

PiPI

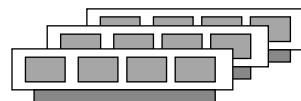
08-Datoteke

zš g01

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



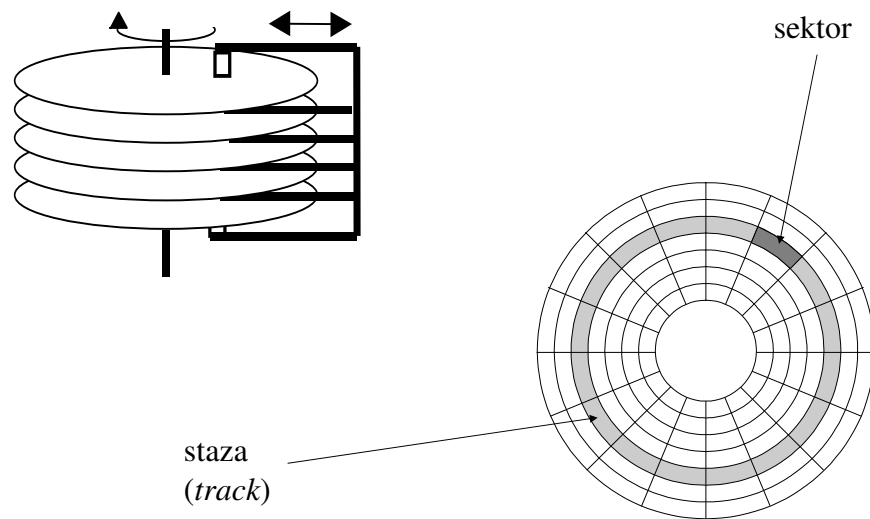
magnetski i optički diskovi



memorijske kartice



Shematski prikaz i fizička organizacija magnetskog diska



3

Logička organizacija magnetskog diska

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

4

OS, direktoriji, datoteke i zapisi

- **Operacijski sustav računala:** program koji povezuje sklopovlje računala s programskom opremom. Između ostalog, vodi evidenciju o fizičkom smještaju direktorija i datoteka na magnetskom disku.
- **Direktorij (imenik, kazalo):** datoteka koja sadrži popis i karakteristike drugih datoteka tvoreći hijerarhijsku strukturu nalik na stablo
- **Datoteka:** imenovani skup podataka na mediju za pohranu, obično sačinjene od zapisa.
- **Zapis:** skup susjednih podataka unutar datoteke koji se obrađuje kao cjelina.

5

Podjela datoteka

Po načinu pristupa

1. slijedne
2. direktne

Po načinu upisa

1. formatirane
2. neformatirane

slijedne



direktne



6

Sadržaj

Otvaranje i zatvaranje datoteke

```
FILE *fopen(const char *filename, const char *mode);  
int fclose(FILE *stream);          exit(int broj);
```

Čitanje i pisanje za (formatirane) datoteke

```
int fgetc(FILE *stream);  
int fscanf(FILE *stream, const char *format [, address, ...]);  
char *fgets(char *s, int n, FILE *stream);  
int fputc(int c, FILE *stream);  
int fprintf(FILE *stream, const char *format, [, address, ...]);  
int fputs(char *s, FILE *stream);
```

Čitanje i pisanje za (neformatirane) datoteke

```
size_t fread(void *ptr, size_t size, size_t n, FILE *stream);  
size_t fwrite(void *ptr, size_t size, size_t n, FILE *stream);
```

Pozicioniranje u datoteci

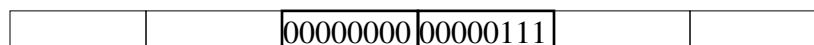
```
int fseek(FILE *stream, long offset, int whence);  
long ftell(FILE *stream);
```

