

Programiranje i programsko inženjerstvo

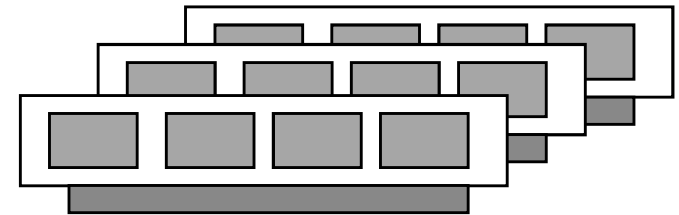
Predavanja
2015. / 2016.

10. Datoteke

Memorija računala

1) privremena (unutarnja)

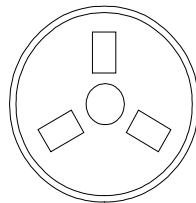
RAM (Random Access Memory)



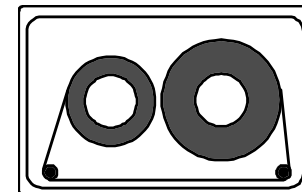
2) stalna (vanjska)

a) sa slijednim pristupom podacima, npr.

magnetska traka



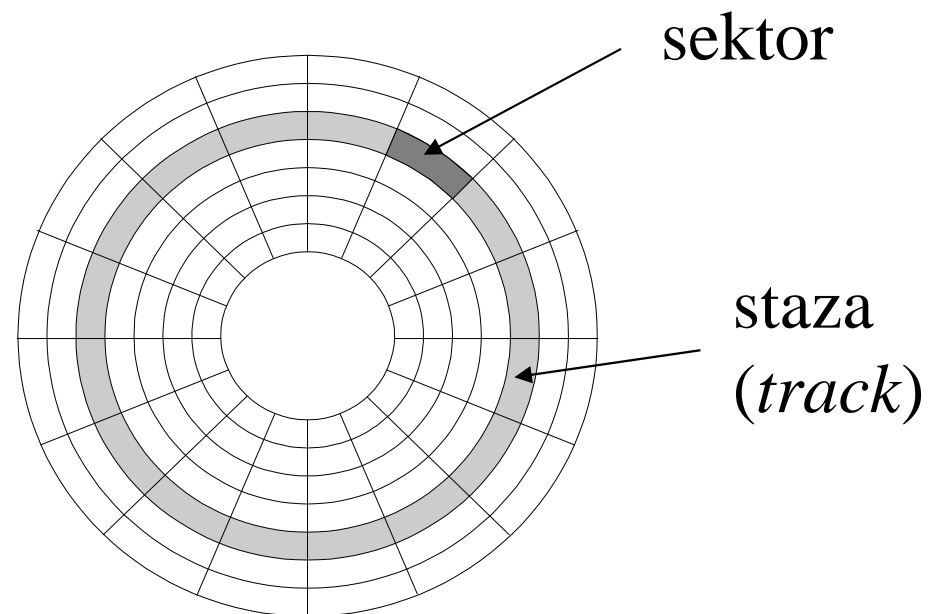
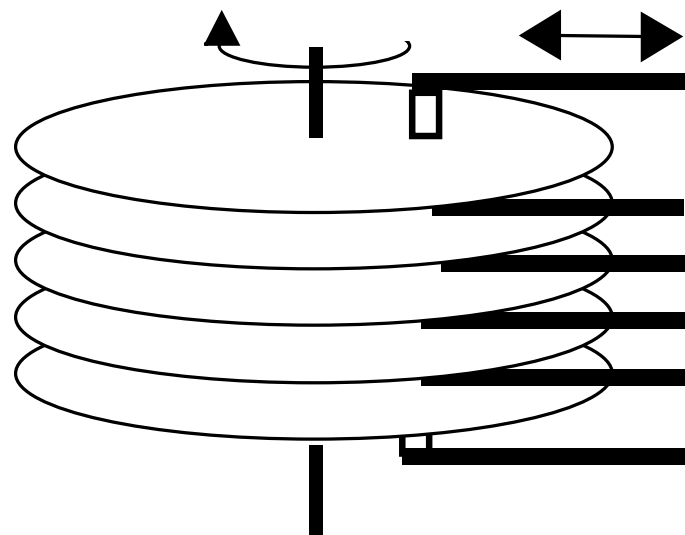
streamer traka



