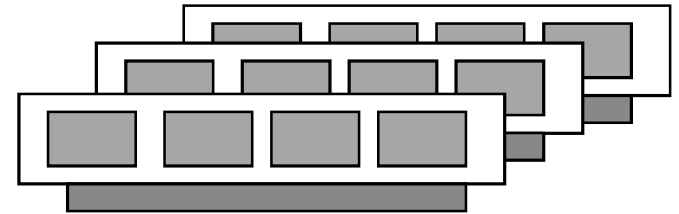


Datoteke

Memorija računala

1) privremena (unutarnja)

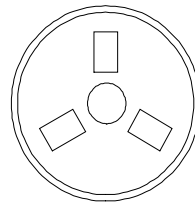
RAM (Random Access Memory)



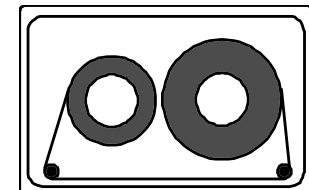
2) stalna (vanjska)

a) sa slijednim pristupom podacima, npr.

magnetske trake

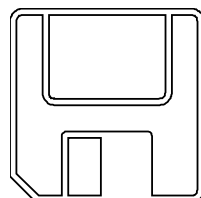


streamer trake

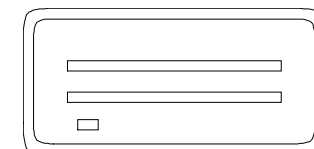


b) s direktnim pristupom podacima, npr.

