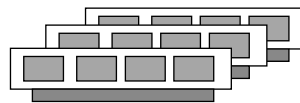


# Datoteke

## Memorija računala

---

- 1) privremena (unutarnja)  
RAM (Random Access Memory)



- 2) stalna (vanjska)  
a) sa slijednim pristupom podacima, npr.

magnetske trake



streamer trake



- b) s direktnim pristupom podacima, npr.

diskete

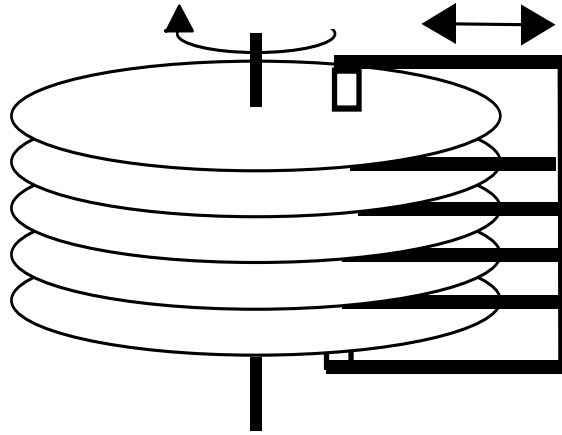


magnetski diskovi



## Shematski prikaz magnetskog diska

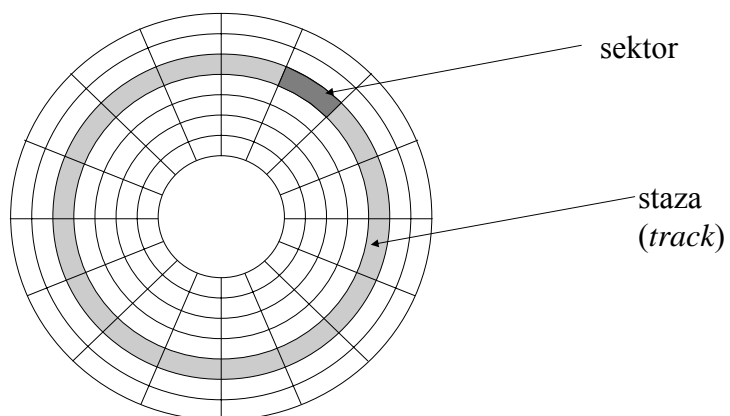
---



3

## Fizička organizacija magnetskog diska

---



4

## Logička organizacija magnetskog diska

---

Direktoriji

(imenici, kazala)

Datoteke

c:\		&..			
0395		atapi_cd.sys	24144	10.28.94	20:14:58
original		cdplay.exe	34494	05.18.94	2:11:00
update		cr520.sys	10871	02.26.95	17:30:10
69!		eject.com	7987	01.17.91	13:23:06
69!.ins		install.exe	18971	09.12.94	1:00:00
access		load.exe	10700	06.28.94	11:16:16
aw		lock.com	7987	01.17.91	13:23:28
bin		mscdex.exe	25377	02.26.95	17:30:10
bo		mtmcdac.sys	17351	04.18.94	1:16:00
brink		playcd.exe	17790	01.16.92	1:30:00
cdrom		readme.txt	6196	11.04.94	10:59:50
corel50		unlock.com	7459	01.17.91	13:23:50

5

## Direktoriji i datoteke

---

- **Datoteka:** imenovani skup podataka na mediju za pohranu (traka, disk, disketa, CD, *memory stick*, ...)
- **Zapis:** skup susjednih podataka unutar datoteke koji se obrađuje kao cjelina
- **Direktorij (imenik, kazalo):** datoteka koja sadrži popis i podatke o karakteristikama drugih datoteka. Direktoriji su organizirani hijerarhijski, u strukturu nalik na stablo
- **Operacijski sustav računala:** program koji povezuje sklopovlje računala s programskom opremom. Između ostalog, vodi evidenciju o fizičkom smještaju direktorija i datoteka na mediju za pohranu