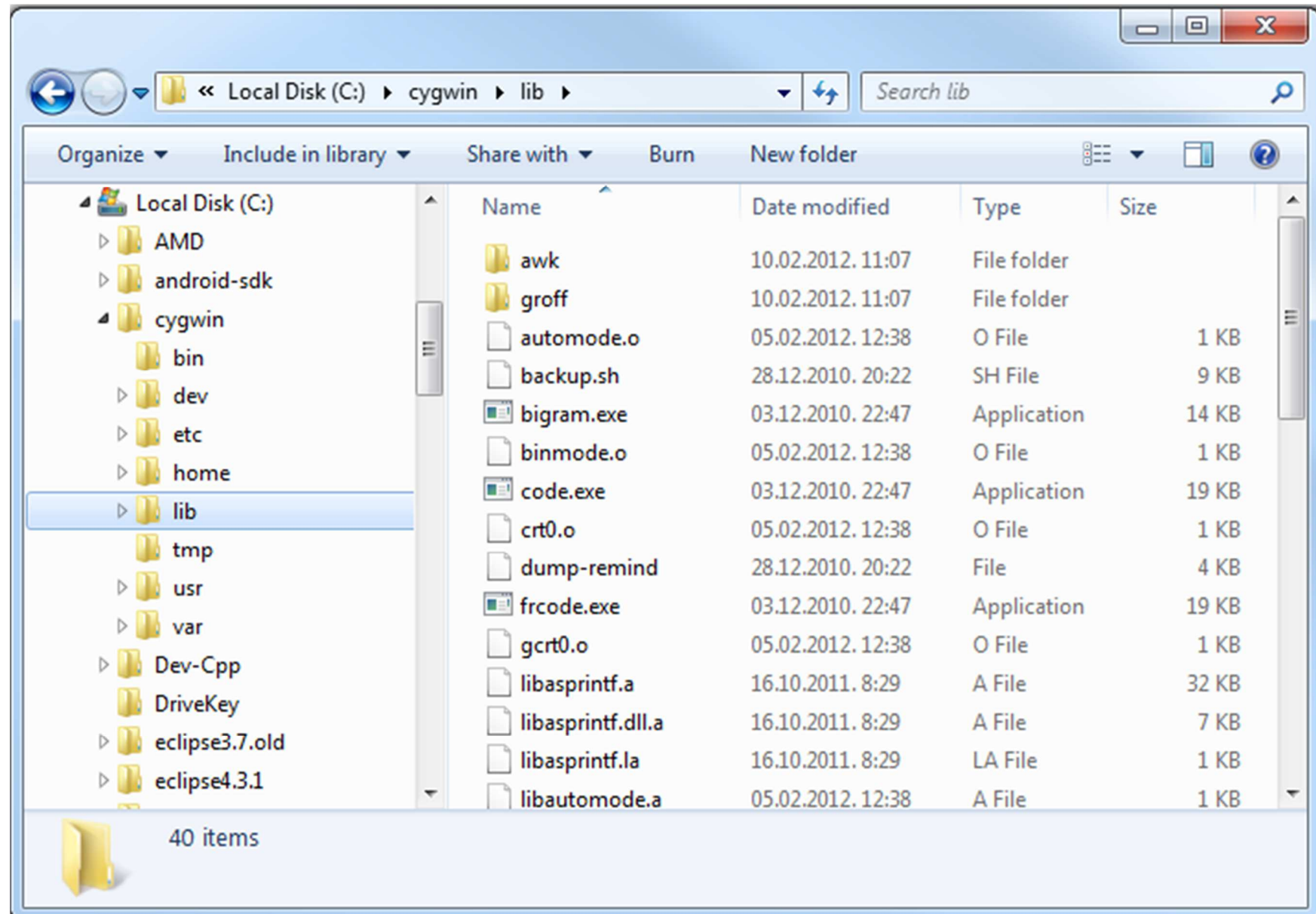
b) s direktnim pristupom podacima, npr.

disketa, magnetski disk, SSD (*solid state drive*)

Fizički prikaz magnetskog diska



Logička organizacija diska



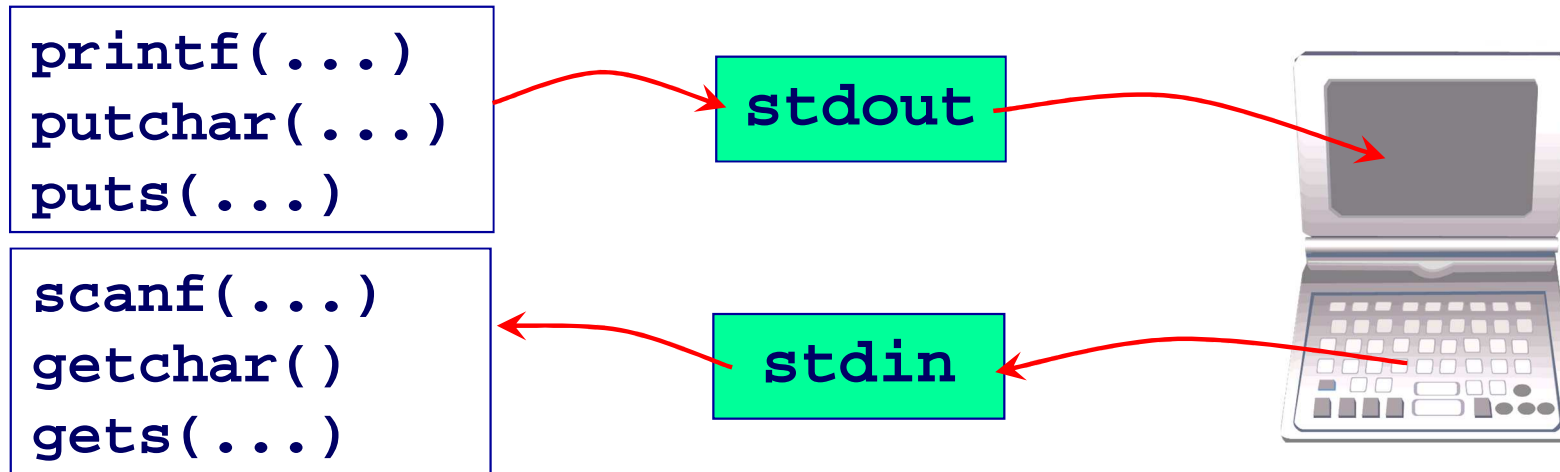
Direktoriji i datoteke

- **Datoteka:** imenovani skup podataka koji sačinjavaju logičku cjelinu, pohranjen na mediju za pohranu (traka, disk, disketa, CD, *memory stick*, ...)
- **Direktorij (imenik, kazalo, mapa):** 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