Definiranje i korištenje zapisa/struktura

Neformatirane datoteke

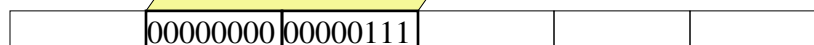
Radna memorija

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



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

Datoteka



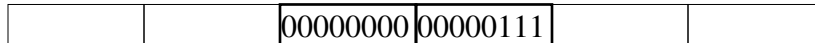
trenutna pozicija

trenutna pozicija

Formatirane datoteke

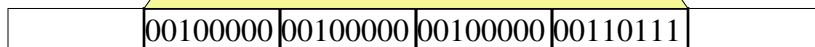
Radna memorija

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



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

Datoteka



početna pozicija

32 (' ')

trenutna pozicija

55 ('7')

9

Funkcija fopen

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

Veza programa s datotekom.

Vraća "pokazivač na datoteku" ili NULL kada datoteke nema.

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

filename ime datoteke

mode način rada

"w" pisanje (ako datoteka ne postoji, stvara se; ako postoji, briše se sadržaj; nije dozvoljeno čitanje)

"a" pisanje (ako datoteka ne postoji, stvara se; ako postoji, podaci se dodaju na kraj; nije dozvoljeno čitanje)

"r" čitanje (ako datoteka ne postoji, vraća NULL pokazivač; nije dozvoljeno pisanje)

"r+" čitanje i pisanje (ako datoteka ne postoji, vraća NULL pokazivač)

"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)

10

Funkcija `fclose`

Prekida vezu programa s datotekom.

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

stream pokazivač na FILE strukturu (datoteku) koja se zatvara

Funkcija vraća **0** ako je uspjelo , **EOF** ako je pogreška

```
void exit (int izlazni_kod);
```

izlazni_kod - 0 ili **EXIT_SUCCESS** ako je uspjelo,
ili **EXIT_FAILURE** za pogrešku

Funkcija zatvara sve datoteke i završava rad programa.

11

Funkcije za čitanje iz formatirane datoteke

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

Funkcija vraća pročitani znak, **EOF** ako je pogreška ili kraj datoteke

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

Funkcija vraća broj pročitanih polja ili **EOF** za pogrešku ili kraj datoteke

```
char *fgets(char *s, int n, FILE *stream);
```

s pokazivač na područje u memoriji gdje će biti smješteni podaci

n maksimalni broj znakova koji se može smjestiti na **s**

stream ukazovateľ na FILE struktúru (datoteku) z ktorej sa číta

Funkcija vraća pokazivač na učitani niz ili **NULL** za pogrešku ili kraj datoteke

12

Primjer: Prijepis datoteke `test.txt` na zaslon a) s pomoću funkcije `fgetc`

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

void main () {
    FILE *d;  int c;
    d = fopen ("test.txt", "r");

    if (d == NULL) {
        printf ("Datoteka se ne moze otvoriti\n");
        exit (1);
    }
    while ((c = fgetc (d)) != EOF) {
        putchar (c);
    }
    fclose (d);
}
```

1.redak
drugi

4.red

6.redak
KRAJ

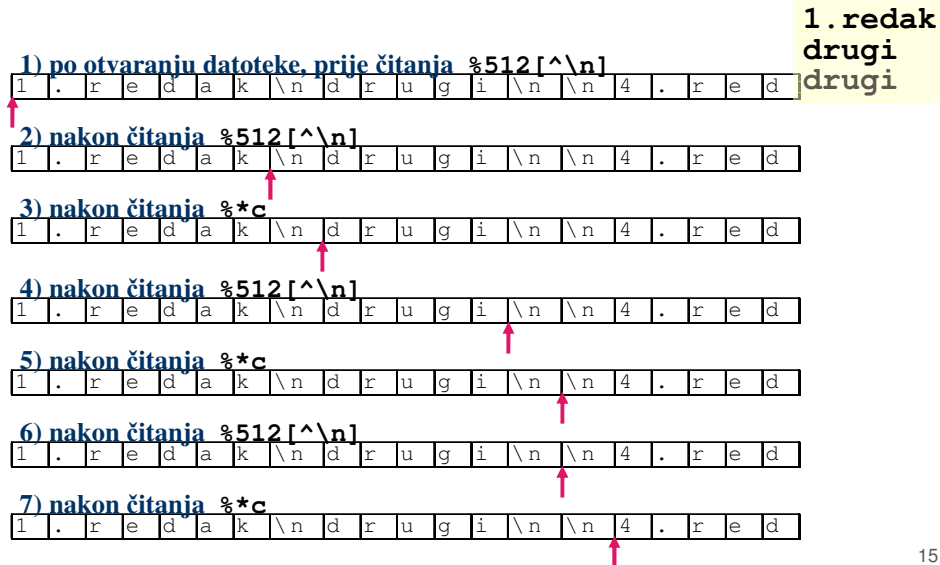
Primjer: Prijepis datoteke `test.txt` na zaslon b) s pomoću funkcije `fscanf`