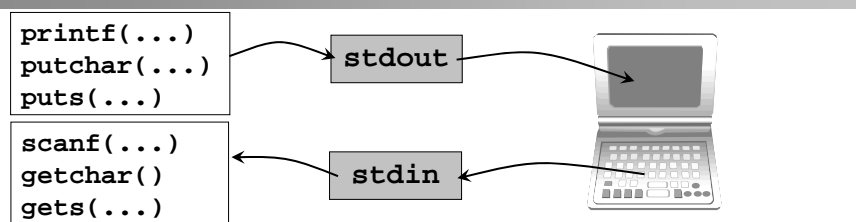
6

## Tok podataka (*stream*)

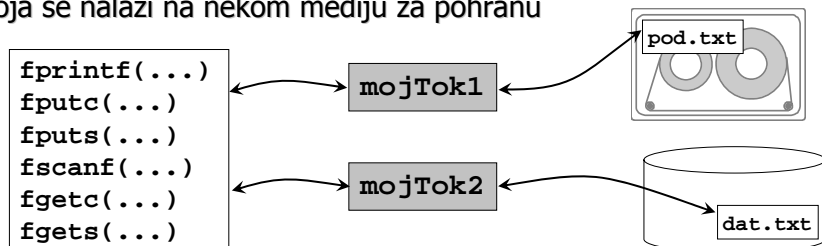
- Tok podataka ili *stream*: bilo koji **izvor** ulaznih podataka i/ili **odredište** izlaznih podataka
- Do sada smo koristili sljedeće tokove podataka:
  - standardni ulaz (najčešće tipkovnica): izvor podataka
    - npr. funkcije `scanf`, `getchar`, `gets` **uvijek** čitaju sa standardnog ulaza
  - standardni izlaz (najčešće zaslon): odredište podataka
    - npr. funkcije `printf`, `putchar`, `puts` **uvijek** ispisuju na standardni izlaz
- standardni ulaz (*stdin*) i standardni izlaz (*stdout*) se automatski otvaraju pri pokretanju programa
  - globalne konstante definirane u `<stdio.h>`

7

## Tok podataka (*stream*)



- moguće je otvoriti "vlastite" tokove podataka - npr. tok podataka pomoću kojeg se čitaju i/ili pišu podaci u datoteku koja se nalazi na nekom mediju za pohranu



8

**Primjer:** napisati program kojim će se iz datoteke `tekst.txt` čitati jedan po jedan znak. Svaki pročitani znak ispisati na ekran (*stdout*)

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

int main () {
    int c;
    FILE *tokPod;
    tokPod = fopen("tekst.txt", "r");

    if (tokPod == NULL) {
        printf("ne mogu otvoriti tekst.txt\n");
        exit(-99);
    }

    while ((c = fgetc(tokPod)) != EOF)
        putchar(c); /* ili fputc(c, stdout); */

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

datoteka tekst.txt  
Funkcije `scanf`, `getchar` i `gets` uvijek citaju iz toka podataka koji se naziva `stdin`.  
A funkcije `printf`, `putchar` i `puts`?

9

## fopen

## <stdio.h>

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

- omogućuje vezu programa s datotekom: otvara tok podataka za čitanje i/ili pisanje u datoteku `filename`
- ukoliko je otvaranje toka podataka uspješno, vraća pokazivač na strukturu tipa `FILE`. Ako otvaranje nije uspješno, vraća `NULL`
  - tip strukture `FILE` je definiran u `<stdio.h>`
  - pokazivač na strukturu tipa `FILE` koristi se kao tok podataka (*stream*) u raznim funkcijama za rad s datotekama: `fprintf`, `fgetc`, `fclose` ...

10

## fopen

<stdio.h>

---

- **filename** ime datoteke (uobičajeno na disku)

Primjeri:

- Unix

```
FILE *pod1, *pod2, *pod3;
pod1 = fopen("podaci.txt", "w");
pod2 = fopen("glavni.c", "r");
pod3 = fopen("/usr/source/glavni.c", "w+");
```
- DOS

```
FILE *pod1, *pod2, *pod3;
pod1 = fopen("podaci.txt", "w");
pod2 = fopen("glavni.c", "r");
pod3 = fopen("c:\\tmp\\pomocna.dat", "w+");
```

11

## fopen

<stdio.h>

---

- **mode** način korištenja
  - "w" pisanje (ako datoteka ne postoji, stvara se;  
ako postoji, briše se sadržaj;  
nije dopušteno čitanje)
  - "a" pisanje (ako datoteka ne postoji, stvara se;  
ako postoji, podaci se dodaju na kraj;  
nije dopušteno čitanje)
  - "r" čitanje (ako datoteka ne postoji, vraća NULL;  
nije dopušteno pisanje)
  - "r+" čitanje i pisanje (ako datoteka ne postoji, vraća NULL)
  - "w+" čitanje i pisanje (ako datoteka ne postoji, stvara se)
  - "a+" čitanje i pisanje (ako datoteka ne postoji, stvara se;  
podaci se dodaju na kraj)

Na DOS-u za čitanje i pisanje neformatiranih datoteka treba dodati **b** npr. "**rb**"

12

## **fclose**

**<stdio.h>**

---

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

- prekida vezu programa s datotekom: zatvara tok podataka `stream`
- ukoliko je zatvaranje toka podataka uspješno, vraća 0, inače vraća EOF

Primjer:

```
FILE *pod;  
pod = fopen("podaci.txt", "r");  
... /* fscanf, fgetc, ... iz toka podataka pod */  
fclose(pod);
```

13

## Podjela datoteka prema načinu upisa

---

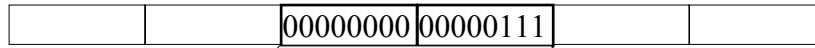
- formatirane datoteke (tekstualne, *text*)
- neformatirane datoteke (binarne, *binary*)

