Uvod u programiranje

- predavanja -

siječanj 2020.

13. Datoteke

Datoteke

Uvod

Memorija računala

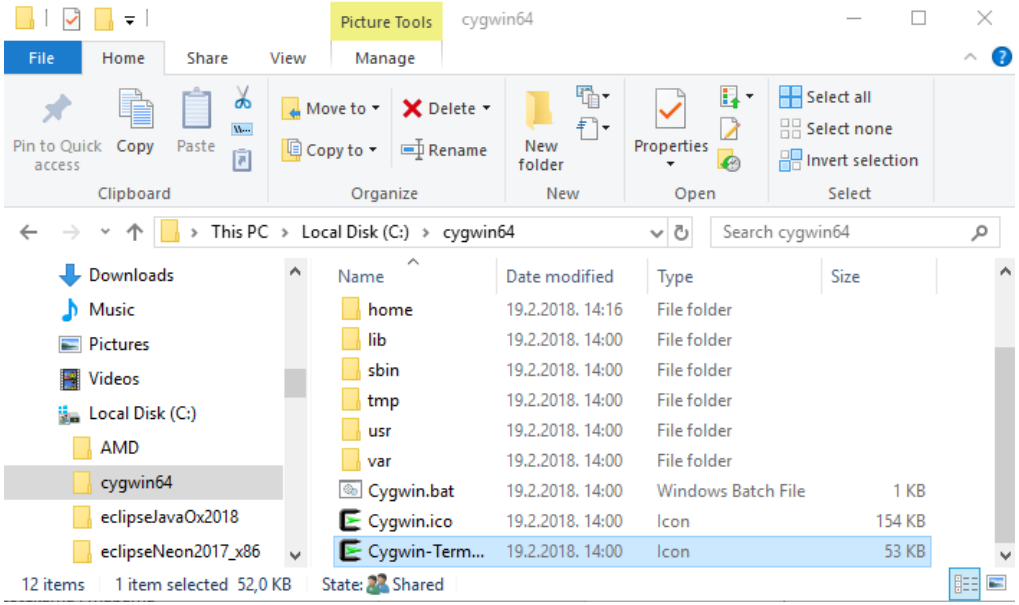
- Primarna
  - privremena (sadržaj se gubi po gubitku napajanja), relativno skupa, manjeg kapaciteta, brža
  - RAM (*Random Access Memory*)
- Sekundarna i tercijarna
  - trajna (sadržaj ostaje sačuvan po gubitku napajanja), relativno jeftina, većeg kapaciteta, sporija
  - sekundarna memorija
    - stalno priključena na računalo, npr. magnetski diskovi
  - tercijarna memorija
    - nije priključena na računalo, npr. kazete
    - treba je pronaći i priključiti ljudskom intervencijom ili automatikom

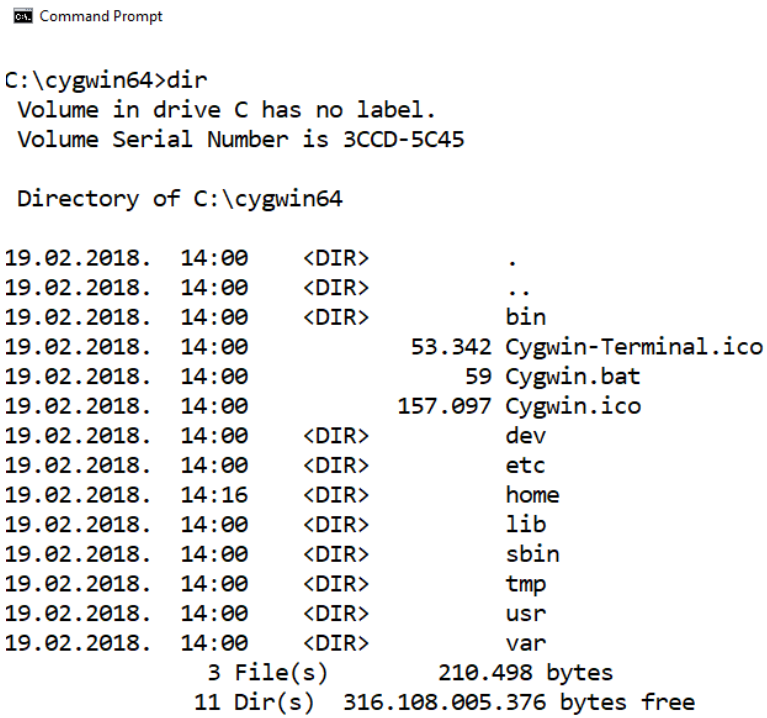
Sekundarna i tercijarna memorija računala