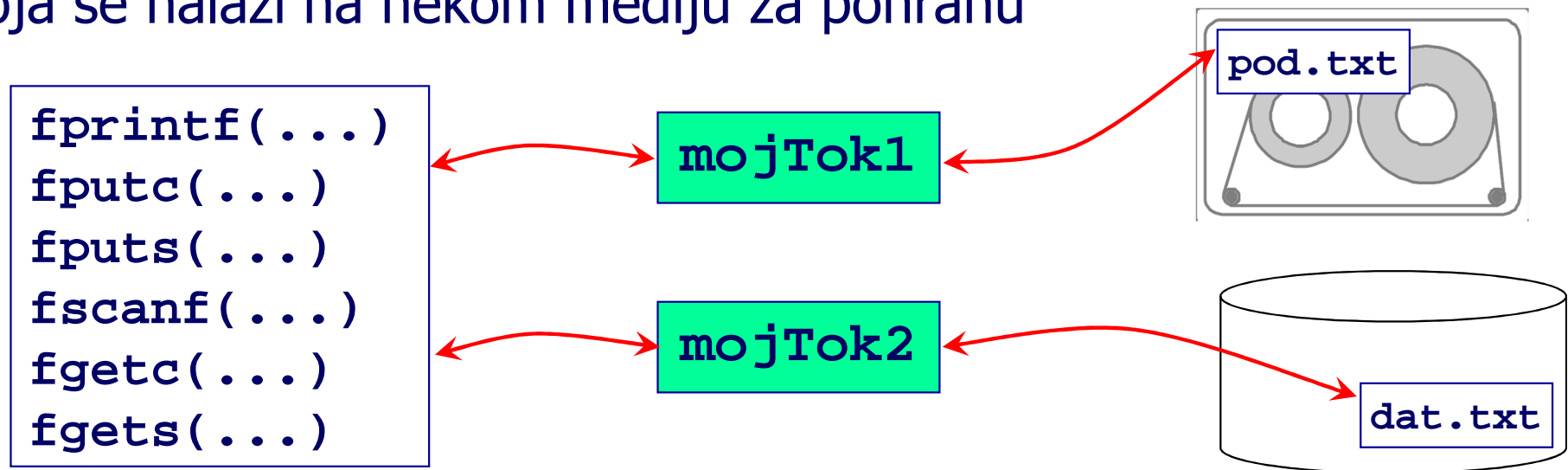
Tok podataka (*stream*)

- Tok podataka (*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 i ispisivati na zaslon (*stdout*) jedan po jedan znak.

sadržaj datoteke `tekst.txt`

Funkcije `scanf`, `getchar` i `gets` uvijek citaju iz toka podataka koji se naziva `stdin`.

A funkcije `printf`, `putchar` i `puts`?

ispis na zaslon

Funkcije `scanf`, `getchar` i `gets` uvijek citaju iz toka podataka koji se naziva `stdin`.

A funkcije `printf`, `putchar` i `puts`?

Rješenje

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

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

    if (tokPod == NULL) {
        printf("Pogreska: ne mogu otvoriti tekst.txt\n");
        exit(1);
    }

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

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

```
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+");
```

- Windows

```
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 binarnih 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 pohrane podataka

- Tekstualne ili formatirane datoteke (eng. *text files*)
- Binarne ili neformatirane datoteke (eng. *binary files*)

Tekstualne (formatirane) datoteke

Središnja memorija

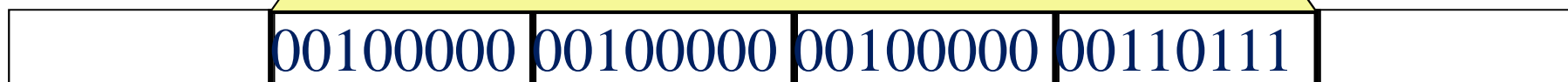
```
short int i=7;
```



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

Datoteka

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



početna
pozicija



nova pozicija



Binarne (neformatirane) datoteke

Središnja memorija

`short int i=7;`



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

Datoteka



početna
pozicija



nova pozicija



Funkcije za čitanje iz tekstualne 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 tekstualne 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 učitanoz 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

Funkcije za pisanje u tekstualnu 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'`

Primjer

- Napisati program koji će sadržaj postojeće tekstualne datoteke `stara.txt` prepisati u novu tekstualnu datoteku `nova.txt`. Za čitanje i pisanje koristiti funkcije `fgetc` i `fputc`.

Rješenje

```
#include <stdio.h>
#include <stdlib.h>
int main(void) {
    FILE *du, *di;
    int c;
    du = fopen ("stara.txt", "r");
    if (du == NULL) {
        printf ("Ne mogu otvoriti datoteku stara.txt\n");
        exit (1);
    }
    di = fopen ("nova.txt", "w");
    if (di == NULL) {
        printf ("Ne mogu otvoriti datoteku nova.txt\n");
        exit (1);
    }
    while ((c = fgetc (du)) != EOF)
        fputc (c, di);
    fclose (du);
    fclose (di);
    return 0;
}
```

Primjer

- Napisati program koji će sadržaj postojeće tekstualne datoteke `stara.txt` prepisati u novu tekstualnu datoteku `nova.txt`. Za čitanje i pisanje koristiti funkcije `fgets` i `fputs`. Svakim pozivom funkcije `fgets` iz datoteke učitati najviše 20 znakova.

Rješenje

```
#include <stdio.h>
#include <stdlib.h>
#define MAXZNAKOVA 20
int main(void) {
    FILE *du, *di;
    char redak[MAXZNAKOVA+1];
    du = fopen ("stara.txt", "r");
    /* dodati provjeru uspjesnosti otvaranja */
    di = fopen ("nova.txt", "w");
    /* dodati provjeru uspjesnosti otvaranja */

    while (fgets(redak, MAXZNAKOVA+1, du) != NULL ) {
        fputs (redak, di);
    }

    fclose (du);
    fclose (di);
    return 0;
}
```

Funkcija za pisanje u binarnu 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

- Napisati odsječak programa koji će u binarnu datoteku `podaci.bin` upisati sadržaj varijable `m` i polja `x`. Varijabla `m` i polje `x` su definirani na sljedeći način:

```
int m = 10;
```

```
double x[3] = {1.5, -3.5, 3.25};
```

- ako zapisivanje varijable `m` ne uspije, na zaslon ispisati poruku s pogreškom i prekinuti program
- ako zapisivanje svih članova polja `x` ne uspije, ispisati koliko se članova polja uspjelo upisati u datoteku i prekinuti program

Rješenje

