PROGRAMIRANJE

PROGRAMSKI JEZIK C

Datoteke

Tokovi podataka

- C ulazno izlazna biblioteka stdio je najveća i najznačajnija standardna biblioteka
- Funkcije stdio biblioteke komunciraju sa ulazno izlaznim uređajima na uniforman način
  - Koristi se mehanizam toka podataka (stream)
    - Tok podataka obično reprezentuje datoteku smeštenu na različitim medijima, ali se može pridružiti i uređaju na kome se ne smešta datoteka (tastatura, ekran, printer, port, …)
  - Bajt orijentisane funkcije
- Svakom toku podataka se pridružuje određena struktura podataka sa imenom FILE definisana u stdio.h
  - Pristup toku podataka se vrši preko odgovarajućeg identifikatora koji je tipa `FILE *` i kreira se otvaranjem toka podataka

Standardni tokovi podataka

- Tri standardna toka podataka su deklarisana u stdio.h

FILE *	Tok podataka	Podrazumevani uređaj
stdin	standardni ulaz	tastatura
stdout	standardni izlaz	ekran
stderr	standardna prijava greške	ekran

  - Navedeni tokovi podataka su direktno dostupni
  - Funkcije printf, scanf, putchar, getchar, puts, gets
    - prihvataju ulaz sa standardnog ulaza stdin
    - vrše ispis na standardni izlaz stdout

Redirekcija ulaza, izlaza

- Redirekcija ulaza
  - Program će prihvatati ulaz iz datoteke umesto sa tastature, ako se u komandnoj liniji navede ime datoteke kome prethodi karakter <

```
demo < in.dat
```

- Redirekcija izlaza
  - Slično ispis će se vršiti u datoteku umesto na ekran, ako se u komandnoj liniji navede ime datoteke kome prethodi karakter >

```
demo > out.dat
```

- Pri tome se mogu kombinovati redirekcije

```
demo < in.dat > out.dat
```

Binarne i tekstualne datoteke

- Stanadrdna biblioteka podržava dva tipa datoteka
  - Tekstualne
  - Binarne
- U tekstualnim datotekama
  - bajtovi reprezentuju karaktere
  - sadržaj je čitljiv
- U binarnim datotekama
  - bajtovi ne moraju reprezentovati karaktere
  - grupe bajtova mogu reprezentovati vrednosti drugih tipova podataka (int, float, … vrednosti)
  - sadržaj datoteke je kopija sadržaja memorije računara (nema konverzije)

Binarne i tekstualne datoteke

- Zapis vrednosti 32767
  - tekstualna datoteku (ASCII karakter set)

00110011	00110010	00110111	00110110	00110111
'3'	'2'	'7'	'6'	'7'

  - Binarna datoteka

01111111	11111111

- Za razliku od binarnih tekstualne datoteke su
  - Organizovane po linijama
    - Linija se zaršava sa jednim ili dva karaktera koji su markeri kraja linije (CR-LF na Windows, odnosno LF na Unix platformi)
  - Mogu sadržati marker kraja datoteke (Ctrl-Z na Windows odnosno Ctrl-D na Unix platformi)

Otvaranje datoteke

- Pre korišćenja datoteka se otvara (zahteva se od operativnog sistema dozvola pristupa)

```
FILE *fopen(const char datoteka[], const char mod[]);
```

- daoteka: ime datoteke
  - ime datoteke može uključivati i njenu lokaciju (putanju)
- mod definiše vrstu aktivnosti nad datotekom
  - r: Otvara se datoteka za čitanje (read)
    - Datoteka mora da postoji
      - Ako datoteka ne postoji fopen() vraća NULL
  - w: Otvara se datoteka za pisanje (write)
    - Datoteka ne mora da potoji
      - Ako ne postoji datoteka sa zadanim imenom kreira se nova datoteka
      - Ako postoji datoteka sa zadanim imenom njen sadržaj se briše
    - Podaci se upisuju od početka datoteke
  - a: Otvara se datoteka za dodavanje sadržaja (append)
    - Datoteka ne mora da postoji
      - Ako ne postoji datoteka sa zadanim imenom kreira se nova datoteka
    - Podaci se dodaju na kraj datoteke

Otvaranje datoteke

- r+: Otvara se datoteka za čitanje i pisanje
  - Upis se vrši od početka datoteke
- w+: Otvara se datoteka za čitanje i pisanje
  - Ako postoji datoteka sa zadanim imenom njen sadržaj se briše
- a+: Otvara se datoteka za čitanje ododavanje sadržaja
  - Ako ne postoji datoteka sa zadanim imenom kreira se nova datoteka
- b: Otvara se binarna datoteka
  - rb, wb, ab, r+b, w+b, rb+, wb+, ab+

```
FILE *fp;
fp=fopen("c:\\project\\demo.txt", "r");
```