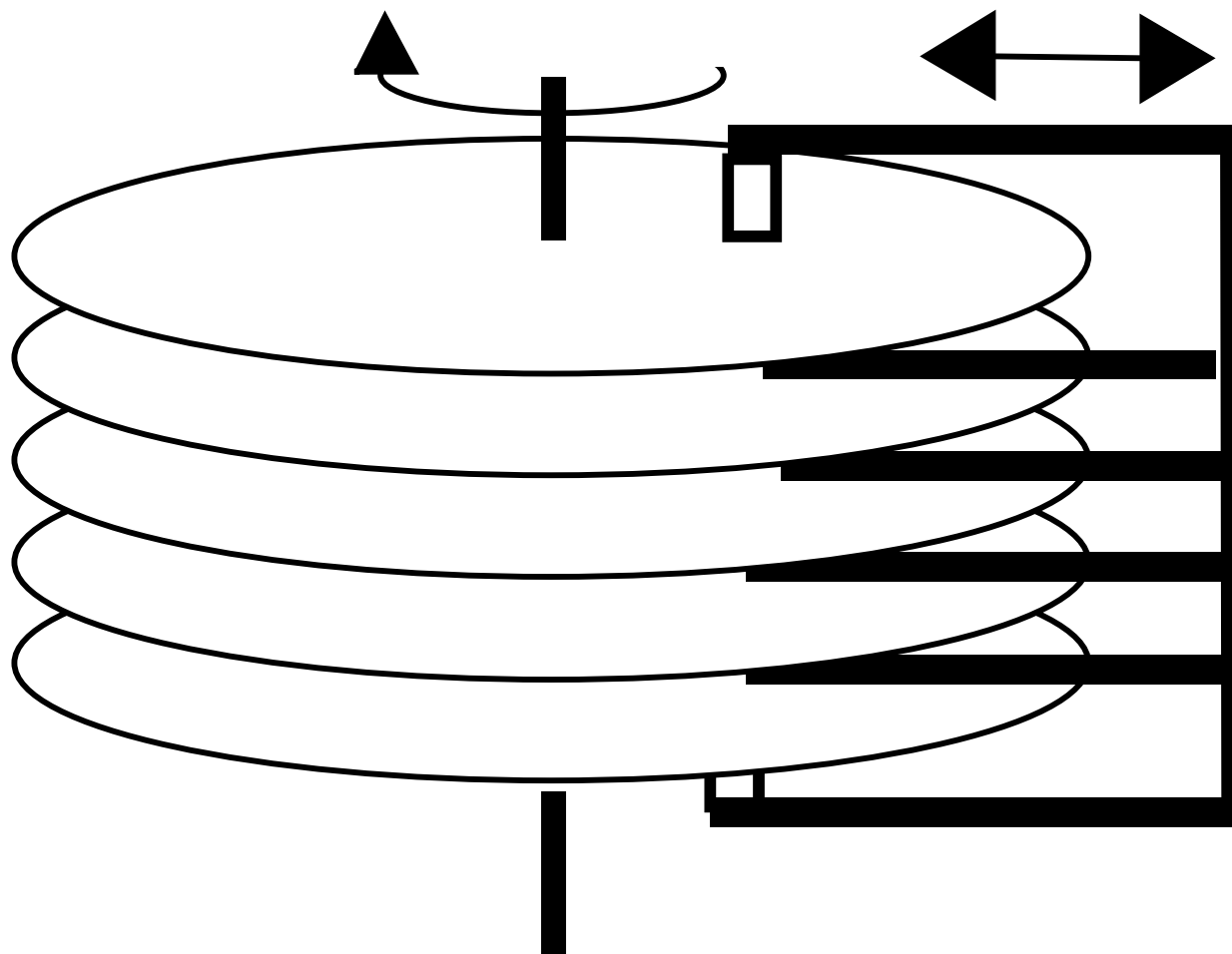
diskete



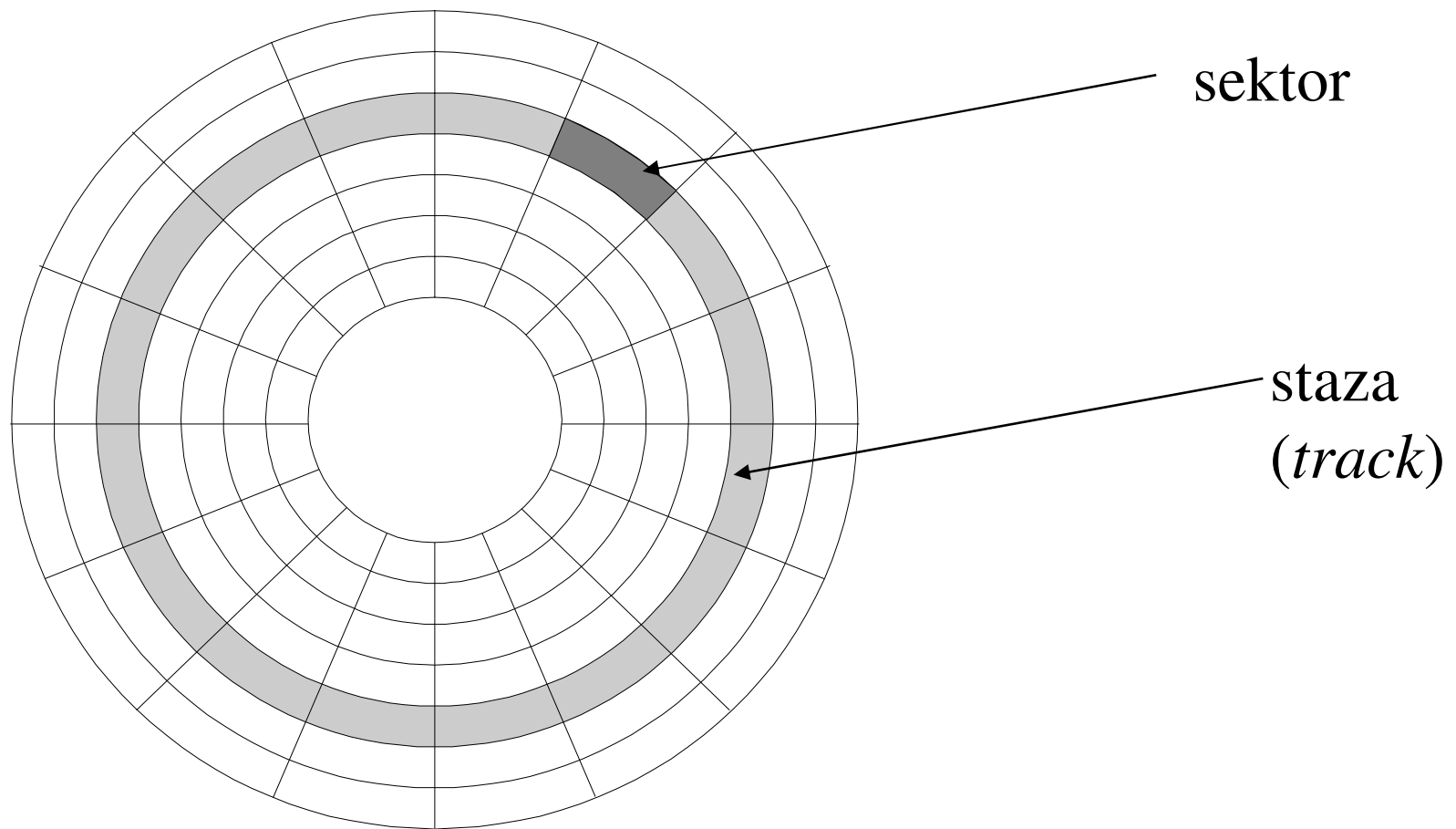
magnetski diskovi



Shematski prikaz magnetskog diska



Fizička organizacija magnetskog diska





























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	 mtmcdae.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