```
...
int n1, n2; FILE *izTok;
izTok = fopen ("podaci.bin", "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 %d clanova\n", n2);
    exit(2);
}
fclose(izTok);
...
```

radi binarne datoteke

Sadržaj datoteke `podaci.bin` iz prethodnog primjera:

10 → 00 00 00 0A₁₆
1.5 → 3F F8 00 00 00 00 00 00₁₆
-3.5 → C0 0C 00 00 00 00 00 00₁₆
3.25 → 40 0A 00 00 00 00 00 00₁₆

Sadržaj datoteke (u heksadekadskom prikazu):

10 1.5 -3.5

00 00 00 0A 3F F8 00 00 00 00 00 00 00 C0 0C

00 00 00 00 00 00 40 0A 00 00 00 00 00 00 00

3.25

Funkcija za čitanje iz binarne 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

- Napisati odsječak programa koji će iz binarne datoteke `podaci.bin`, koja je stvorena programom prikazanim u prethodnom primjeru, pročitati sadržaj varijabli `k` i polja `y`. Varijabla `k` i polje `y` su definirani na sljedeći način:

```
int k;  
double y[3];
```

- ako čitanje varijable `m` ne uspije, na zaslon ispisati poruku s pogreškom i prekinuti program
- ako čitanje svih članova polja `x` ne uspije, ispisati koliko se članova polja uspjelo pročitati iz datoteke i prekinuti program

Rješenje

```
...
int n1, n2;
FILE *ulTok;
ulTok = fopen ("podaci.bin", "rb");
n1 = fread (&k, sizeof(k), 1, ulTok);
if (n1 < 1) {
    printf("Citanje k nije uspjelo\n");
    exit(5);
}
n2 = fread (y, sizeof(y[0]), 3, ulTok);
if (n2 < 3) {
    printf("Ucitano je %d clanova\n", n2);
    exit(7);
}
fclose(ulTok);
```

Primjer

- Sadržaj postojeće tekstualne datoteke `rez.txt` prepisati u novu binarnu datoteku `rez.bin`.

Primjer sadržaja datoteke
`rez.txt` promatran editorom:

```
Iva Pek 156  
Ante Horvat 12
```

- ime, odnosno prezime, nije dulje od 8 znakova, a broj bodova je cijeli broj manji od 100 000.
- svaki zapis datoteke `rez.bin` sadrži niz znakova *ime* (8+1 znak), niz znakova *prezime* (8+1 znak) i broj bodova (int).

Primjer (komentar)

```
rez.txt  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
```


Rješenje

```
...  
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 mogu razlikovati 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;  
};
```

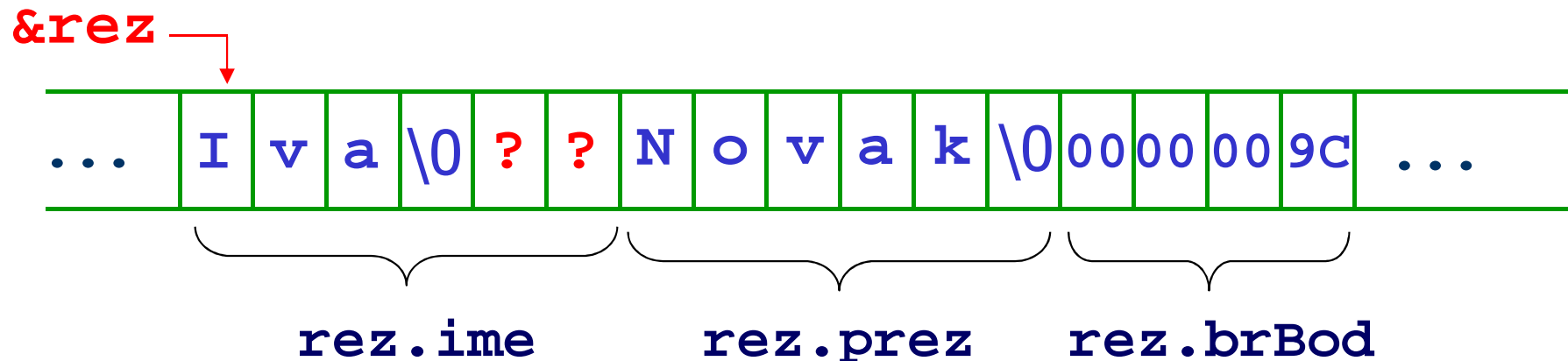
```
struct osoba o1, o2;  
struct osoba o3;
```

```
o3.visina = 178;
```

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

Dodatak uz predavanja o strukturama

- Varijabla tipa strukture jest L-value.
- Kada se struktura koristi kao argument pri pozivu funkcije, prenosi se vrijednost (tj. "kopija") strukture.
- Rezultat funkcije može biti struktura (tj. tip funkcije može biti struktura).

Inicijalizacija strukture

- Slično kao varijable drugih tipova, varijable tipa strukture mogu se inicijalizirati kod definicije:

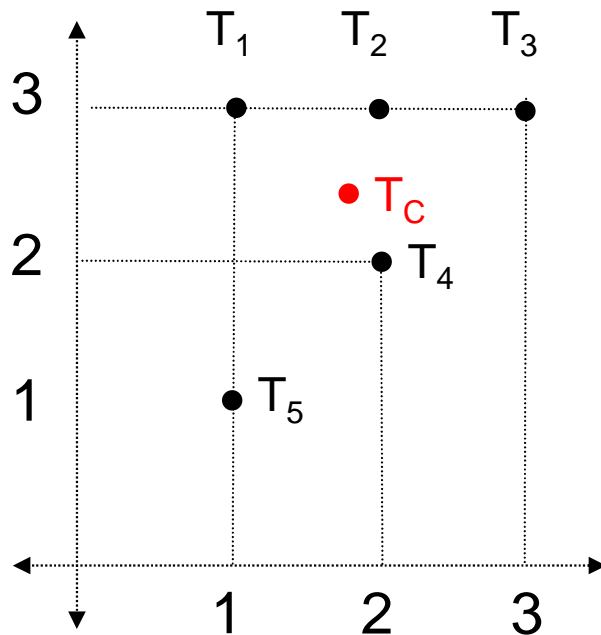
```
#include <stdio.h>
int main(void) {
    struct osoba_s {
        char ime[100+1];
        int visina;
        float tezina;
    };
    struct osoba_s o1 = {
        "Ivo Car",
        180,
        80.f
    };
    printf("%s %d %5.2f", o1.ime, o1.visina, o1.tezina);
    return 0;
}
```

Ivo Car 180 80.00

Primjer

- Napisati funkciju `centroid` koja izračunava *centroid* T_c za zadani skup točaka T_1, T_2, \dots, T_n .

$$x_c = \frac{\sum_{i=1}^n x_i}{n} \quad y_c = \frac{\sum_{i=1}^n y_i}{n}$$



- Točku treba predstaviti strukturom:

```
struct tocka_s {  
    double x;  
    double y;  
};
```

- Skup točaka treba predstaviti poljem struktura:

```
struct tocka_s skupTočaka[5];
```

- U glavnom programu inicijalizirati polje točkama (koristiti podatke sa slike), pozvati funkciju za izračunavanje centroida i ispisati njegove koordinate.

Rješenje

- 1. dio -

Dodatak uz predavanja
o strukturama

centroid.h

```
/* deklaracija strukture tocka_s */
struct tocka_s {
    double x;
    double y;
};

/* prototip funkcije centroid */
struct tocka_s centroid(struct tocka_s *skupTocaka, int n);
```




rezultat funkcije može
biti struktura

Rješenje

- 2. dio -

centroid.c

```
#include "centroid.h"
struct tocka_s centroid (struct tocka_s *skupTocaka
                        , int n) {
    int i;
    struct tocka_s centroid = {0, 0};
    for (i = 0; i < n; ++i) {
        centroid.x += skupTocaka[i].x;
        centroid.y += skupTocaka[i].y;
    }
    centroid.x /= n;
    centroid.y /= n;
    return centroid;
}
```

 inicijalizator strukture

Rješenje

- 3. dio -

Dodatak uz predavanja
o strukturama

glavni.c

```
#include <stdio.h>
#include "centroid.h"
int main(void) {
    struct tocka_s skupNekihTocaka[5] =
        { {1, 3}
          , {2, 3}
          , {3, 3}
          , {2, 2}
          , {1, 1} };
    struct tocka_s rezultat;
    rezultat = centroid(skupNekihTocaka, 5);
    printf("\nCentroid skupa tocaka je: (%.3f, %.3f)"
           , rezultat.x, rezultat.y);
    return 0;
}
```

→ varijabla rezultat jest L-Value

Rješenje

- 4. dio: alternativno, definirati tip -

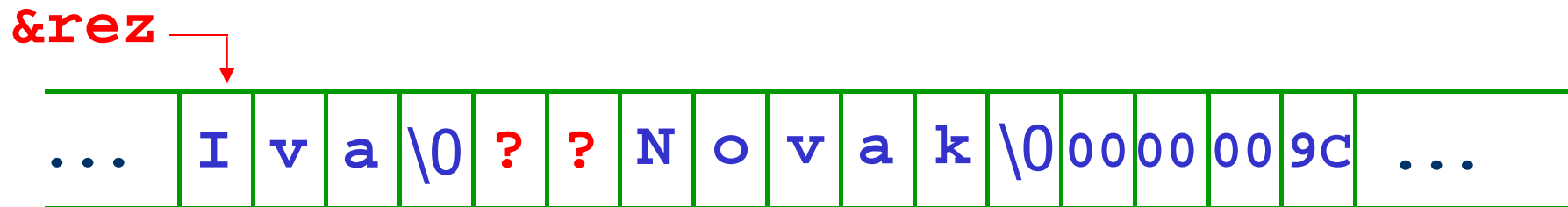
centroid.h

```
/* definicija tipa tocka_t */  
typedef struct {  
    double x;  
    double y;  
} tocka_t;  
  
/* prototip funkcije centroid */  
tocka_t centroid(tocka_t *skupTocaka, int n);
```

itd.

Korištenje struktura za čitanje i pisanje zapisa binarnih datoteka

Uobičajeno je pisanje/čitanje zapisa binarnih 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 00 9C
```

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

Primjer

- U binarnu datoteku `rez.bin` upisani su rezultati završnog ispita. Svaki zapis datoteke sadrži niz znakova *ime* (8+1 znak), niz znakova *prezime* (8+1 znak) i broj bodova (int).
- napisati odsječak programa kojim će prva dva zapisa iz datoteke `rez.bin` učitati u varijable

```
char ime1[8+1]; char prez1[8+1]; int brBod1;  
char ime2[8+1]; char prez2[8+1]; int brBod2;
```

Rješenje

- bez korištenja strukture -

...

```
FILE *ulTok;
```

```
char ime1[8+1]; char prez1[8+1]; int brBod1;
```

```
char ime2[8+1]; char prez2[8+1]; int brBod2;
```

```
ulTok = fopen("rez.bin", "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

- U binarnu datoteku `rez.bin` upisani su rezultati završnog ispita. Svaki zapis datoteke sadrži niz znakova *ime* (8+1 znak), niz znakova *prezime* (8+1 znak) i broj bodova (int).
- napisati odsječak programa kojim će se prva dva zapisa iz datoteke `rez.bin` učitati u varijable `rez1` i `rez2`, čija struktura odgovara zapisu datoteke:

```
struct zapisRez {  
    char ime[8+1];  
    char prez[8+1];  
    int brBod;  
} rez1, rez2;
```

Rješenje

- uz korištenje strukture -

```
...  
FILE *ulTok;  
struct zapisRez {  
    char ime[8+1];  
    char prez[8+1];  
    int brBod;  
} rez1, rez2;  
ulTok = fopen("rez.bin", "rb");  
  
fread(&rez1, sizeof(rez1), 1, ulTok);  
fread(&rez2, sizeof(rez2), 1, ulTok);  
...
```

Primjer

- Napisati program kojim će se stvoriti nova binarna datoteka tocke.bin koja sadržava točno 10^8 točaka (1.48 GB). Svaka točka pohranjena je kao struktura opisana u zadatku za izračunavanje centroida skupa točaka.
- Koordinate točaka generirati generatorom pseudoslučajnih brojeva. Koordinate su realni brojevi iz intervala $[0, 100]$, pri čemu mogu imati jednu decimalu iza decimalne točke.
- Primjer vrijednosti koordinata točaka:

```
(17.5, 30.0)  
(81.9, 0.3)  
(56.2, 0.0)  
(0.0, 19.8)  
...
```


Rješenje

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include "centroid.h"
#define BROJ_TOCAKA 100000000
int main(void) {
    struct tocka_s tocka; int i;
    FILE *izTok;
    izTok = fopen ("tocke.bin", "wb");
    srand((unsigned)time(NULL));
    for (i = 0; i < BROJ_TOCAKA; ++i) {
        tocka.x = (int)(rand()/((float)RAND_MAX+1)*1001)/10.;
        tocka.y = (int)(rand()/((float)RAND_MAX+1)*1001)/10.;
        fwrite(&tocka, sizeof(tocka), 1, izTok);
    }
    fclose(izTok);
    return 0;
}
```

Primjer

- Napisati program kojim će se za svaku grupu od po 100 000 točaka iz postojeće datoteke tocke.bin (iz prethodnog zadatka) pomoću funkcije centroid (iz zadatka u kojem se izračunavao centroid skupa točaka) izračunati i na zaslon ispisati njihov centroid.
- Može se računati na to da u datoteci tocke.bin sigurno ima točno 10^8 zapisa o točkama.

```
1. grupa: 50.07 49.85
2. grupa: 49.89 50.00
3. grupa: 49.93 49.89
...
998. grupa: 49.89 49.86
999. grupa: 50.09 50.10
1000. grupa: 49.86 49.99
```

Rješenje

- 1. dio -

```
#include <stdio.h>
#include "centroid.h"
#define TOCAKA_U_GRUPI 100000
#define BROJ_GRUPA 1000
int main(void) {
    struct tocka_s centar;
    int grupa, rbrToc;
    struct tocka_s skup[TOCAKA_U_GRUPI];
    FILE *ulTok;
    ulTok = fopen ("tocke.bin", "rb");
```

Rješenje


- 2. dio -

```
for (grupa = 0; grupa < BROJ_GRUPA; ++grupa) {
    for (rbrToc = 0; rbrToc < TOCAKA_U_GRUPI; ++rbrToc) {
        fread(&skup[rbrToc]
            , sizeof(struct tocka_s)
            , 1
            , ulTok);
    }
    centar = centroid(skup, TOCAKA_U_GRUPI);
    printf("%4d. grupa: %5.2f %5.2f\n"
        , grupa + 1
        , centar.x
        , centar.y);
}
fclose(ulTok);
return 0;
}
```

→ čita se jedna po jedna točka

Rješenje

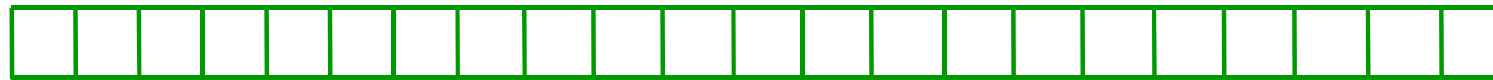
- 2. dio (bolje rješenje) -

```
for (grupa = 0; grupa < BROJ_GRUPA; ++grupa) {  
    fread(skup  
        , sizeof(struct tocka_s)  
        , TOCAKA_U_GRUPI  odjednom se čita 100 000 točaka  
        , ulTok);  
    centar = centroid(skup, TOCAKA_U_GRUPI);  
    printf("%4d. grupa: %5.2f %5.2f\n"  
        , grupa + 1  
        , centar.x  
        , centar.y);  
}  
fclose(ulTok);  
return 0;  
}
```

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`:

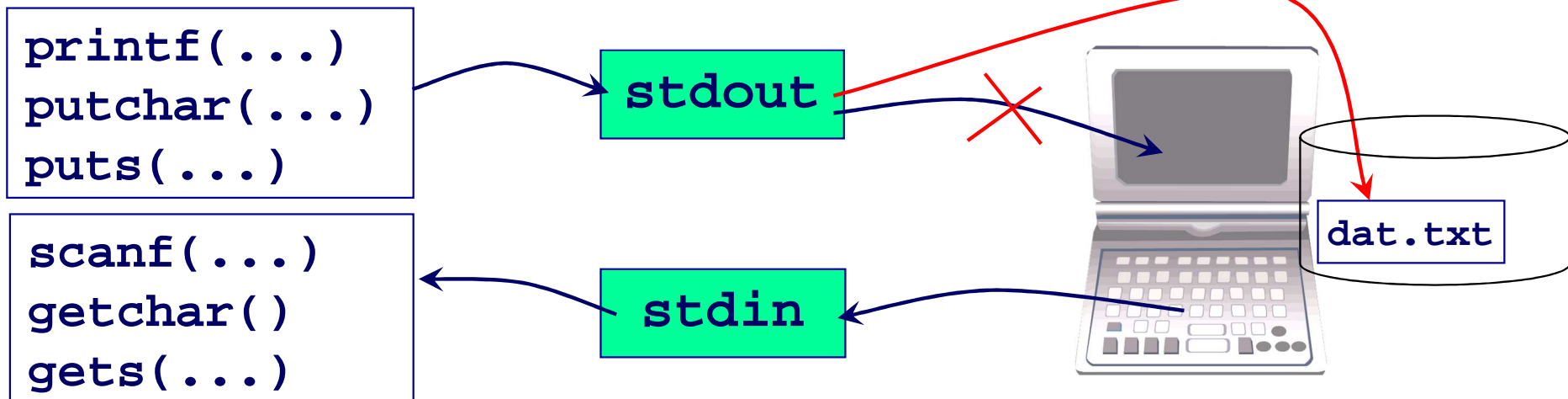
```
c:\papi>psi.exe
```

Ispis na `stdout` rezultira ispisom na zaslonu

```
c:\papi>psi.exe > dat.txt
```