- s direktnim pristupom podacima
  - magnetski disk (HDD - *Hard Disk Drive*)
  - *flash* memorija (memory stick, SSD - *Solid State Drive,*)
  - optički diskovi (CD, DVD)
- sa slijednim pristupom podacima
  - magnetske trake

Operacijski sustav, datoteke i mape

- Operacijski sustav povezuje sklopovlje s programskom opremom
  - jedna od zadaća: preslikavanje fizičke organizacije podataka na mediju u logičku organizaciju koja se prema korisniku može prezentirati kao skup mapa i datoteka putem različitih sučelja
    - datotečni sustav (*file system*)



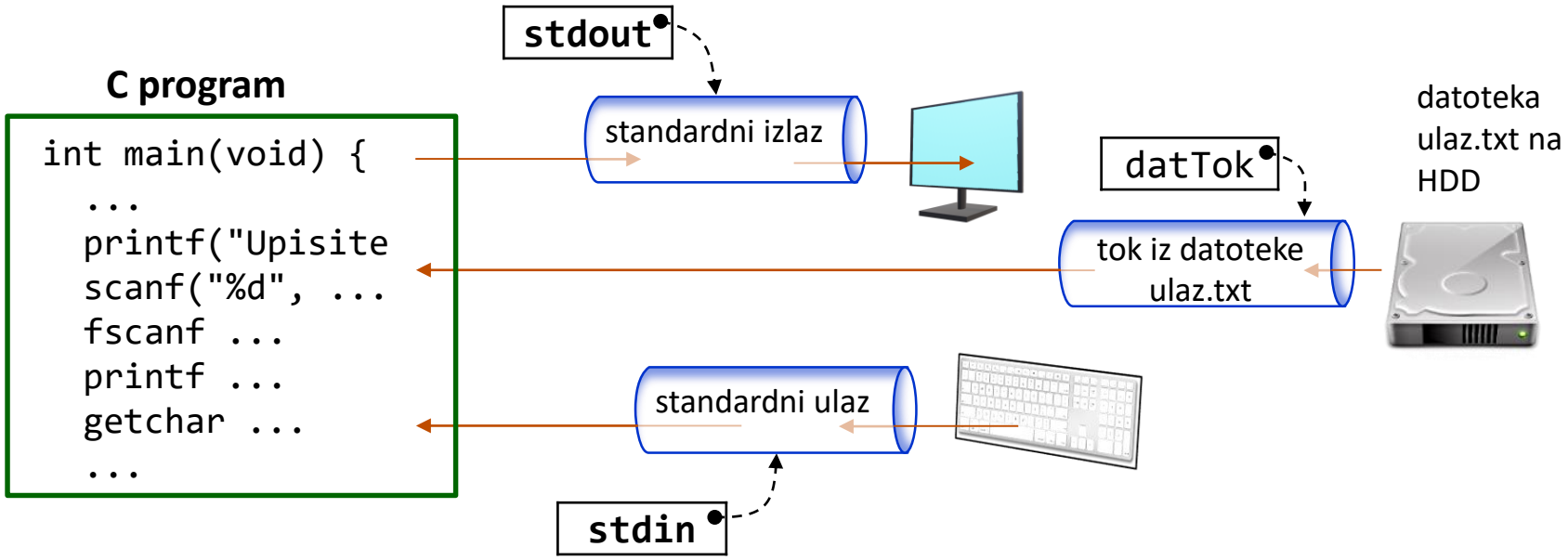


Datoteke i mape

- Datoteka (*file*)
  - imenovani skup podataka koji sačinjavaju logičku cjelinu, pohranjen na nekom od medija za pohranu
- Mapa, direktorij, kazalo (*folder*, *directory*)
  - datoteka koja sadrži popis drugih datoteka i mapa i podatke o njima. Mape su organizirane hijerarhijski, tvoreći strukturu nalik na stablo

Tok (*stream*)