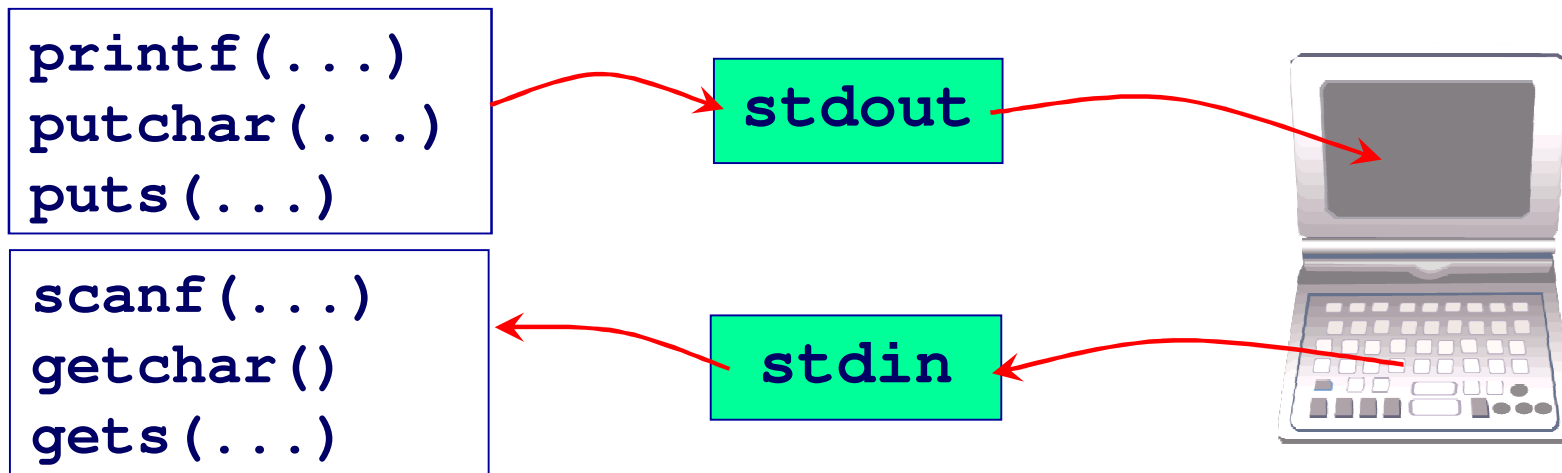
Direktoriji i datoteke

- **Datoteka:** imenovani skup podataka na mediju za pohranu (traka, disk, disketa, CD, *memory stick*, ...)
- **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
- **Zapis:** skup susjednih podataka unutar datoteke koji se obrađuje kao cjelina

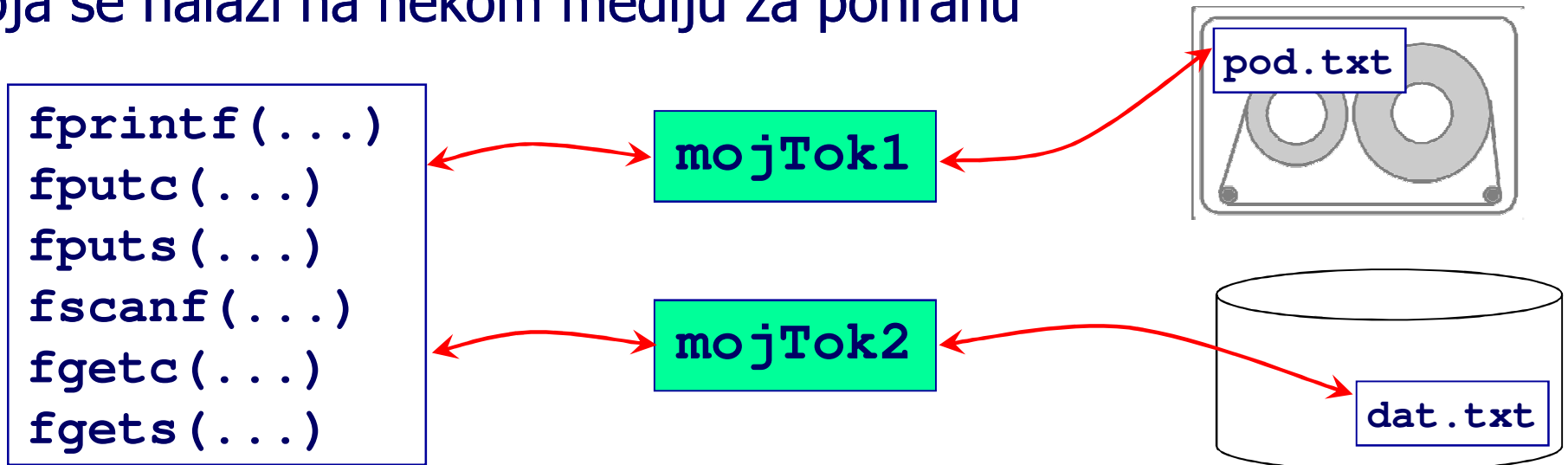
Tok podataka (*stream*)

- 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>`

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



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`?

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 uspjelo, vraća pokazivač na strukturu tipa `FILE`. Ako otvaranje nije uspjelo, 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` ...

