str, stdout)**.

Formatirani upis/ispis

Za formatirani ispis/upis u datoteku koristimo funkcije:

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

Jedina razlika obzirom na **printf** i **scanf**: prvi argument im je **pokazivač na datoteku** u koju se vrši upis/iz koje se čita.

- **fscanf** vraća broj učitanih objekata ili **EOF** ako je došlo do greške ili kraja datoteke.
- **fprintf** vraća broj upisanih znakova ili negativan broj u slučaju greške.
- **printf(...)** je ekvivalentno s **fprintf(stdout, ...)**, a **scanf(...)** je ekvivalentno s **fscanf(stdin, ...)**.

Formatirani upis/ispis za stringove

Funkcije `sscanf`, `sprintf` čitaju/pišu u `string`, umjesto u datoteku.

```
int sprintf(char *s, const char *format, ...);  
int sscanf(char *s, const char *format, ...);
```

- `sprintf` dodaje nul–znak '\0' na kraj stringa, ali ga ne broji kad vraća broj napisanih znakova.

Prepoznavanje greške

Funkcije za upis podataka vraćaju **EOF** u dva slučaja: ako je došlo do **greške** i ako se došlo na **kraj datoteke**.

Funkcije:

```
int ferror(FILE *fp);  
int feof(FILE *fp);
```

služe za razlikovanje pojedinih situacija.

- **ferror** vraća broj različit od nule (istina) ako je došlo do **greške**, a nulu (laž) ako nije.
- **feof** vraća broj različit od nule (istina) ako smo došli do **kraja datoteke**, a nulu (laž) u suprotnom.

Prepoznavanje greške (nastavak)

Primjer. Sljedeći program kopira standardni **ulaz** na standardni **izlaz** (liniju po liniju) i detektira ulaznu grešku korištenjem funkcije **ferror**.

```
#include <stdio.h>
const int MAXLINE = 128;

int main(void) {
    char buf[MAXLINE];

    while (fgets(buf, MAXLINE, stdin) != NULL)
        if (fputs(buf, stdout) == EOF) {
            fprintf(stderr,
                    "\nIzlazna greska.\n");
            exit(1); }
```

Prepoznavanje greške (nastavak)

```
if (ferror(stdin)) {  
    fprintf(stderr, "\nUlagana greska.\n");  
    exit(2);  
}  
return 0;  
}
```

Binarni upis/ispis

Funkcije za **binarno** čitanje i pisanje su:

```
size_t fread(void *ptr, size_t size,
            size_t nobj, FILE *fp);
size_t fwrite(const void *ptr, size_t size,
             size_t nobj, FILE *fp);
```

One **ne vrše** konverziju iz **binarnog** zapisa u **znakovni** zapis.

Pritom je:

- **fp** pokazivač na datoteku u koju se piše/iz koje se čita,
- **ptr** pokazivač na varijablu (polje) u koju **fread** upisuje, odnosno iz koje **fwrite** čita,
- **nobj** broj objekata koje treba ispisati/učitati,
- **size** veličina pojedinog objekta.

Binarni upis/ispis (nastavak)

- `fread` čita iz datoteke na koju pokazuje `fp` niz od `nobj` objekata dimenzije `size` i smješta ih u polje na koje pokazuje `ptr`.
- Vrijednost koju funkcija vraća je **broj učitanih objekata**, koji može biti manji od `nobj` ako je došlo do greške ili kraja datoteke.
- `fwrite` upisuje u datoteku na koju pokazuje `fp` niz od `nobj` objekata dimenzije `size` iz polja na koje pokazuje `ptr`.
- Vrijednost koju funkcija vraća je **broj upisanih objekata**. Taj broj može biti manji od `nobj` ako je došlo do greške.

Binarni upis/ispis (nastavak)

Primjer. Zapis (jedne) strukture u datoteku.

```
FILE *fpt;
struct account{
    short count;
    long total;
    char name[64];
} d11;

fpt = fopen(...);
...
if (fwrite(&d11, sizeof(d11), 1, fpt) != 1) {
    fprintf(stderr, "Greska pri upisu\n");
    exit(1); }
```

Binarni upis/ispis (nastavak)

- **Prednost** binarnog ulaza/izlaza: brzina i (mala) veličina zapisa.
- **Nedostatak** binarnog ulaza/izlaza: zavisnost o arhitekturi računala i prevoditelju.