⇔

```
FILE *fp;
fp=fopen("c:/project/demo.txt", "r");
```

```
if( fp == NULL) {
   printf("Greska pri otvaranju datoteke");
   exit(EXIT_FAILURE);
}
```

← Test uspešnosti otvaranja

Zatvaranje datoteke

- `int fclose(FILE* stream);` Nakon korišćenja datoteka se zatvara

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

#define FILE_NAME "example.dat"

int main(void) {
    FILE * fp=fopen(FILE_NAME, "r");
    …
    if( fp == NULL) {
        printf("Greska pri otvaranju datoteke: %s\n", FILE_NAME);
        exit(EXIT_FAILURE);
    }
    …
    fclose(fp);
    return EXIT_SUCCESS; /* ili return 0; */
}
```

Baferovanje (buffering)

- Prenos podataka sa, odnosno na dati tok podataka (periferni uređaj) je relativno spora operacija
  - Performanse se značajno popravljaju korišćenjem tehnike baferovanja
    - Podaci koji se upisuju na tok podataka se smeštaju u poseban memorijski prostor (bafer)
      - Kada se bafer napuni ili se tok podataka zatvori, sadržaj bafera se prebacuje na dati tok podataka u celosti (flush)
    - Slično se i ulazni tok podataka baferuje
      - Bafer prihvata podatke sa ulaznog toka, a zatim se
      - Podaci učitavaju iz bafera

`fflush(fp);` ← Prazni bafer pridružen toku podataka fp

Formatirani Ulaz/Izlaz

```
int fscanf(FILE *stream, const char fmt[], ...);
```

```
int fprintf(FILE *stream, const char fmt[], ...);
```

```
scanf(fmt, ...);   <-->   fscanf(stdin, fmt, ...);
```

```
printf(fmt, ...);  <-->   fprintf(stdout, fmt, ...);
```

- fprintf (printf) vraća broj odštampanih karaktera
- fscanf (scanf) vraća broj uspešno učitanih i pridruženih vrednosti
  - stream je prethodno otvoreni tok podataka
  - fmt je upravljački niz znakova
    - Najvećim delom opisan u prezentaciji PR01

Indikator kraja datoteke i indikator greške

- Pri radu sa tokovima podataka moguće su sledeće situacije
  - kraj ulaznog toka podataka
  - greška pri čitanju sa ulaznog toka podataka
  - greška pri upisu na izlazni tok podataka
- Svakom toku podataka su pridružena dva inidkatora (brišu se otvaranjem toka podataka)
  - Indikator kraja datoteke (end-of-file indicator)
  - Indikator greške (error indicator)

Indikator kraja datoteke i indikator greške

- Jednom postavljen indikator ostaje u tom stanju sve dok se eksplicitno ne obriše

```
clearerr(fp);
```
← briše oba indikatora pridružena toku podataka fp

- Testiranje indikatora

```
feof(fp);
```

```
ferror(fp);
```

```
while ( !feof(fp) ) {
    /* obrada */
}
```

```
if(fscanf(fp, "%d", &i) != 1) {
    if(ferror(fp)) {
        fclose(fp);
        return -1; /* greška čitanja*/
    }
    if(feof(fp)) {
        fclose(fp);
        return -2; /* kraj datoteke */
    }
}
```

Karakter Ulaz/Izlaz

```
int fgetc(FILE *stream)
```

```
int fputc(int c, FILE *stream)
```

getchar()	⟷	fgetc(stdin)
putchar(c)	⟷	fputc(c, stdout)

- `int ungetc(int c , FILE *stream);` vraća karakter na ulazni tok

```
int c, n = 0;

while((c = getc(fp)) >= '0' && c <= '9')
        n = 10 * n + (c - '0');
ungetc(c, fp);  /* nije cifra – vrati znak na ulazni tok */
```

Linijski Ulaz

`fgets(char str[], int n, FILE *stream)`

- fgets: Učitava se linija sa datog toka podataka stream sve do pojave new-line karaktera (koji se takođe smešta u prihvatni niz str)
  - Broj učitanih karaktera je najviše n (bezbednija od gets funkcije)
- gets: Učitava se linija sa standardnog ulaza sve do pojave new-line karaktera (koji se odbacuje)
  - Ne može ograničiti broj karaktera koji će se učitati
- Umesto `gets(str)` bezbednije je `fgets(str, sizeof(str), stdin)`

Linijski Izlaz

```
int fputs(char str[], FILE *stream)
```

- fputs: Upisuje niz karaktera str na izlazni tok stream
  - new-line karakter se ispisuje samo ako je elemenat niza karaktera str
- puts: Nakon što se izvrši ipis niz karaktera na standardni izlaz automatski ispisuje new-line karakter

fcopy.c