14

## Formatirane (tekstualne) datoteke

Središnja memorija

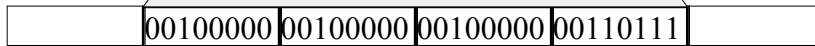
```
short int i=7;
```



```
fprintf(tokPod, "%4d", i);
```

Datoteka

32(' ') 32(' ') 32(' ') 55('7')



početna  
pozicija

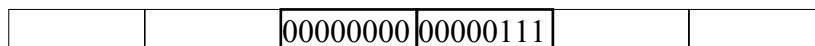
nova pozicija

15

## Neformatirane (binarne) datoteke

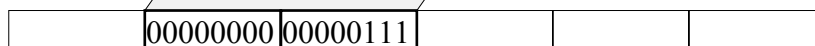
Središnja memorija

```
short int i=7;
```



```
fwrite(&i, sizeof (i), 1, tokPod);
```

Datoteka



početna  
pozicija

nova pozicija

16

## Funkcije za čitanje iz formatirane datoteke

---

```
int fgetc(FILE *stream);           <stdio.h>
```

- čita jedan znak iz toka podataka `stream`. Vraća pročitani znak ako je čitanje uspjelo i nije pročitano kraj datoteke. Vraća `EOF` ako čitanje nije uspjelo ili se dogodila pogreška

```
int fscanf (FILE *stream,          <stdio.h>
            const char *format
            arg_1, ..., arg_n);
```

- jedina razlika u odnosu na `scanf` je u tome što čita iz `stream` (funkcija `scanf` uvijek čita iz `stdin`)

17

## Funkcije za čitanje iz formatirane datoteke

---

```
char *fgets(char *s,              <stdio.h>
            int n,
            FILE *stream);
```

`s` pokazivač na područje u memoriji gdje će biti smješteni učitani znakovi

`n` najveća dopuštena duljina učitano niza (uključujući znak `'\0'`)

- učitava znakove u niz `s` dok ne učitati `'\n'` ili učitati `n-1` znakova ili dospije do kraja datoteke. Na učitani niz (koji može, ali ne mora sadržavati `'\n'`) **dodaje** znak `'\0'` (za razliku od funkcije `gets` koja učitani `'\n'` zamjenjuje s `'\0'`)
- u slučaju pogreške ili pokušaja čitanja nakon kraja datoteke, vraća `NULL`, inače vraća pokazivač na učitani niz

18

**Primjer:** Prepisati sadržaj datoteke `prog.c` na zaslon (ekran). Datoteku čitati pomoću funkcije `fgetc`

---

```
#include <stdio.h>
#include <stdlib.h>
int main () {
    FILE *dat;
    int c;
    dat = fopen ("prog.c", "r");
    if (dat == NULL) {
        printf ("Datoteka se ne moze otvoriti\n");
        exit (1);
    }
    while ((c = fgetc (d)) != EOF) {
        putchar (c);
    }
    fclose (d);
    return 0;
}
```

19

## Funkcije za pisanje u formatiranu datoteku

---

`int fputc(int c, FILE *stream);`     `<stdio.h>`

- jednako kao `putchar`, osim što na `stdout`, ispisuje na `stream`

`int fprintf (FILE *stream,`     `<stdio.h>`  
                  `const char *format,`  
                  `arg_1, ..., arg_n)`

- jednako kao `printf`, osim što na `stdout`, ispisuje na `stream`

`int fputs(char *s, FILE *stream);`     `<stdio.h>`

- slično kao `puts`
  - umjesto na `stdout`, ispisuje na `stream`
  - ne ispisuje dodatni znak `'\n'`

20

**Primjer:** Prepisati sadržaj datoteke `stara` u datoteku `nova`. Koristiti funkcije `fgetc` i `fputc`

---

```
#include <stdio.h>
int main () {
    FILE *du, *di;
    int c;

    du = fopen ("stara", "r");
    di = fopen ("nova", "w");
    /* trebalo bi provjeriti uspjesnost otvaranja */

    while ((c = fgetc (du)) != EOF)
        fputc (c, di);
    fclose (du);
    fclose (di);

    return 0;
}
```

21

**Primjer:** Prepisati sadržaj datoteke `stara` u datoteku `nova`. Koristiti funkcije `fgets` i `fputs`

---

```
#include <stdio.h>
#include <stdlib.h>
#define MAXLIN 80
int main () {
    FILE *du, *di;
    char linija[MAXLIN];

    du = fopen ("stara", "r");
    di = fopen ("nova", "w");
    /* trebalo bi provjeriti uspjesnost otvaranja */

    while (fgets(linija, MAXLIN, du) != NULL ) {
        fputs (linija, di);
    }

    fclose (du);
    fclose (di);

    return 0;
}
```