```
#include <stdio.h>
#include <stdlib.h>
void main () {
    FILE *d;  int c;  char linija[512+1];
    d = fopen ("test.txt", "r");
    if (d == NULL) {
        printf ("Datoteka se ne moze otvoriti\n");
        exit (1);
    }
    while(fscanf(d, "%512[^\n]*c", linija) >= 0) {
        printf ("%s\n", linija);
    }
    fclose (d);
}
```

1.redak
drugi
drugi
4.red
4.red
6.redak
KRAJ

Pomak pozicije u datoteci

```
while (fscanf(d, "%512[^\n]*c", linija) >= 0) {
```



15

Primjer: Prijepis datoteke test.txt na zaslon c) s pomoću funkcije fgets

```
#include <stdio.h>
#include <stdlib.h>
void main () {
    FILE *d; int c; char linija[512+1];
    d = fopen ("test.txt", "r");
    if (d == NULL) {
        printf ("Datoteka se ne moze otvoriti\n");
        exit (1);
    }
    while (fgets (linija, 512, d) >= 0) {
        // fgets ostavlja '\n' i on se može ukloniti jer puts dodaje '\n'
        linija[strlen(linija)-1] = '\0';
        puts (linija);
    }
    fclose (d);
}
```

1. redak
drugi
4. red
6. redak
KRAJ

1. redak
drugi
4. red
6. redak
KRAJ

16

Funkcije za pisanje u formatiranu datoteku

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

Funkcija vraća ispisani znak ili EOF ako je pogreška

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

Funkcija vraća broj ispisanih znakova (byte)

```
int fputs(char *s, FILE *stream);
```

s pokazivač na područje u memoriji gdje su smješteni podaci

stream pokazivač na FILE strukturu (datoteku) u koju se upisuje

Funkcija vraća posljednji ispisani znak ili EOF za pogrešku

17

Primjer: Prijepis datoteke u drugu datoteku a) s pomoću fgetc i fputc

```
#include <stdio.h>  
void main () {  
    FILE *du, *di;  
    int c;  
  
    du = fopen("dat.txt", "r");  
    di = fopen("datc.txt", "w");  
  
    while ((c=fgetc(du))!=EOF)  
        fputc (c, di);  
  
    fclose (du);  fclose (di);  
}
```

- kopiranje neformatirane datoteke:

```
du = fopen("sl.jpg", "rb");  
di = fopen("slc.jpg", "wb");
```

18

Primjer: Prijepis datoteke u drugu datoteku

b) fscanf i fprintf

```
#include <stdio.h>

void main () {
    FILE *du, *di;
    int c; char linija[512+1];

    du = fopen("dat.txt", "r");           ... "rb");
    di = fopen("dat1.txt", "w");          ... "wb");

    while(fscanf(du, "%512[^\n]*c", linija) >=0)
        fprintf (di"%s\n", linija);

    fclose (du);  fclose (di);
}
```

Problem s uzastopnim '\n' – jedno rješenje je brisana linija[0]='\0'
– drugo rješenje "b"

19

Primjer: Prijepis datoteke u drugu datoteku

c) fgets i fputs