```
#include <stdio.h>

void fcopy(FILE *, FILE *);

int main(void) {
    FILE *infp, *outfp;
    char inf[80], outf[80];

    fgets(inf, sizeof(inf), stdin);
    fgets(outf, sizeof(outff), stdin);
    infp=fopen(inf, “rb”); // izostavljen je test
    outfp=fopen(inf, “wb”); //uspesnosti otvaranja

    fcopy(infp, outfp)

    fclose(ifp); fclose(outfp);
    return 0;
}
```

```
void fcopy(FILE *ifp, FILE *ofp) {
    int c;
    while((c=fgetc(ifp))!=EOF) {
        fputc(c, ofp);
    }
}
```

Blokovski Ulaz/Izlaz

- fread, fwrite funkcije se koriste za ulaz i izlaz velikih blokova podataka u jednom koraku
  - Obično se koriste sa binarnim tokovima podataka
  - fread: učitava se niz sa ulaznog toka

```
n=fread(a, sizeof(a[0]), sizeof(a)/sizeof(a[0]), fp);
```

    - Argumenti su redom niz koji se učitava, veličina memorijskog prostora u bajtovima za smeštanje pojedinog elementa niza, zatim broj elemenata niza koji će se učitati i kao poslednji ulazni toka podataka
    - Povratna vrednost fread funkcije je stvarni broj elemenata niza koji su učitani sa ulaznog toka

Blokovski Ulaz/Izlaz

- fwrite: niz se kopira na izlazni tok podataka

```
n=fwrite(a, sizeof(a[0]), sizeof(a)/sizeof(a[0]), fp);
```

  - Argumenti su redom niz koji se učitava, veličina memorijskog prostora u bajtovima za smeštanje pojedinog elementa niza, zatim broj elemenata niza koji će se učitati i kao poslednji ulazni toka podataka
  - Povratna vrednost fwrite funkcije je stvarni broj elemenata niza koji su upisani na izlazni tok

Pozicioniranje u okviru toka podataka

- Za svaki tok podataka beleži se tekuća pozicija
  - Otvaranjem toka podataka tekuća pozicija je sam početak toka
    - Ako je tok podataka otvoren u modu za dodavanje sadržaja ("a"), tekuća pozicija može biti i kraj toka podataka (zavisi od konkretne implementacije)
    - Dalje učitavanjem/upisivanjem tekuća pozicija sekvencijalno napreduje
    - Sekvencijalan pristup nije uvek pogodan
- Funkcije za pozicioniranje najčešće se koriste sa binarnim tokovima podataka

Pozicioniranje u okviru toka podataka

- fseek: relativno pozicioniranje u odnosu na tri fiksne pozicije (koja se zadaje kao treći argument)
  - SEEK_SET: početak toka podataka
  - SEEK_CUR: tekuća pozicija
  - SEEK_END: kraj toka podataka

```
fseek(fp, 0L, SEEK_SET); /* idi na početak toka podataka */
```

```
fseek(fp, 0L, SEEK_END); /* idi na kraj toka podataka */
```

```
fseek(fp, -10L, SEEK_CUR); /* vrati se 10 bajtova nazad */
```

  - Prvi argument je tok podataka
  - Drugi argument je pomak u bajtovima koji je tipa long int

Pozicioniranje u okviru toka podataka

- ftell: vraća tekuću poziciju u datom toku podataka kao long int

```
long file_pos;
…
file_pos=ftell(fp); /* zabeleži tekuću poziciju */
…
fseek(fp, file_pos, SEEK_SET); /* vrati se na zabeleženu poziciju */
```

- frewind: pozicioniranje na početak datog toka podataka

```
frewind(fp);   ⟷   fseek(fp, 0L, SEEK_END);
```

Čitanje elementa sa indeksom i celobrojnog niza iz binarne datoteke

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

#define MAXNAME 20

int main() {
    char datoteka[MAXNAME+1];
    FILE *indat;
    int n, i, a[1];

    printf("Unesite naziv datoteke: ");
    scanf("%s", datoteka);
    printf("indeks (>=0): ");
    scanf("%d", &i);

    if((indat=fopen(datoteka, "rb"))==NULL) {
        printf("Error: otvaranje datoteke: "
               "%s!!!", datoteka);
        exit(EXIT_FAILURE);
    }
```

```
    fseek(indat, 0L, SEEK_END);
    n=ftell(indat)/sizeof(int);

    if(i>=n) {
        printf("Error: Nepotojeci indeks: "
               "%d>=%d!!!", i, n);
        exit(EXIT_FAILURE);
    }
    fseek(indat, (long)i*sizeof(int), SEEK_SET);
    fread(a, sizeof(int), 1, indat);

    printf("a[%d]=%d\n", i, a[0]);

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