22

## Funkcija za pisanje u neformatiranu datoteku

```
size_t fwrite(void *ptr,           <stdio.h>
               size_t size,
               size_t n,
               FILE *stream);
```

**ptr** adresa u memoriji na kojoj se nalaze podaci koje treba zapisati

**size** veličina jednog objekta kojeg treba zapisati

**n** broj objekata koje treba zapisati

- u tok podataka **stream** zapisuje **n** objekata (od koji je svaki veličine **size** bajtova). Podaci koje treba zapisati nalaze se u memoriji na lokaciji određenoj pokazivačem **ptr**
- funkcija vraća broj uspješno zapisanih objekata. Ako se pri pisanju dogodi pogreška, vratit će broj manji od **n**

23

**Primjer:** u neformatiranu datoteku podaci zapisati jedan podatak tipa `long` i 3 člana `double` polja

```
long m = 10L; double x[3] = {1.5, -3.5, 3.25};
int n1, n2; FILE *izTok;
izTok = fopen ("podaci", "wb");
n1 = fwrite (&m, sizeof(m), 1, izTok);
if (n1 < 1) {
    printf("Zapisivanje m nije uspjelo\n");
    exit(-1);
}
n2 = fwrite (x, sizeof(x[0]), 3, izTok);
if (n2 < 3) {
    printf("Zapisano je samo %d clanova\n", n2);
    exit(-2);
}
fclose(izTok);
```

zbog pretpostavke korištenja DOS operacijskog sustava

24



**Primjer:** iz neformatirane datoteke `podaci` učitati jedan podatak tipa `long` i 3 člana `double` polja

```
long m; double x[3]; int n1, n2;
FILE *ulTok;
ulTok = fopen ("podaci", "rb");
n1 = fread (&m, sizeof(m), 1, ulTok);
if (n1 < 1) {
    printf("Citanje m nije uspjelo\n");
    exit(-1);
}
n2 = fread (x, sizeof(x[0]), 3, ulTok);
if (n2 < 3) {
    printf("Ucitano je samo %d clanova\n", n2);
    exit(-2);
}
fclose(ulTok);
```

27

**Primjer:** prepisati sadržaj formatirane datoteke `rez.txt` u neformatiranu datoteku `rez.bin`. (ime: 8+1 znakova, prezime: 8+1 znakova, broj bodova: int).

Sadržaj datoteke `rez.txt` promatran editorom: 

Iva Pek 156
Ante Horvat 12

Sadržaj datoteke `rez.txt` u heksadekadskom prikazu:

```
49 76 61 20 50 65 6B 20 31 35 36 0A 41 6E
74 65 20 48 6F 72 76 61 74 20 31 32
```

Sadržaj datoteke `rez.bin` u heksadekadskom prikazu:

```
49 76 61 00 ?? ?? ?? ?? ?? 50 65 6B 00 ??
?? ?? ?? ?? 00 00 00 9C 41 6E 74 65 00 ??
?? ?? ?? 48 6F 72 76 61 74 00 ?? ?? 00 00
00 0C
```

Što je zapis u `rez.txt`, a što je zapis u `rez.bin` ?

28

### Primjer: rješenje u C-u

---

```
FILE *ulTok, *izTok;
char ime[8+1], prez[8+1]; int brBod;
ulTok = fopen ("rez.txt", "r");
izTok = fopen ("rez.bin", "wb");
while (fscanf(ulTok, "%s%s%d",
             ime, prez, &brBod) == 3) {
    fwrite(ime, sizeof(ime), 1, izTok);
    fwrite(prez, sizeof(prez), 1, izTok);
    fwrite(&brBod, sizeof(brBod), 1, izTok);
}
fclose (ulTok);
fclose (izTok);
```

29

## Struktura (zapis)

---

- Struktura je složeni tip podatka čiji se elementi razlikuju po tipu:

```
struct naziv_strukture {
    tip_elementa_1 ime_elementa_1;
    tip_elementa_2 ime_elementa_2;
    ...
    tip_elementa_n ime_elementa_n;
};
```

```
struct osoba {
    char jmbg[13+1];
    char prezime[40+1];
    char ime[40+1];
    int visina;
    float tezina;
};
```

- Ovime nije definirana varijabla u koju se može pohraniti konkretan podatak. Struktura je ovime tek deklarirana (opisana).