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+");
```

ili

```
pod3 = fopen("c:/tmp/pomocna.dat", "w+");
```

■ 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)

Za čitanje i pisanje neformatiranih datoteka treba dodati **b** npr. "**rb**"

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);
```

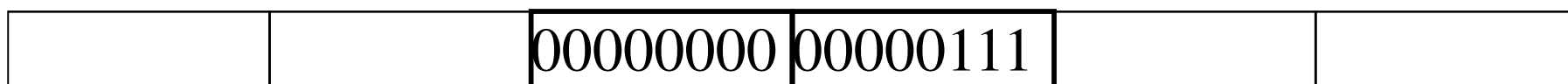
Podjela datoteka prema načinu upisa

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

Formatirane (tekstualne) datoteke

Središnja memorija

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



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

Datoteka

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



početna
pozicija



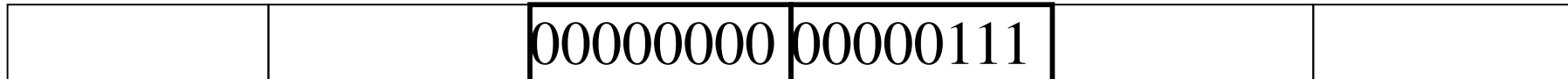
nova pozicija



Neformatirane (binarne) datoteke

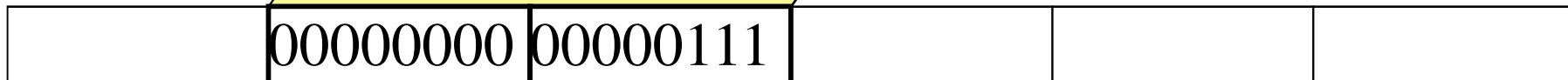
Središnja memorija

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



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

Datoteka



početna
pozicija



nova pozicija



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 je pročitana oznaka kraja datoteke.

```
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`)

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 učitanoog niza (uključujući znak '\0')
- učitava znakove u niz **s** dok ne učitava '\n' ili učitava 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

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 (dat)) != EOF) {
        putchar (c);
    }
    fclose (dat);
    return 0;
}
```

Funkcije za pisanje u formatiranu datoteku

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

- jednako kao `putc`, 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`'

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;
}
```

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;
}
```

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**

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);
```

radi neformatirane
(binarne) datoteke

Sadržaj datoteke `podaci` iz prethodnog primjera:

`10L` → `00 00 00 0A`₁₆
`1.5` → `3F F8 00 00 00 00 00 00`₁₆
`-3.5` → `C0 0C 00 00 00 00 00 00`₁₆
`3.25` → `40 0A 00 00 00 00 00 00`₁₆

Sadržaj datoteke (u heksadekadskom prikazu):

<code>10L</code>				<code>1.5</code>								<code>-3.5</code>			
⏟				⏟								⏟			
<code>00</code>	<code>00</code>	<code>00</code>	<code>0A</code>	<code>3F</code>	<code>F8</code>	<code>00</code>	<code>00</code>	<code>00</code>	<code>00</code>	<code>00</code>	<code>00</code>	<code>00</code>	<code>00</code>	<code>C0</code>	<code>0C</code>
<code>00</code>	<code>00</code>	<code>00</code>	<code>00</code>	<code>00</code>	<code>00</code>	<code>40</code>	<code>0A</code>	<code>00</code>	<code>00</code>	<code>00</code>	<code>00</code>	<code>00</code>	<code>00</code>	<code>00</code>	<code>00</code>
						⏟									
						<code>3.25</code>									

Funkcija za čitanje iz neformatirane datoteke

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

ptr adresa u memoriji na koju će se smjestiti učitani podaci

size veličina jednog objekta kojeg treba učitati

n broj objekata koje treba učitati

- iz toka podataka **stream** učitava **n** objekata (od koji je svaki veličine **size** bajtova) i smješta ih u memoriju na lokaciju određenu pokazivačem **ptr**
- funkcija vraća broj uspješno učitanih objekata. Ako pri čitanju naiđe na kraj datoteke ili se dogodi pogreška, vratit će broj manji od **n**

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);
```

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` ?

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);
```

Ponavljanje: Struktura (zapis)

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

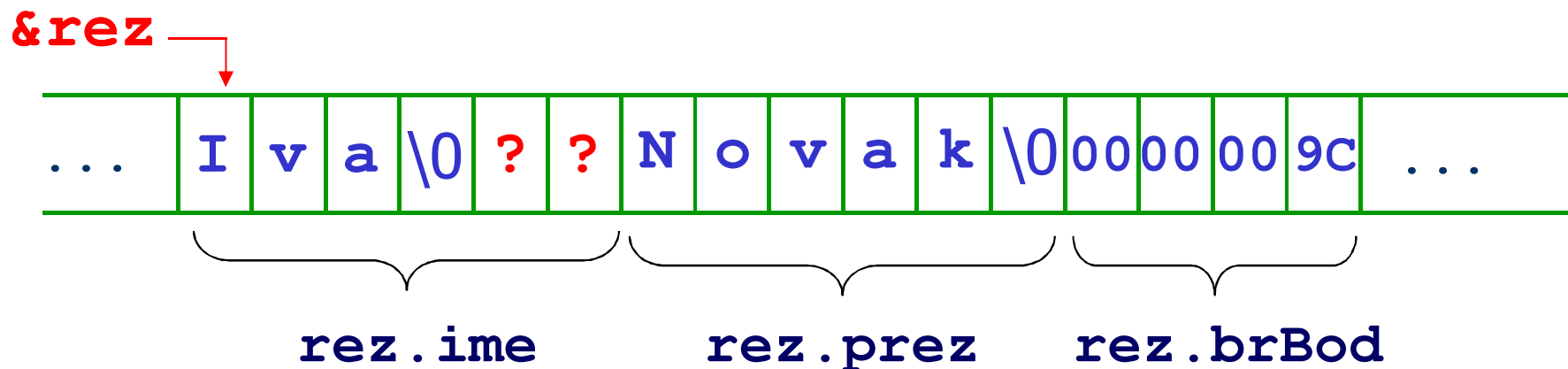
```
struct naziv_strukture {  
    tip_elementa_1   ime_elementa_1;  
    tip_elementa_2   ime_elementa_2;  
    ...  
    tip_elementa_n   ime_elementa_n;  
};
```