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

program `psi.c`



Tok podataka `stderr`

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

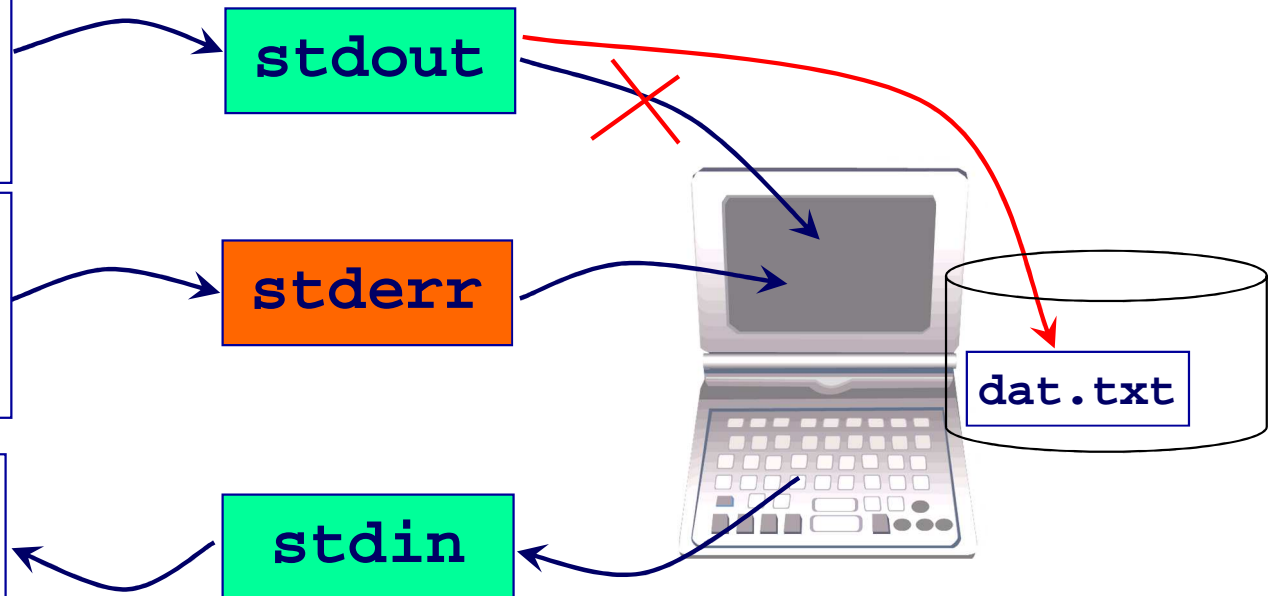
```
c:\pipi>pisi.exe > dat.txt
```

program `pisi.c`

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

```
fprintf(stderr ...)  
fputchar(stderr ...)  
fputs(stderr ...)
```

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



Primjer

- preusmjerenje standardnog izlaza -

program pisi.c

```
#include <stdio.h>
int main(void) {
    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>pisi.exe > dat.txt
```

U datoteku dat.txt upisat će se:

Prvi
Drugi
Cetvrti
Sesti

Na zaslonu će se ispisati:

Treci
Peti

Kome `return` i `exit` vraćaju rezultate?

program `komeReturnVraca.c`

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

```
int main(void) {
```

```
    return 99;
```

```
}
```

Operacijski sustav Windows

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

Operacijski sustavi se međusobno razlikuju. Npr. Unix:

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

```
$ komeReturnVraca
$ echo $?
99
```

Slijedne i direktne datoteke

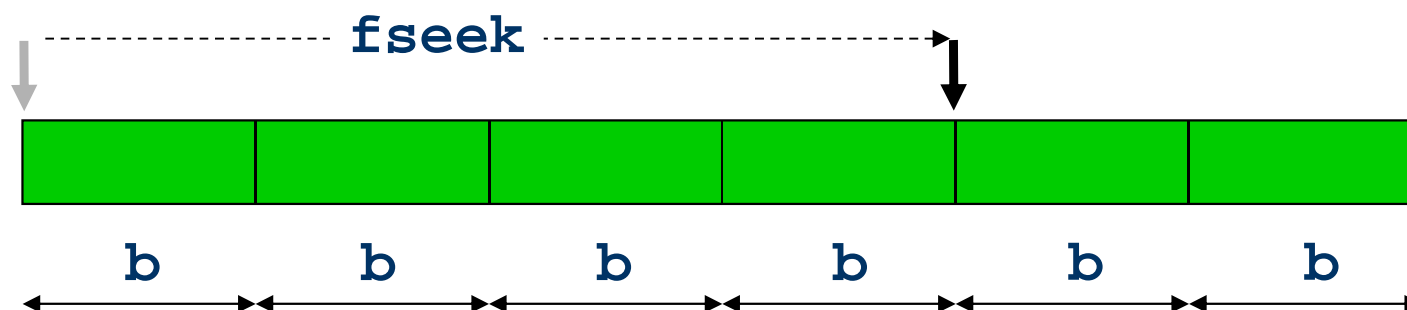
- zahvaljujući funkciji `fseek` moguće je pozicionirati se na bilo koju poziciju (redni broj bajta) unutar datoteke i obaviti operaciju čitanja ili pisanja
- međutim, kako pristupiti n-tom **zapisu** u datoteci?

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:

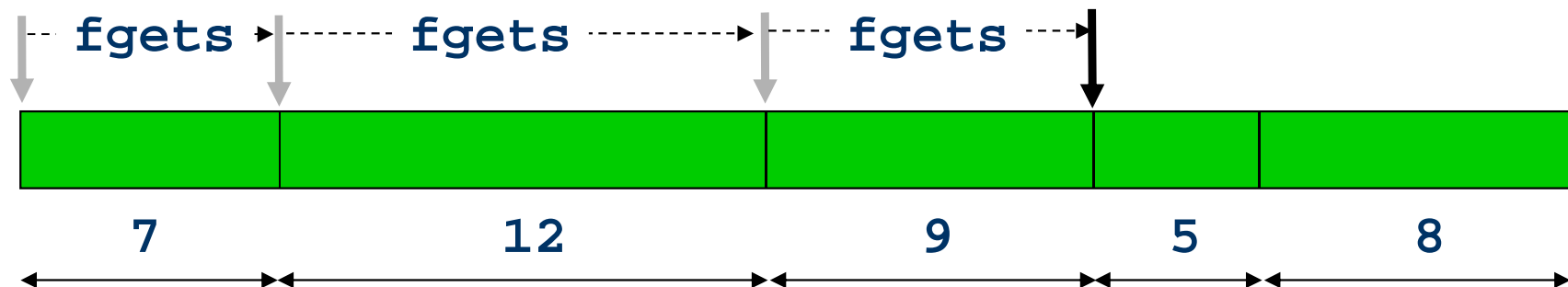
```
fseek (tok, (long)(n-1)*sizeof(struct zapis), SEEK_SET);
```