30

## Definicija varijabli tipa strukture

---

```
struct naziv_strukture var1, var2, ... , varN;
```

npr.

```
struct osoba o1, o2;
```

- Moguće je istovremeno deklarirati strukturu i definirati varijable:

```
struct tocka {  
    int x;  
    int y;  
} t1, t2, t3;
```

Može i ovako, ali je manje pregledno:

```
struct tocka {  
    int x, y;  
} t1, t2, t3;
```

```
struct tocka t4;
```

31

## Deklaracija strukture bez naziva

---

- Naziv strukture može se izostaviti ako je potrebno definirati jednu ili više varijabli tipa strukture, a takva struktura se drugdje neće koristiti:

```
struct {  
    int dan;  
    int mjesec;  
    int godina;  
} datum;
```

32

## Strukture i typedef

---

- Deklaracija strukture često se koristi zajedno s `typedef`

```
typedef struct {
    int x;
    int y;
} tocka;
tocka t1, t2;
```

33

## Postavljanje i korištenje vrijednosti elemenata strukture

---

```
structVarijabla.element = vrijednost;  
vrijednost = structVarijabla.element;
```

npr.

```
scanf ("%s %s %s %d",
        o1.jmbg, o1.prezime, o1.ime, &o1.visina);
o1.tezina = 75.5;
t1.x = 7; t1.y = 2;
t2.x = 5; t2.y = 3;
udaljenost = sqrt(pow(t1.x - t2.x, 2.) +
                  pow(t1.y - t2.y, 2.));
printf ("Datum = %d.%d.%d\n", datum.dan,
        datum.mjesecc, datum.godina);
```

34

## Složene strukture

---

- Moguće je definiranje podatkovne strukture proizvoljne složenosti jer pojedini element može također biti `struct`:

```
struct student {
    int maticni_broj;
    struct osoba osobni_podaci;
    struct osoba otac;
    struct osoba majka;
};
```

35

## Složene strukture

---

- Alternativno, korištenjem naredbe `typedef`:

```
typedef struct {
    char jmbg[13+1];
    char prezime[40+1];
    char ime[40+1];
    int visina;
    float tezina;
} osoba;
```

```
typedef struct {
    int maticni_broj;
    osoba podaci_stud;
    osoba podaci_otac;
    osoba podaci_majka;
} student;
student pero;
pero.podaci_majka.tezina = 114.5;
```

36

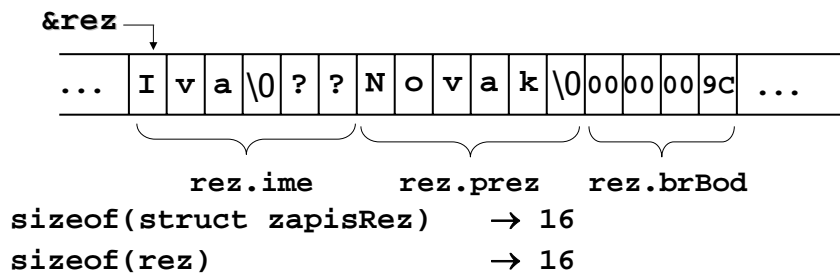
## Primjeri struktura

```

struct zapisRez {
    char ime[5+1];
    char prez[5+1];    strcpy(rez.ime, "Iva");
    int brBod;         strcpy(rez.prez, "Novak");
} rez;                rez.brBod = 156;

```

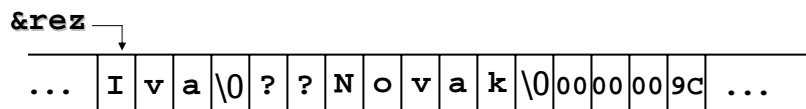
Slično kao i članovi polja, članovi jedne strukture uvijek su u memoriji pohranjeni kao susjedni elementi:



37

## Korištenje struktura za čitanje i pisanje zapisa neformatiranih datoteka

Uobičajeno je pisanje/čitanje zapisa neformatiranih datoteka obavljati uz pomoć struktura.



```
n = fwrite (&rez, sizeof (rez), 1, izTok);
```

Sadržaj datoteke u heksadekadskom prikazu:

```

49 76 61 00 ?? ?? 4E 6F 76 61
6B 00 00 00 00 9C

```

```
n = fread (&rez, sizeof (rez), 1, ulTok);
```

38

**Primjer:** iz neformatirane datoteke `pod.dat` učitati ime (7+1 znakova), prezime (7+1 znakova) i brojBod (int) za dvoje studenata

---

Rješenje u kojem se struktura **ne koristi**

```
char ime1[7+1]; char prez1[7+1]; int brBod1;
char ime2[7+1]; char prez2[7+1]; int brBod2;
ulTok = fopen("pod.dat", "rb");

fread(ime1, sizeof(ime1), 1, ulTok);
fread(prez1, sizeof(prez1), 1, ulTok);
fread(&brBod1, sizeof(brBod1), 1, ulTok);
fread(ime2, sizeof(ime2), 1, ulTok);
fread(prez2, sizeof(prez2), 1, ulTok);
fread(&brBod2, sizeof(brBod2), 1, ulTok);
```

39

**Primjer:** iz neformatirane datoteke `pod.dat` učitati ime (7+1 znakova), prezime (7+1 znakova) i brojBod (int) za dvoje studenata

---

Rješenje u kojem se **koristi** struktura

```
struct zapisRez {
    char ime[7+1];
    char prez[7+1];
    int brBod;
} rez1, rez2;
ulTok = fopen("pod.dat", "rb");


fread(&rez1, sizeof(rez1), 1, ulTok);
fread(&rez2, sizeof(rez2), 1, ulTok);
```

40

## Trenutna pozicija u datoteci

Čitanje i pisanje uvijek se obavlja od trenutne pozicije u datoteci.