```
#include <stdio.h>
#define MAXLIN 512

void main () {
    FILE *du, *di;
    int c; char linija[MAXLIN+1];

    du = fopen("dat.txt", "r");
    di = fopen("dat2.txt", "w");

    while(fgets(linija, MAXLIN, du))
        fputs (linija, di);    // ne dodaje \n

    fclose (du);  fclose (di);
}
```

20

stdin, stdout, stderr i redirekcija

Pokretanjem programa već su otvoreni tokovi podataka ("datoteke") **stdin** (tipkovnica), **stdout** (zaslon) i **stderr** (zaslon, ali bez redirekcije), pa se prijepis iz datoteke na zaslon može načiniti kao u datoteku **stdout**.

Primjer: **ispis.c**

```
#include <stdio.h>
void main () {
    fprintf (stdout, "Na stdout!\n");
    fprintf (stderr, "Na stderr!\n");
}
```

```
C:\>ISPIS
Na stdout!
Na stderr!
C:\>
```

```
C:\>ISPIS >test.dat
Na stderr!
C:\>type test.dat
Na stdout!
```

21

Čitanje neformatirane datoteke: funkcija **fread**

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

ptr adresa u memoriji na koju će se smjestiti učitani podaci
velicina veličina jednog objekta koji će se učitati
n broj objekata koji će se učitati pozivom funkcije
stream pokazivač na FILE strukturu (datoteku) iz koje se čita

Funkcija vraća broj učitanih objekata, za dolazan na kraj datoteke ili pogrešku, vraćena vrijednost je < n

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

Primjeri korištenja funkcije `fread`

```
long v; FILE *du; int n;
du = fopen ("datoteka", "rb");
n = fread (&v, sizeof (v), 1, du)
...
```

```
#define MAXPOLJE 10
int p[MAXPOLJE]; FILE *du; int n;
du = fopen ("datoteka", "r+b");
n = fread (p, sizeof (int), MAXPOLJE, du);
...
```

23

Pisanje u neformatiranu datoteku: funkcija `fwrite`

```
size_t (void *, size_t, size_t, FILE *);

fwrite (ptr, velicina, n, stream);
```

ptr adresa u memoriji na kojoj se nalaze podaci za upisivanje
velicina veličina jednog objekta koji se zapisuje
n broj objekata koji će se zapisati pozivom funkcije
stream pokazivač na FILE strukturu (datoteku) u koju se upisuje
Funkcija vraća broj zapisanih objekata, ako se dogodi pogreška
vraćena vrijednost je < n

```
size_t fwrite (void *ptr, size_t size, size_t n, FILE *stream);

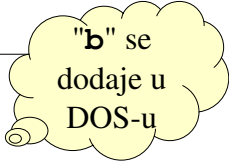
<stdio.h>
```

24

Primjeri korištenja funkcije `fwrite`

```
double mjer; FILE *di; size_t n;
...
di = fopen ("datoteka", "wb");
n = fwrite (&mjer, sizeof (mjer), 1, di);
....
```

```
#define MAXL 80
char p[MAXL]; FILE *di;
di = fopen ("dat1.dat", "w+b");
...
n = fwrite (p, sizeof (char), MAXL, di)
```



"b" se
dodaje u
DOS-u

25

Primjeri korištenja funkcije `fwrite`

```
#include <stdio.h>
void info (int n, int m){
    printf("Problem: zapisano je %d od %d elemenata!\n", n, m);
    exit(-1);
}
void main () {
    char ac[]="Neformatirana datoteka - zapis:";
    long x = 8L, y; double ax[3] = {1.0, -3.5, 3.25};
    size_t n, m, k;
    FILE *idat;
    idat = fopen ("podaci.xyz", "wb");
    n = strlen(ac)+1;
    m = fwrite (ac, 1, n, idat);
    if (m < n) info(m, n);
    k = fwrite (&x, sizeof(x), 1, idat);
    k += fwrite (&y, sizeof(y), 1, idat);
    k += fwrite (ax, sizeof(ax[0]), 3, idat);
    if (k < 5) info(k, 5);
    fclose(idat);
}
```

26

Primjeri korištenja funkcije `fwrite`

- Zapisano u `podaci.xyz`
(heksadecimalni prikaz) :