```
struct osoba {  
    char jmbg[13+1];  
    char prezime[40+1];  
    char ime[40+1];  
    int visina;  
    float tezina;  
};  
struct osoba o1, o2;  
struct osoba o3;
```

Pohrana sadržaja strukture u memoriji

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

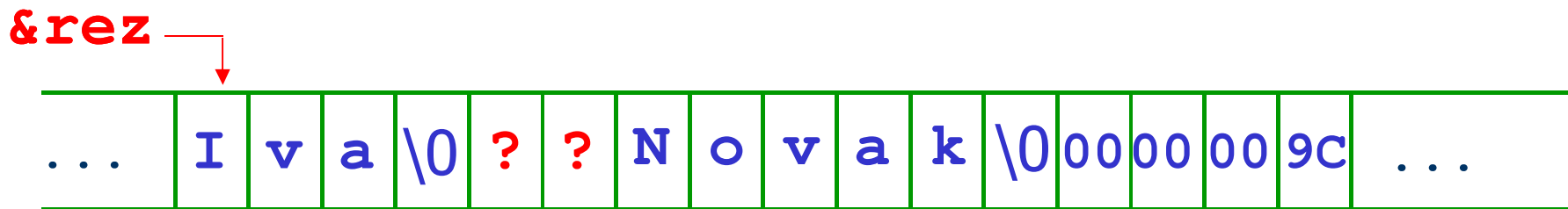
Slično kao i članovi polja, članovi jedne strukture uvijek su u memoriji pohranjeni kao susjedni elementi:



```
sizeof(struct zapisRez)    → 16
sizeof(rez)                → 16
```

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);
```

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);
```

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);
```

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) ↑
----->

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

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);
```

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));
```

...