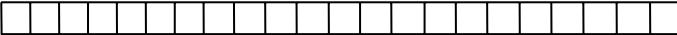
```
izTok = fopen ("dat", "wb");
```



```
↑
```

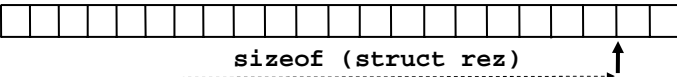
```
int m = 10;
```

```
fwrite (&m, sizeof(m), 1, izTok);
```



```
sizeof(m)↑
```

```
fwrite (&rez, sizeof(rez), 1, izTok);
```



```
sizeof (struct rez)↑
```

41

## Promjena pozicije u datoteci funkcija **fseek**

```
int fseek(FILE *stream,          <stdio.h>  
          long offset,  
          int whence);
```

**offset** pomak u bajtovima u odnosu na *whence*

**whence** **SEEK\_SET** - početak datoteke

**SEEK\_CUR** - trenutna pozicije

**SEEK\_END** - kraja datoteke

- pomiče trenutnu poziciju u datoteci
- vraća 0 ako je pomak uspio, inače vraća broj različit od 0

42

## Promjena pozicije u datoteci funkcija `fseek`

---

Pozicioniranje na početak datoteke:

```
fseek (tokPod, 0L, SEEK_SET);
```

Pozicioniranje na kraj datoteke:

```
fseek (tokPod, 0L, SEEK_END);
```

Pozicioniranje na  $n$ -ti bajt:

```
fseek (tokPod, (long) n, SEEK_SET);
```

Pomak unatrag za  $n$  bajtova:

```
fseek (tokPod, - (long) n, SEEK_CUR);
```

43

## Trenutna pozicija u datoteci - funkcija `ftell`

---

```
long ftell(FILE *stream);          <stdio.h>
```

- vraća trenutnu poziciju u datoteci izraženu u broju bajtova od početka datoteke
- u slučaju pogreške vraća `-1L`

**Primjer:** Ispisati trenutnu veličinu datoteke.

...

```
fseek (tokPod, 0L, SEEK_END);
```

```
printf ("Velicina datoteke: %d bajtova\n",  
        ftell (tokPod));
```

...

44

## Preusmjeravanje (redirekcija) tokova podataka

Tokove podataka `stdout` i `stdin` je moguće preusmjeriti u trenutku pokretanja programa. Ovdje je prikazan način preusmjeravanja toka podataka `stdout`:

```
c:\pipi>pipi.exe
```

Ispis na `stdout` rezultira ispisom na zaslonu

```
c:\pipi>pipi.exe > dat.txt
```

Ispis na `stdout` sada je preusmjeren u datoteku `dat.txt`

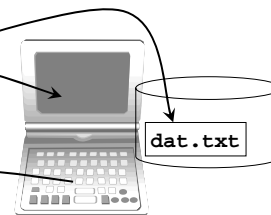
program `pipi.exe`

```
printf(...)  
putchar(...)  
puts(...)
```

```
scanf(...)  
getchar()  
gets(...)
```

`stdout`

`stdin`



45

## Tok podataka `stderr`

U trenutku pokretanja programa automatski se otvara i tok podataka `stderr` (također ispisuje na zaslon ekrana). Na `stderr` ne djeluje prikazana redirekcija

```
c:\pipi>pipi.exe > dat.txt
```

program `pipi.exe`

```
printf(...)  
putchar(...)  
puts(...)
```

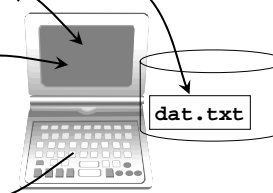
```
fprintf(stderr ...)  
fputc(stderr ...)  
fputs(stderr ...)
```

```
scanf(...)  
getchar()  
gets(...)
```

`stdout`

`stderr`

`stdin`



46

## Primjer: preusmjerenje standardnog izlaza

---

```
lisi.c
#include <stdio.h>
int main() {
    fprintf(stdout, "Prvi\n");
    printf("Drugi\n");
    fprintf(stderr, "Treci\n");
    puts("Cetvrti");
    fputs("Peti\n", stderr );
    fputs("Sesti\n", stdout);
    return 0;
}
```

```
c:\pipi>lisi.exe > dat.txt
```

U datoteku `dat.txt` upisat će se:            Na zaslону će se ispisati:

Prvi	Treci
Drugi	Peti
Cetvrti	
Sesti	

47

## Slijedne i direktne datoteke

---

- zahvaljujući funkciji `fseek` moguće je pozicionirati se na bilo koju poziciju (redni broj bajta) unutar datoteke i obaviti operaciju čitanja ili pisanja
- međutim, kako pristupiti n-tom **zapisu** u datoteci?