```
4e 65 66 6f 72 6d 61 74   Neformat
69 72 61 6e 61 20 64 61   irana da
74 6f 74 65 6b 61 20 2d   toteka -
20 7a 61 70 69 73 3a 00   zapis:
08 00 00 00 ?? ?? ?? ??   ....????   8L   <smeće>
00 00 00 00 00 00 f0 3f   .....d?   1.0 (double)
00 00 00 00 00 00 0c c0   .....Ė   -3.5 (double)
00 00 00 00 00 00 0a 40   .....@   3.25 (double)
```

27

Primjeri korištenja funkcije `fread`

```
#include <stdio.h>
void info (int n, int m){
    printf("Problem: zapisano je %d od %d elemenata!\n", n, m);
    exit(-1);
}
void main () {
    char ac[31+1];
    long x, y;      double ax[3];
    size_t m;       FILE *udat;
    udat = fopen ("podaci.xyz", "rb");
    n = 31+1;
    m = fread (ac, 1, n, udat);
    m += fread (&x, sizeof(x), 1, udat);
    m += fread (&y, sizeof(y), 1, udat);
    m += fread (ax, sizeof(ax[0]), 3, udat);
    if (m < 3) info(m, 3);
    fclose(udat);
    printf("%s %dL %dL ", ac, x, y);
    printf("\n%f %f %f", ax[0], ax[1], ax[2]);
}
```

Zapisi (strukture)

- Strukture podataka čiji se elementi razlikuju po tipu.

```
struct naziv_strukture {  
    tip_a ime_elementa_1;  
    tip_b ime_elementa_2;  
    ...  
    tip_a ime_elementa_n;  
};
```

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

- Ovime nije definiran konkretan zapis, već je samo opisana struktura zapisa (**deklaracija**).

29

Definicija konkretnih zapisa (strukture)

```
struct naziv_strukture zapis1, zapis2, ... , zapisN;
```

npr.

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

- Moguće je opisati strukturu zapisa i definirati konkretan zapis zajedno:

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

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

- Za opis zapisa/strukture vrijede sva pravila pisanja C koda

30

Deklaracija zapisa (strukture) bez naziva i s typedef

- Naziv strukture može se izostaviti ako se takva struktura drugdje ne koristi:

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

ili

```
struct {  
    int dan, mjesec, godina;  
} datum1, datum2;
```

varijabla
zapisa

- Deklaracija strukture često se koristi s typedef

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

ili

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

tip zapisa

31

Vrijednosti elemenata zapisa (strukture)

```
zapis.element = vrijednost;  
vrijednost = zapis.element;
```

Npr.:

```
strcpy (os1.jmbg, "0101970330513");  
scanf ("%s %s %d", os1.prezime, os1.ime, &os1.visina);  
os1.tezina = 75.5;  
t1.x = 7; t1.y = 2;  
t2.x = 5; t2.y = 3;  
udaljenost = sqrt(pow((double) (t1.x - t2.x), 2.) +  
                  pow((double) (t1.y - t2.y), 2.));  
printf ("Datum = %02d.%02d.%d\n", datum.dan,  
        datum.mjesec, datum.godina);
```

32

Složeni zapisi (strukture)

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

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

struct student grupa01[78], neki;
grupa01[67].osobni_podaci.ime = "Pavo";
neki.osobni_podaci.ime = "Maja";
```

Odabir imena za tip i varijablu vrlo je važan za preglednost programa!