Preusmjeravanje (redirekcija) tokova podataka

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

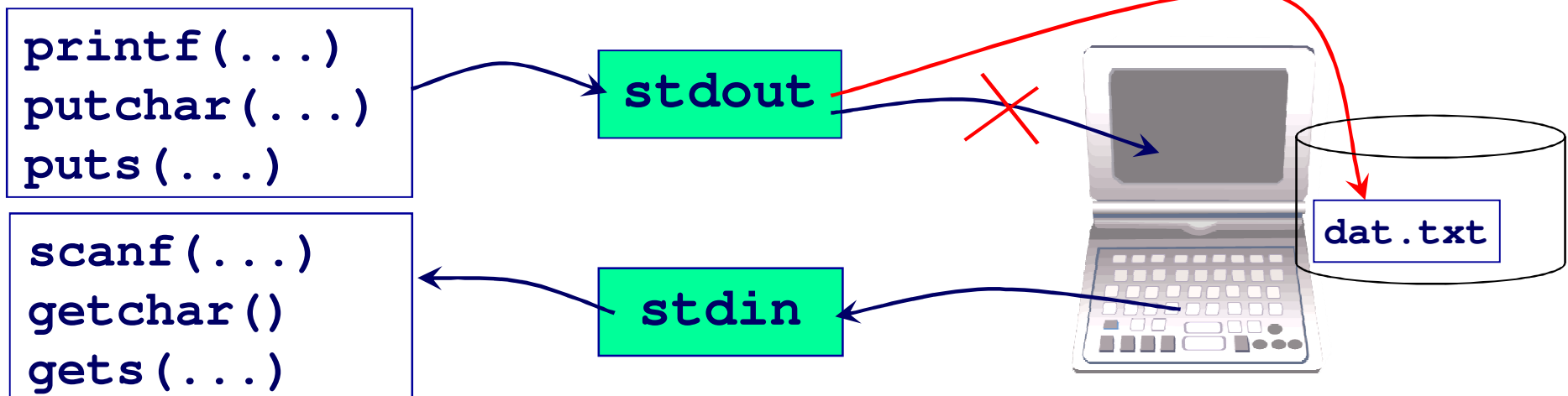
```
c:\pipi> pis.exe
```

Ispis na `stdout` rezultira ispisom na zaslonu

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

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

program `pis.exe`

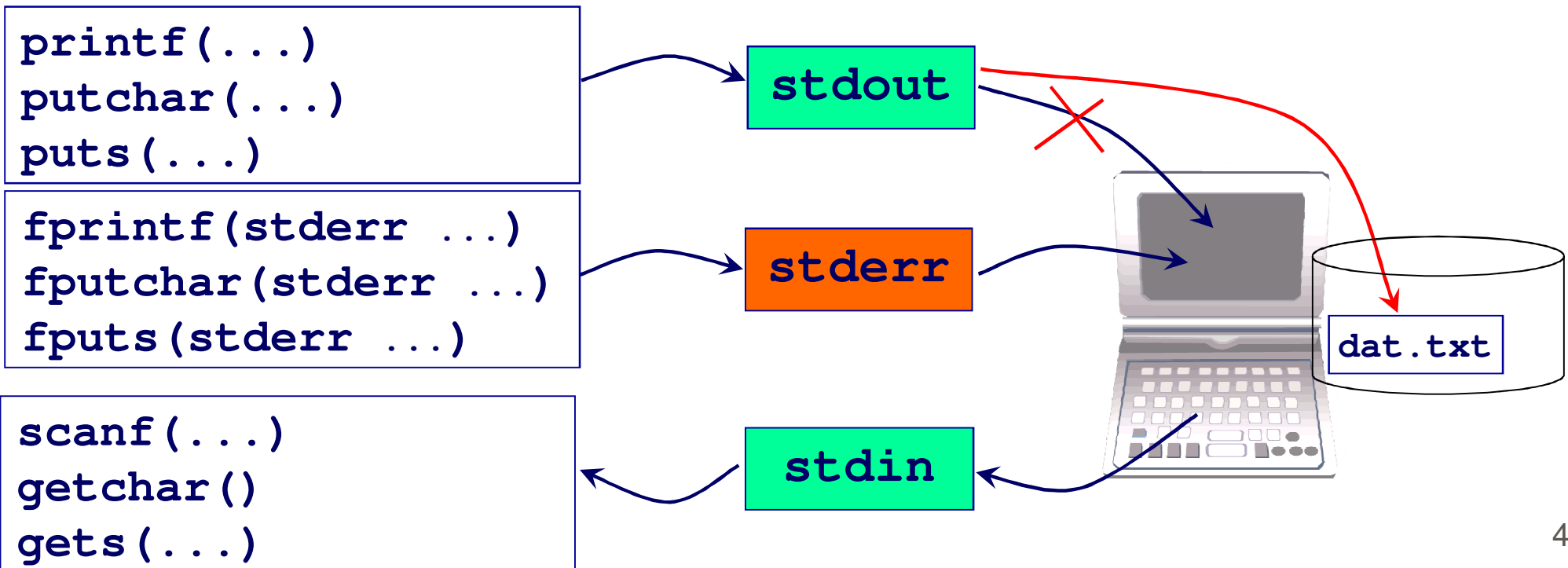


Tok podataka `stderr`

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`



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:

`Prvi`

`Drugi`

`Cetvrti`

`Sesti`

Na zaslону će se ispisati:

`Treci`

`Peti`

Primjer: kome return (i exit) vraćaju rezultate?

```
komeReturnVraca.c
```

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

```
int main() {
```

```
    return -99;
```

```
}
```

Operacijski sustav DOS (NT)

```
c:\pipi>komeReturnVraca.exe
c:\pipi>echo %ERRORLEVEL%
-99
```

Operacijski sustavi se međusobno razlikuju. Npr. Unix:

```
$ komeReturnVraca
$ echo $status
-99
```

Slijedne i direktne datoteke

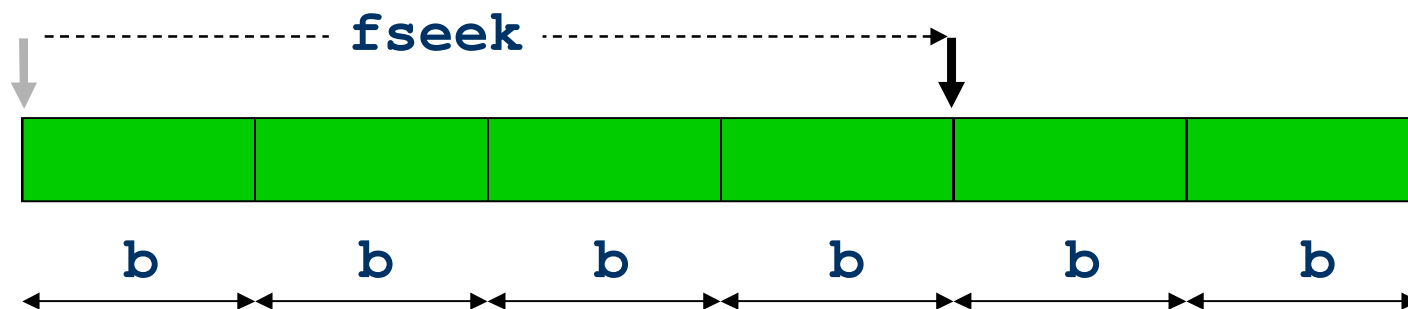
- zahvaljujući funkciji `fseek` moguće je pozicionirati se na bilo koju poziciju (redni broj bajta) unutar datoteke i obaviti operaciju čitanja ili pisanja
- međutim, kako pristupiti n-tom **zapisu** u datoteci?