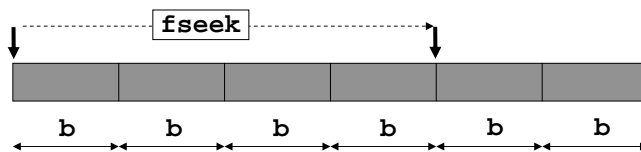
48

## Direktne datoteke

- ako su svi zapisi u datoteci jednake duljine (npr. duljina svakog zapisa je  $b$  bajtova), moguće je izravno ("direktno") pozicioniranje na početak  $n$ -tog zapisa, npr. ovako:

```
fseek (tok, (long)(n-1)*b, SEEK_SET);
```

**Primjer:** pozicioniranje na početak 5. zapisa datoteke



Ako je zapis u datoteci definiran strukturom naziva `zapis`, pozicioniranje na početak  $n$ -tog zapisa se obavlja ovako:

```
fseek (tok, (long)(n-1)*sizeof(struct zapis), SEEK_SET);
```

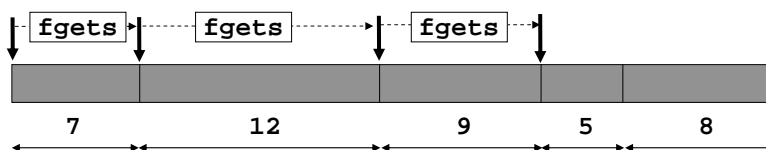
49

## Slijedne datoteke

- ako svi zapisi u datoteci NISU jednake duljine,  $n$ -tom zapisu se može pristupiti jedino tako da se redom ("slijedno"), od početka datoteke, prvo pročita prethodnih  $n-1$  zapisa

```
for (i = 1; i <= n-1; i++)  
    fgets(...); ili fscanf(...); ili fread(...);
```

**Primjer:** pozicioniranje na početak 4. zapisa datoteke



50



## Praktična primjena direktnih datoteka

---

- mogućnost direktnog pristupa n-tom zapisu u datoteci praktično je iskoristiva samo onda kada redni broj zapisa odgovara nekom *ključu potrage*, npr:
  - zapis o osobi s matičnim brojem n smješten je kao n-ti zapis u datoteci (ključ potrage je matični broj osobe)
  - zapis o srednjoj temperaturi i vlažnosti zraka za n-ti dan 2006. godine jest n-ti zapis u datoteci (ključ potrage je redni broj dana u 2006. godini)
- moguće je da će neki zapisi u tim datotekama biti "prazni", npr.
  - u datoteku su upisani podaci o 100 osoba, a ne postoje osobe s matičnim brojevima 2, 17, 33, 34
  - mjerenja temperature i vlažnosti se ne obavljaju nedjeljom
- kako organizirati direktnu datoteku s poštanskim brojevima mjesta i nazivima mjesta (poštanski brojevi od 10000 do 60000)?

53

### **Primjer:** Prijepis slijedne formatirane u direktnu neformatiranu datoteku

---

- sadržaj slijedne formatirane datoteke "upis.txt" treba prepisati u direktnu neformatiranu datoteku "tezine"
- u svakom zapisu datoteke "upis.txt" nalazi se matični broj (4 znamenke), prezime i ime (40 znakova), godina rođenja (4 znamenke) i tjelesna težina (realni broj s 3 cijela mjesta i jednom decimalom)
- zapis datoteke "tezine" sastoji se od matičnog broja (short), prezimena i imena (40+1 znak), godine starosti (short) i težine (float). Redni broj zapisa u datoteci "tezine" mora odgovarati matičnom broju
- ako otvaranje datoteke, pozicioniranje u datoteci "tezine" ili pisanje u datoteku "tezine" ne uspije, ispisati poruku i završiti program

54

**Primjer:** Prijepis slijedne formatirane u direktnu neformatiranu datoteku

---

Primjer sadržaja datoteke "upis.txt":

```
          1          2          3          4          5
12345678901234567890123456789012345678901234567890123
0001Peric Pero                                1947 76.8
0002Ivic Ivo                                  1952 86.3
0006Markovic Marko                            1940101.5
0012Petrovic Petar                            1972 67.8
0003Anic Ana                                  1968 56.7
```

55

**Primjer:** Prijepis slijedne formatirane u direktnu neformatiranu datoteku

---

Primjer sadržaja datoteke "tezine":

redni broj bajta "na kojem započinje zapis"

0	1	Peric Pero	48	76.8
49	2	Ivic Ivo	43	86.3
98	3	Anic Ana	27	56.7
147	?	?	?	?
196	?	?	?	?
245	6	Markovic Marko	55	101.5
294	?	?	?	?
343	?	?	?	?
392	?	?	?	?
441	?	?	?	?
490	?	?	?	?
539	12	Petrovic Petar	23	67.8