Korištenjem `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 osobni_podaci;
    osoba otac;
    osoba majka;
    adresa adresa_roditelja;
    adresa adresa_u_Zagrebu;
} student;

student stprvi;

stprvi.otac.tezina = 92.5;
```

Pokazivač na strukturu

- Definicija pokazivača

```
struct osoba *p, osprva, osdruga;
```

`*p` - zapis o osobi
`p` - adresa zapisa o osobi

- Referenciranje na element strukture preko pokazivača

```
(*p).ime = "Ivica" ili
p->ime = "Marica"
```

P4S.c

34

Pokazivač na strukturu i funkcije

<pre>struct x { int a; int b; int c; }; void funkcija(struct x); main() { struct x z; z.a = 10; // 1 z.a++; funkcija(z); // 2 } void funkcija(struct x z) printf(" prvi clan "); printf("%d \n", z.a); }</pre>	<pre>struct x {int a; int b; int c;}; void funkcija(struct x *); main() { struct x z, *pz; // 3 pz = &z; // 4 z.a = 10; z.a++; funkcija(pz); // 5 } // 6 void funkcija(struct x * pz) { printf(" prvi clan "); printf("%d \n", (*pz).a); } // ili pz->a</pre>
---	--

Pokazivač na strukturu i funkcije s typedef

<pre>typedef struct { int a; int b; int c; } x; void funkcija(x); main() { x z; z.a = 10; z.a++; funkcija(z); } void funkcija(x z) printf(" prvi clan "); printf("%d \n", z.a); }</pre>	<pre>typedef struct { int a; int b; int c; } x; void funkcija(x *); main() { x z, *pz; pz = &z; z.a = 10; z.a++; funkcija(pz); } void funkcija(x * pz) { printf(" prvi clan "); printf("%d \n", pz->.a); }</pre>
--	--

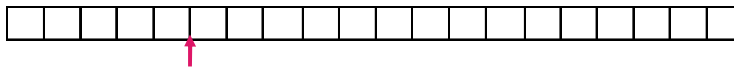
Čitanje i pisanje zapisa/strukture funkcijama **fread i fwrite**

Neformatirane datoteke obično se sastoje od zapisa definiranih strukturom.

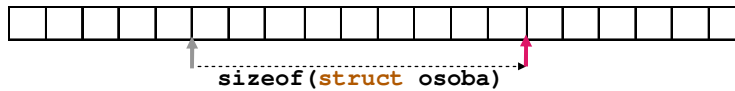
```
m = fwrite (&o1, sizeof(o1), 1, d);  
m = fread (&o2, sizeof(struct osoba), 1, d);
```

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

Prije čitanja ili pisanja



Nakon čitanja ili pisanja



37

Promjena pozicije u datoteci - funkcija **fseek**

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

offset pomak u byte

otkuda **SEEK_SET** – od početka
 SEEK_CUR – od trenutne pozicije
 SEEK_END – od kraja datoteke

stream pokazivač na FILE strukturu (datoteku) u
 u kojoj se mijenja pozicija

Funkcija vraća 0 ako je uspjelo, <> 0 ako je pogreška.

```
int fseek(FILE *stream, long offset, int whence);
```

38

Promjena pozicije u datoteci - funkcija `fseek`



Pozicioniranje na **n**-ti byte:

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

Pomak **unatrag** za **m** byte:

```
fseek(d, -(long) m, SEEK_CUR);
```

Pozicioniranje na **kraj** datoteke:

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

Pozicioniranje na **početak** datoteke:

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

39

Trenutna pozicija u datoteci - funkcija `ftell`

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

```
long ftell(FILE *stream);
```

Funkcija vraća trenutnu poziciju u datoteci u *byte* ili `-1` u slučaju pogreške.

Primjer: Ispisati trenutnu veličinu datoteke.