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:

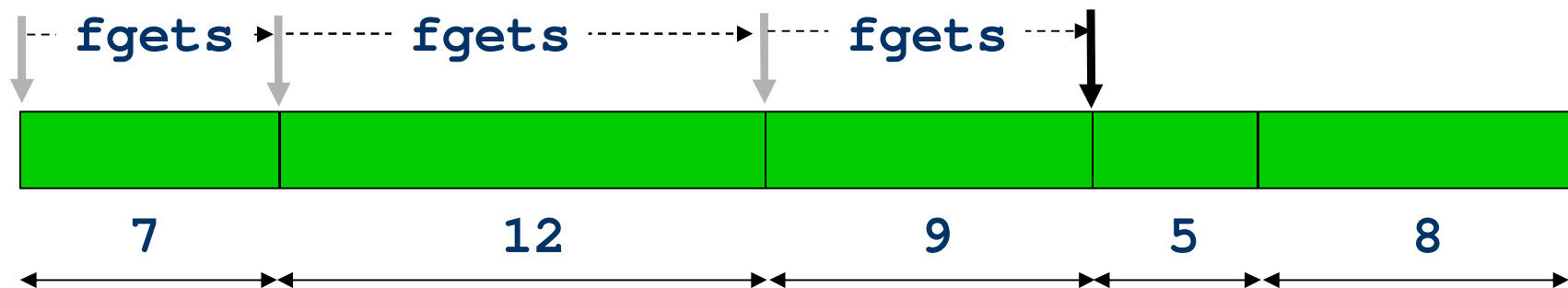
```
fseek (tok, (long) (n-1) * sizeof(struct zapis), SEEK_SET);
```

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(...);
```

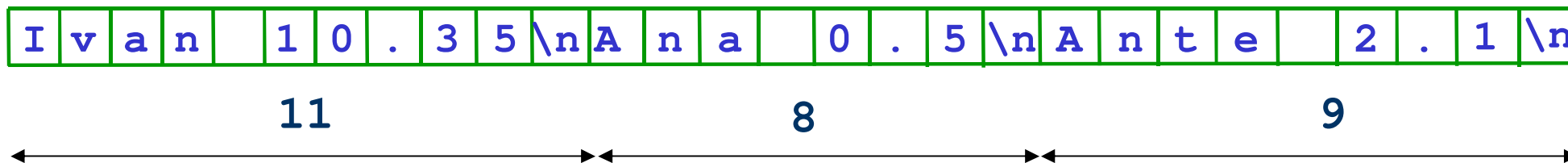
Primjer: pozicioniranje na početak 4. zapisa datoteke



Slijedne datoteke

Primjer: Tekstualna datoteka
načinjena editorom:

```
Ivan 10.35
Ana 0.5
Ante 2.1
```



Do 3. zapisa se može pristupiti isključivo slijedno, npr. ovako:

```
fscanf(tok, "%s%f", ime, &kolicina);
fscanf(tok, "%s%f", ime, &kolicina);
fscanf(tok, "%s%f", ime, &kolicina);
/* sada se u varijablama ime i kolicina
   nalaze vrijednosti 3. zapisa datoteke */
```

Slijedne i direktne datoteke

- Slijedne datoteke: zapisima se pristupa slijedno



- Direktne datoteke: zapisima se pristupa direktno



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)?

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

Primjer: Prijepis slijedne formatirane u direktnu neformatiranu datoteku

Primjer sadržaja datoteke "upis.txt":

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

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

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;
```

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 2010 */
tek_godina = 2009;
```

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;
}
```

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 zaslon
- nakon toga treba uzastopno učitavati s tipkovnice matične brojeve. Za svaki učitani matični broj na zaslon 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

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;
```

Primjer: Korištenje direktne datoteke

```
/* prvo slij. procitati sve zapise i izr. 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);
```

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;
}
```

Rezultat izvođenja programa

Prosječna težina je: 77.82

Unesite matični broj:1

1 Peric Pero

59 76.80

Osoba je lakša od prosjeka.

Unesite matični broj:2

2 Ivic Ivo

54 86.30

Unesite matični broj:6

6 Markovic Marko

66 101.50

Unesite matični broj:11

Zadatak: Ispis računa

U formatiranoj datoteci `kupljeno.txt` upisani su podaci o kupljenim artiklima. Zapis datoteke sadrži šifru artikla (3 znamenke) i broj kupljenih komada tog artikla (2 znamenke):

```
101 12
115 2
```

Zapis direktne neformatirane datoteke `artikli` sadrži šifru artikla (short), naziv artikla (20+1 znak) i cijenu jednog komada artikla (float). Redni broj zapisa u datoteci odgovara šifri artikla. Napisati program koji će na zaslon ispisati račun u sljedećem obliku:

	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>
	<i>1234567890</i>	<i>1234567890</i>	<i>1234567890</i>	<i>1234</i>
Telefon Kanasonic	12	10.00	120.00	kn
CD Player Suny	2	1100.10	2200.20	kn
UKUPNO:			2320.20	kn