56

**Primjer:** Prijepis slijedne formatirane u direktnu neformatiranu datoteku

---

```
#include <stdio.h>
#include <stdlib.h>
void fatal (char *poruka) {
    fputs (poruka, stderr);
    fputs ("\n", stderr);
    exit (1);
}
int main () {
    FILE *du, *di;
    struct zapis_osobe {
        short mat_br;
        char prez_ime[40+1];
        short starost;
        float tezina;
    } zapis;
```

57

**Primjer:** Prijepis slijedne formatirane u direktnu neformatiranu datoteku

---

```
short tek_godina, god_rod;

if ((du = fopen ("upis.txt", "r")) == NULL)
    fatal ("Ne mogu otvoriti datoteku \"upis.txt\");

if ((di = fopen ("tezine", "wb")) == NULL)
    fatal ("Ne mogu otvoriti datoteku \"tezine\");

/* tekuca godina - program nece raditi dobro 2007 */
tek_godina = 2006;
```

58

### Primjer: Prijepis slijedne formatirane u direktnu neformatiranu datoteku

---

```
while (fscanf (du, "%4hd%40[^\n]%4hd%5f",
             &zapis.mat_br,
             zapis.prez_ime,
             &god_rod,
             &zapis.tezina) == 4) {
    zapis.starost = tek_godina - god_rod;
    if (fseek (di, (long)(zapis.mat_br-1) *
              sizeof (zapis), SEEK_SET) != 0)
        fatal("Nije uspjelo pozicioniranje u \"tezine\");
    if (fwrite (&zapis, sizeof (zapis), 1, di) != 1)
        fatal("Nije uspjelo zapisivanje u \"tezine\");
}
fclose (du);
fclose (di);
return 0;
}
```

59

### Primjer: Korištenje direktne datoteke

---

- izračunati prosječnu tjelesnu težinu osoba čiji su podaci upisani u datoteku "tezine" (datoteku koja je nastala obavljanjem programa iz prethodnog primjera). Prosječnu težinu ispisati na zaslou
- nakon toga treba uzastopno učitavati s tipkovnice matične brojeve. Za svaki učitani matični broj na zaslou ispisati podatke o osobi, te posebnu napomenu u slučaju da osoba ima tjelesnu težinu manju od prosjeka. Učitavanje matičnih brojeva s tipkovnice treba završiti kada pozicioniranje na zapis ne uspije ili se upiše matični broj nepostojeće osobe

60

## Primjer: Korištenje direktne datoteke

---

```
#include <stdio.h>
#include <stdlib.h>
int main () {
    FILE *du;
    struct zapis_osobe {
        short mat_br;
        char prez_ime[40+1];
        short starost;
        float tezina;
    } zapis;

    int brojZapisa = 0, rbrZapis = 0;
    short mat_br;
    float suma = 0.0f, prosjek;
```

61

## Primjer: Korištenje direktne datoteke

---

```
/* prvo slij. pročitati sve zapise iizr. prosjek */
du = fopen ("tezine", "rb");

while (fread(&zapis, sizeof(zapis), 1, du) == 1) {
    rbrZapis++;
    if (zapis.mat_br == rbrZapis) {
        suma = suma + zapis.tezina;
        brojZapisa++;
    }
}

prosjek = suma/brojZapisa;
printf("Prosjecna tezina je: %5.2f\n", prosjek);
```

62

## Primjer: Korištenje direktne datoteke

---

```
while (1) {
    printf ("\nUnesite maticni broj:");
    scanf ("%hd", &mat_br);
    if (fseek (du, (long) (mat_br-1) *
              sizeof (zapis), SEEK_SET) != 0)
        break;
    fread (&zapis, sizeof (zapis), 1, du);
    if (zapis.mat_br != mat_br)
        break;
    printf ("%4d %s %4d %5.2f\n", zapis.mat_br,
            zapis.prez_ime, zapis.starost, zapis.tezina);
    if (zapis.tezina < prosjek)
        printf ("Osoba je laksa od prosjeka.\n");
}
fclose(du);
return 0;
}
```

63

## Rezultat izvođenja programa

---

Prosjecna tezina je: 77.82

Unesite maticni broj:1

1 Peric Pero

59 76.80

Osoba je laksa od prosjeka.

Unesite maticni broj:2

2 Ivic Ivo

54 86.30

Unesite maticni broj:6

6 Markovic Marko

66 101.50

Unesite maticni broj:11

64

## Kraj semestra

---

Predavači, ovdje završite predavanja, a preostalo slobodno vrijeme iskoristite za uvježbavanje gradiva i pripremu za završni ispit!

Studentima želimo uspješan završetak semestra, a gradivo u nastavku prezentacije može dobro poslužiti kao priprema za slijedeći semestar.