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(...);
```

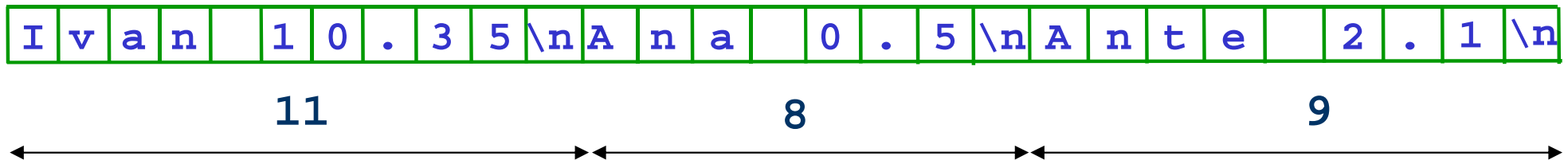
Primjer: pozicioniranje na početak 4. zapisa datoteke



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 izravno



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

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

Primjer


Primjer sadržaja datoteke "upis.txt":

1	2	3	4	5
12345678901234567890123456789012345678901234567890123				
1Peric Pero				194776.8
2Ivic Ivo				195286.3
6Markovic Marko				1940101.5
12Petrovic Petar				197267.8
3Anic Ana				196856.7

Primjer

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

Rješenje

- 1. dio -

```
#include <stdio.h>
#include <stdlib.h>
/* pomocna funkcija za ispis zadanog teksta
   i prekid programa */
void prekini(char *poruka) {
    fputs (poruka, stderr);
    exit (1);
}
int main(void) {
    FILE *du, *di;
    struct zapis_osobe {
        short mat_br;
        char prez_ime[40+1];
        short starost;
        float tezina;
    } zapis;
```

Rješenje

- 2. dio -

```
short tek_godina, god_rod;
```

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

```
if ((di = fopen ("tezine.bin", "wb")) == NULL)  
    prekini("Ne mogu otvoriti datoteku tezine.bin");
```

```
/* tekuca godina - program nece raditi dobro 2015 */  
tek_godina = 2014;
```


Rješenje

- 3. dio -

```
while (fscanf (du, "%4hd%40[^\n]%4hd%f",
            &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)
        prekini("Nije uspjelo pozicioniranje u tezine.bin");
    if (fwrite (&zapis, sizeof (zapis), 1, di) != 1)
        prekini("Nije uspjelo zapisivanje u tezine.bin");
}
fclose (du);
fclose (di);
return 0;
}
```

Primjer

- Izračunati prosječnu tjelesnu težinu osoba čiji su podaci upisani u datoteku "tezine.bin" (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 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 izvođenja programa

Prosjecna tezina je: 77.82

Unesite maticni broj: 1

1 Peric Pero

67 76.80

Napomena: osoba je laksa od prosjeka.

Unesite maticni broj: 2

2 Ivic Ivo

62 86.30

Unesite maticni broj: 6

6 Markovic Marko

74 101.50

Unesite maticni broj: 11

Rješenje

- 1. dio -

```
#include <stdio.h>
#include <stdlib.h>
int main(void) {
    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;
```

Rješenje

- 2. dio -

```
/* slijedno procitati sve zapise i  
   izracunati prosjek */  
du = fopen ("tezine.bin", "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);
```

Rješenje

- 3. dio -

```
while (1) {
    printf ("\nUnesite matični 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 ("Napomena: osoba je lakša od prosjeka.\n");
}
fclose(du);
return 0;
}
```

Primjer

U tekstualnoj 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 binarne 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:

	1	2	3	4
<i>12345678901234567890123456789012345678901234</i>				
Telefon Kanasonic	12	10.00	120.00	kn
CD Player Suny	2	1100.10	2200.20	kn
UKUPNO:			2320.20	kn

Rješenje

- 1. dio -

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

```
int main(void) {  
    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

- 2. dio -

```
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;
}
```

Primjer

- U postojeću tekstualnu datoteku `ulaz.txt`, koja se nalazi u kazalu `c:/tmp`, 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 tekstualnu datoteku `izlaz.txt` prepisati podatke o osobama čije prezime sadrži slovo **r**. Primjer sadržaja izlazne datoteke prikazan je ovdje:

```
Vurnek 101  
Voras 412
```

Rješenje

```
#include <stdio.h>
#include <string.h>
int main(void) {
    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;
}
```

Primjer

- U direktnoj binarnoj 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.

Primjer

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

- 1. dio -

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

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

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

```
int main(void) {
```

```
    FILE *dU1Iz1;
```

```
    struct {
```

```
        int mbr;
```

```
        char prezIme[21+1];
```

```
        int brBod;
```

```
    } zapis;
```

```
    int mbr;
```

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

Rješenje

- 2. dio -

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

/* Postavi kazaljku neposredno ISPRED
   odgovarajućeg zapisa */
fseek(dU1Iz1, (long)sizeof(zapis)*(mbr-1), SEEK_SET);

/* Procitaj cijeli zapis u strukturu */
fread(&zapis, sizeof(zapis), 1, dU1Iz1);

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

Rješenje

- 3. dio -

```
/* 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(dU1Izl, -1L*(long)sizeof(zapis), SEEK_CUR);
```

```
/* Zapisi sadržaj cijele strukture u datoteku */  
fwrite(&zapis, sizeof(zapis), 1, dU1Izl);
```

```
fclose(dU1Izl);
```

```
return 0;
```

```
}
```