Direktan pristup podacima

Dosad uvedene funkcije podacima pristupaju **sekvencijalno**. Sljedeće funkcije omogućavaju **direktan** pristup podacima:

```
int fseek(FILE *fp, long offset, int pos);  
long ftell(FILE *fp);
```

- Funkcija **ftell** uzima kao argument pokazivač na **otvorenu** datoteku i vraća **trenutni položaj** unutar datoteke. U slučaju greške vraća **-1L**.
- Funkcija **fseek** uzima pokazivač na **otvorenu** datoteku, te dva parametra koji specificiraju **položaj** u datoteci. Treći parametar **pos** indicira od kog mesta se mjeri **offset** (odmak). Funkcija **postavlja trenutnu poziciju** na **zadanu lokaciju**.

Direktan pristup podacima (nastavak)

U datoteci `stdio.h` definirane su tri simboličke konstante: `SEEK_SET`, `SEEK_CUR` i `SEEK_END`, koje se mogu koristiti kao treći argument funkcije `fseek`, a imaju sljedeće značenje:

pos	Pozicija u datoteci: offset znakova od
<code>SEEK_SET</code>	početka datoteke
<code>SEEK_CUR</code>	trenutne pozicije u datoteci
<code>SEEK_END</code>	kraja datoteke

Direktan pristup podacima (nastavak)

Primjer:

Poziv <code>fseek(argumenti)</code>	Značenje
<code>fp, 0L, SEEK_SET</code>	idi na početak datoteke
<code>fp, 0L, SEEK_END</code>	idi na kraj datoteke
<code>fp, 0L, SEEK_CUR</code>	ostani na trenutnoj poziciji
<code>fp, ftell_pos, SEEK_SET</code>	idi na poziciju je dao prethodni poziv <code>ftell</code>

Standard postavlja neka ograničenja na `fseek`. Za binarne
datoteke `SEEK_END` nije dobro definiran, dok su za tekstualne
datoteke jedino pozivi iz prethodne tablice dobro definirani.

Direktan pristup podacima (nastavak)

Primjer. Funkcija koja u datoteci traži određeni **account** i povećava njegovo polje **total** za 100, ako je ono manje od 5000.

```
void Povecaj(const char *file, int n) {
    FILE *fp;
    struct account tmp;
    long pos;
    int size = sizeof(struct account);

    if ((fp = fopen(file,"r+b")) == NULL) {
        fprintf(stderr, "Nije moguce otvoriti"
                " datoteku %s.\n", file);
        exit(-1);
    }
```

Direktni pristup podacima (nastavak)

```
pos = n * size; /* Ispred (n+1)-og zapisa. */
fseek(fp, pos, SEEK_SET);
if (fread(&tmp, size, 1, fp) != 1) {
    fprintf(stderr,"Greska pri citanju.\n");
    exit(-1); }
if (tmp.total < 5000L) {
    tmp.total += 100;
    fseek(fp, -size, SEEK_CUR);
    if (fwrite(&tmp, size, 1, fp) != 1) {
        fprintf(stderr,"Greska pri upisu.\n");
        exit(1); }
}
fclose(fp);
return; }
```

Direktan pristup podacima (nastavak)

Komentari na program:

- Funkcija mijenja **sadržaj** **n + 1**-og zapisa (brojanje zapisa počinje od nule).
- Prvo se otvara datoteka za čitanje i upisivanje (**r+b**), a zatim se pokazivač pozicionira na početak **n+1**-og zapisa (**fseek(fp, pos, SEEK_SET);**).
- Zapis se pročita i ako je **total** manji od 5000 povećava se za 100.
- Nakon što je zapis promijenjen, treba ga upisati nazad u datoteku. U tu svrhu se prvo vraćamo na početak **n+1**-og zapisa (**fseek(fp, -size, SEEK_CUR);**) i onda vršimo upis.