Rješenje: Ispis računa

```
#include <stdio.h>

int main () {
    FILE *kup, *art;

    struct {
        short sifArt;
        char nazArt[20+1];
        float cijena;
    } artZapis;

    short sifArt, kolicina;
    float suma = 0;

    kup = fopen("kupljeno.txt", "r");
    art = fopen("artikli", "rb");
```

Rješenje: Ispis računa

```
while (fscanf (kup, "%3hd%2hd", &sifArt, &kolicina)
        == 2) {
    fseek (art, (long) sizeof (artZapis) * (sifArt-1),
           SEEK_SET);
    /* Procitaj cijeli zapis u strukturu */
    fread (&artZapis, sizeof (artZapis), 1, art);
    /* Ispisi redak racuna na zaslon */
    printf ("%s %2d %8.2f %8.2f kn\n",
            artZapis.nazArt, kolicina,
            artZapis.cijena, artZapis.cijena*kolicina);
    suma += artZapis.cijena * kolicina;
}
printf ("UKUPNO:%34.2f kn", suma);
fclose (kup);
fclose (art);
return 0;
}
```

Zadatak: Izdvajanje zapisa

- U postojeću formatiranu datoteku ulaz.txt editorom su upisani podaci o osobama (matični broj i prezime). Prezime nije dulje od 15 znakova. Primjer sadržaja datoteke prikazan je ovdje:

```
952 Medvedec  
101 Vurnek  
205 Habajec  
412 Voras  
551 Ozimec
```

- U novu formatiranu datoteku izlaz.txt prepisati podatke o osobama čije prezime sadrži slovo **r**. Primjer sadržaja izlazne datoteke prikazan je ovdje:

```
Vurnek 101  
Voras 412
```

Rješenje: Izdvajanje zapisa

```
#include <stdio.h>
#include <string.h>
int main () {
    FILE *ulTok, *izTok;
    int mbr;
    char prez[15+1];

    ulTok = fopen ("c:/tmp/ulaz.txt", "r");
    izTok = fopen ("c:/tmp/izlaz.txt", "w");
    while (fscanf(ulTok, "%d%s", &mbr, prez) == 2) {
        if (strchr(prez, 'r') != NULL) {
            fprintf(izTok, "%s %d\n", prez, mbr);
        }
    }
    fclose (ulTok);
    fclose (izTok);
    return 0;
}
```

Zadatak: Izmjena sadržaja datoteke

- U direktnoj neformatiranoj datoteci **bodovi** nalaze se podaci o 10 studenata i bodovima koje su dobili na nekom predmetu. Svaki zapis sadrži matični broj (int), prezime i ime (21+1 znak) i broj bodova (int). Matični brojevi su u rasponu od 1-10, a redni broj zapisa odgovara matičnom broju. Napisati program kojim će se za jednog slučajno odabranog studenta za 10% povećati dotadašnju vrijednost njegovih bodova. Ograničiti uvećani broj bodova na maksimalnih 500 bodova.

Zadatak: Izmjena sadržaja datoteke

rbr. bajta na kojem
"započinje zapis"

0	1	Horvat Ivan	250
30	2	Novak Ana	340
60	3	Juras Ante	480
90	4	Kolar Marija	320
120	5	Ban Darko	490
150	6	Ciglar Ivana	410
180	7	Bohar Marko	290
210	8	Katan Maja	400
240	9	Pobor Janko	345
270	10	Zdilar Mateja	440

Rješenje: Izmjena sadržaja datoteke

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

int main () {
    FILE *dU1Iz1;
    struct {
        int mbr;
        char prezIme[21+1];
        int brBod;
    } zapis;

    int mbr;

    dU1Iz1 = fopen("bodovi", "r+b");
```

Rješenje: Izmjena sadržaja datoteke

```
/* inicijaliziraj generator slučajnih brojeva */
srand((unsigned) time(NULL));
/* odaberi slučajni broj iz intervala [1,10] */
mbr = rand() % 10 + 1;

/* Postavi kazaljku neposredno ISPRED
   odgovarajućeg zapisa */
fseek(dU1Izl, (long)sizeof(zapis)*(mbr-1), SEEK_SET);

/* Procitaj cijeli zapis u strukturu */
fread(&zapis, sizeof(zapis), 1, dU1Izl);

/* Povecaj broj bodova (ali ne vise od 500) */
zapis.brBod *= 1.1;
if (zapis.brBod > 500)
    zapis.brBod = 500;
```

Rješenje: Izmjena sadržaja datoteke

```
/* VAZNO: ne zaboraviti zapisati promijenjene  
podatke natrag u datoteku!!! */
```

```
/* Postavi kazaljku neposredno ISPRED  
odgovarajućeg zapisa jer nakon  
prethodnog citanja, kazaljka je bila  
neposredno IZA odgovarajućeg zapisa */
```

```
fseek(dU1Iz1, -1L*(long)sizeof(zapis), SEEK_CUR);
```

```
/* Zapisi sadržaj cijele strukture u datoteku */  
fwrite(&zapis, sizeof(zapis), 1, dU1Iz1);
```

```
fclose(dU1Iz1);
```

```
return 0;
```

```
}
```