```
...
```

```
FILE *d = fopen ("neka.dat", "r");
```

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

```
printf ("Velicina datoteke: %ld byte\n", ftell(d));
```

```
...
```

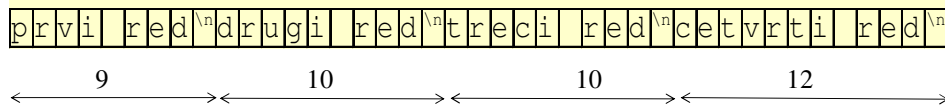
40

Slijedne i direktne datoteke

Zahvaljujući funkciji `fseek`, za razliku od nekih drugih programskih jezika svaka datoteka, ovisno o organizaciji podataka, može biti direktna. Tekstualna datoteka načinjena editorom:

```
prvi red
drugi red
treći red
četvrti red
peti red
```

u datoteci izgleda ovako:

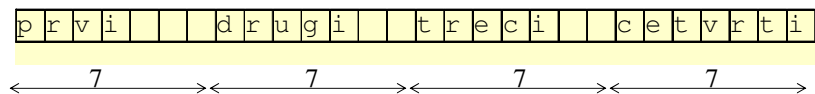


Do n -tog reda (zapisa) može se doći samo slijedno, iako postoji mehanizam pozicioniranja bilo gdje unutar datoteke.

41

Slijedne i direktne datoteke

Kada su zapisi jednake duljine, npr.



do n -tog zapisa može se doći pozicioniranjem na *byte*

$(n-1) * \text{duljina_zapisa}$

Ako je zapis u datoteci definiran strukturom npr. naziva `red`, pozicioniranje na n -ti zapis načinit će se pozivom funkcije

```
fseek(d, (long) (n-1) * sizeof(struct red), SEEK_SET);
```

42

Primjer: Prijepis slijedne formatirane u direktnu neformatiranu datoteku

Svaki zapis formatirane datoteke "upis.txt" sadrži:

- matični broj (4 znamenke),
- prezime i ime (40 znakova),
- godina rođenja (4 znamenke) i
- tjelesna težina (format xxx.x).

Zapise neformatirane datoteke "tezine.dat" čine:

- matični broj,
- prezimena i imena,
- godine starosti i težine,

Matični broj odgovara rednom broju zapisa.

Zadatak:

1. Iz datoteke "upis.txt" treba formirati datoteku "tezine.dat".
2. Izračunati prosječnu tjelesnu težinu.
3. Učitavati s tipkovnice matični broj i za njega iz datoteke "tezine.dat" pročitati i ispisati na zaslon godinu starosti i težinu uz napomenu ako je tjelesna težina manja od prosječne.
4. Program treba završiti kad se zada neispravan ili nepostojeći matični broj.

43

Primjer: Prijepis slijedne formatirane u direktnu neformatiranu datoteku (S2D.c)

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

void fatal (char *poruka) {
    fputs (poruka, stderr); fputs ("\n", stderr);
    exit (1);
}

void main (void) {
    FILE *du, *di;
    struct zapis_osobe {
        short mat_br;
        char prez_ime[40+1];
        short starost;
        float tezina;
    } zapis;
```

44

Primjer: Prijepis slijedne formatirane u direktnu neformatiranu datoteku

```
...

int n, godina, god_rod, mat_br;
float prosjek;

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

if ((di = fopen ("tezine.dat", "w+b")) == NULL)
    fatal ("Ne mogu otvoriti datoteku \"tezine.dat\");

godina = uzmi_godinu ();
prosjek = 0; n = 0;

...
```

45

Primjer: Prijepis slijedne formatirane u direktnu neformatiranu datoteku

```
/*  Ulazni podaci (datoteka "upis.txt") su oblika:
      1           2           3           4           5
123456789012345678901234567890123456789012345678901234567890123
0001Peric Pero                                1947 76.8
0002Ivic Ivo                                  1952 86.3
0003Anic Ana                                  1968 56.7
0006Markovic Marko                            1940101.5
0012Petrovic Petar                            1972 67.8
*/
```

46

Primjer: Prijepis slijedne formatirane u direktnu neformatiranu datoteku

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

47

Primjer: Prijepis slijedne formatirane u direktnu neformatiranu datoteku

sizeof (zapis) = (2+41+2+4) = 49 byte
datoteka tezine.dat ima velicinu 588 *byte* (12*49=588):

C:\>dir tezine.dat

tezine.dat 588 05-01-06 6:19p

0	0001	Peric Pero	48	76.8
49	0002	Ivic Ivo	43	86.3
98	0003	Anic Ana	27	56.7
147	?	?	?	?
196	?	?	?	?
245	0006	Markovic Marko	55	101.5
294	?	?	?	?
343	?	?	?	?
392	?	?	?	?
441	?	?	?	?
490	?	?	?	?
539	0012	Petrovic Petar	23	67.8

48

Primjer: Prijepis slijedne formatirane u direktnu neformatiranu datoteku

```
...

if (n > 0) {
    prosjek /= n;
    printf ("Prosjecna tezina: %5.2f\n", prosjek);
} else {
    fprintf (stderr, "Datoteka \"upis.txt\" je prazna!\n");
    exit (1);
}

...
```

49

Primjer: Prijepis slijedne formatirane u direktnu neformatiranu datoteku

```
...
while (1)
{
    printf ("\nUnesite maticni broj:");
    scanf ("%d", &mat_br);
    if (fseek (di, (long) (mat_br-1) * sizeof (zapis),
        SEEK_SET) != 0)
        fatal("Nije uspjelo pozicioniranje u \"tezine.dat\"");
    if (fread (&zapis, sizeof (zapis), 1, di) != 1 ||
        zapis.mat_br != mat_br) break;
    printf ("%04d %s %4d %5.1f\n", zapis.mat_br,
        zapis.prez_ime, zapis.starost, zapis.tezina);
    if (zapis.tezina < prosjek)
        printf ("Osoba ima tezinu manju od prosjecne!\n");
}
fclose (di);
}
```

50

Rezultat izvođenja programa

Prosječna težina: 77.82

Unesite matični broj:1

0001 Peric Pero 48 76.8

Osoba ima težinu manju od prosječne!

Unesite matični broj:2

0002 Ivic Ivo 43 86.3

Unesite matični broj:6

0006 Markovic Marko 55 101.5

Unesite matični broj:11

51

Dohvat tekuće godine iz sistemskog datuma a) prenosivo rješenje (DOS+Unix)

```
int uzmi_godinu (void) {  
    time_t sekunde;  
    /* funkcija time vraca na mjesto koje pokazuje  
       sekunde broj sekundi od 00:00:00 01. siječnja 1970 */  
    sekunde = time(NULL);  
    /* funkcija ctime pretvara broj sekundi od  
       00:00:00 01. siječnja 1970 u oblik  
       Fri May 19 19:05:27 1995 */  
    return atoi ( ctime(&sekunde) + 20 );  
}
```

Funkcija `atoi` se nalazi u standardnoj biblioteci `<stdlib.h>`

Prototip funkcije i *header* biblioteke moraju biti na početku datoteke (programa):

```
#include <time.h>  
int uzmi_godinu (void);
```

52

Rezervacija i oslobađanje memorije: funkcije **malloc i free**

Definicijom polja uvijek se u memoriji rezervira prostor za najveći očekivani broj članova polja bez obzira na stvarnu potrebu, npr.

```
int polje [1000]
```

Racionalniji pristup je da se rezervira točno potrebna količina memorije.

```
void *malloc (size_t broj);
```

Funkcija **malloc** rezervira kontinuirani broj bajtova (blok) u memoriji računala i vraća pokazivač na taj blok. Kada blok tražene veličine nije moguće rezervirati, funkcija vraća **NULL** pokazivač.

```
void free (void *komad);
```

Funkcija **free** oslobađa blok memorije na koji pokazuje pokazivač **komad**.

Pokazivač **komad** smije biti samo jedan od pokazivača nastalih prethodnim pozivima funkcije **malloc**.

▪ Primjer:

```
#include <malloc.h>
```

```
...
```

```
int *polje;
```

```
size_t n;
```

```
... // unese se n
```

```
polje=(int *)malloc(sizeof(int)*n);
```

```
... // puni se i koristi
```

```
... // polje sa n int elemenata
```

```
free (polje);
```

```
...
```