- Za rad s datotekama koristi se aplikacijsko programsko sučelje koje se temelji na pojmu *tok*
  - pored tokova koji se otvaraju automatski (*stdin*, *stdout*, *stderr*) moguće je stvoriti (otvoriti) tok kojim će se programu omogućiti pristup podacima u datoteci



fopen <stdio.h>

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

- otvaranje toka za čitanje i/ili pisanje u datoteku
  - kolokvijalno se može reći: otvaranje datoteke
- `filename`: ime datoteke ili apsolutni ili relativni put (*path*) do datoteke
  - ako se navede samo ime datoteke, otvara se datoteka u radnoj mapi (*working directory*, *current directory*)
- `mode`: modalitet otvaranja i pristupa datoteci. Određuje npr. što se dešava ako datoteka koja se pokušava *otvoriti* tog trenutka ne postoji
- rezultat funkcije
  - ako je otvaranje toka uspjelo, funkcija vraća pokazivač na tok, tj. pokazivač na objekt tipa FILE
  - ako otvaranje toka nije uspjelo, funkcija vraća NULL

fopen <stdio.h>

- `mode` (modalitet otvaranja i pristupa datoteci)

	značenje	što se dešava s datotekom u trenutku otvaranja toka
`w`	(*write*) dopušteno je samo pisanje	ako datoteka postoji, **briše** sadržaj datoteke, inače stvara i otvara novu (praznu) datoteku
`a`	(*append*) dopušteno je samo pisanje, podaci koji se pišu automatski se dodaju na kraj datoteke	ako datoteka postoji, otvara tu datoteku, inače stvara i otvara novu (praznu) datoteku
`r`	(*read*) dopušteno je samo čitanje	ako datoteka postoji, otvara tu datoteku, inače funkcija `fopen` vraća NULL
`r+`	dopušteno je čitanje i pisanje	ako datoteka postoji, otvara tu datoteku, inače funkcija `fopen` vraća NULL
`w+`	dopušteno je pisanje i čitanje	ako datoteka postoji, otvara tu datoteku, inače stvara i otvara novu (praznu) datoteku
`a+`	dopušteno je pisanje i čitanje, podaci koji se pišu automatski se dodaju na kraj datoteke	ako datoteka postoji, otvara tu datoteku, inače stvara i otvara novu (praznu) datoteku

Dodavanjem oznake b (`wb`, `ab`, `rb`, `r+b`, `w+b`, `a+b`) specificira se *otvaranje* **binarne** datoteke.
Pojam **binarne** datoteke objašnjen je kasnije.

Primjer

```
FILE *tok1 = NULL, *tok2 = NULL, *tok3 = NULL;
```

```
tok1 = fopen("podaci.txt", "w");
```

Windows ili Linux: otvara datoteku `podaci.txt` u radnoj mapi. Dopušteno je samo pisanje. Ako datoteka ne postoji, stvara se. Ako postoji, postojeći sadržaj se briše

```
tok2 = fopen("D:/upro/primjeri/ulaz.txt", "r");
```

Windows: otvara datoteku `ulaz.txt` koja se nalazi u mapi `\upro\primjeri` na disku D (bez obzira koja je trenutačno radna mapa). Dopušteno je samo čitanje. Ako datoteka ne postoji, fopen vraća NULL

```
tok2 = fopen("/usr/upro/primjeri/ulaz.txt", "r");
```

Linux: slično kao prethodno, otvara datoteku ulaz.txt koja se nalazi u mapi `/usr/upro/primjeri`

```
tok3 = fopen("../../vjezba23/podaci", "r+b");
```

Windows ili Linux: otvara *binarnu* datoteku `podaci` koja se nalazi u mapi do koje je relativni put (u odnosu na radnu mapu) određen s `../../vjezba23`. Dopušteno je čitanje i pisanje. Ako datoteka ne postoji, fopen vraća NULL

fclose <stdio.h>

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

- zatvaranje toka na kojeg pokazuje parametar `stream`
  - kolokvijalno se može reći: zatvaranje datoteke
- rezultat funkcije
  - ako je zatvaranje toka uspjelo, vraća cijeli broj 0, inače EOF
- tokove koji se otvore treba zatvoriti u trenutku kada više nisu potrebni
  - time se oslobađaju resursi koje operacijski sustav troši dok je tok otvoren
  - omogućava se drugim korisnicima da otvore tok za tu datoteku
    - to ipak ne znači da tok treba otvarati i zatvarati nakon svake operacije čitanja ili pisanja u datoteku - otvaranje/zatvaranje toka je relativno "skupa" operacija
- ispravnim završetkom programa svi tokovi se automatski zatvaraju
  - ipak, ispravna praksa je pozivanjem ove funkcije eksplicitno zatvoriti sve tokove (osim tokova `stdin`, `stdout` i `stder`)

Primjer

- Programski zadatak
  - Sadržaj datoteke `ulaz.txt` koja se nalazi u radnoj mapi, znak po znak prepisati u datoteku (također u radnoj mapi) `izlaz.txt`. Istodobno, svaki znak koji se prepisuje iz jedne u drugu datoteku prikazati i na zaslonu
  - ako se pri otvaranju datoteke `ulaz.txt` dogodi pogreška (npr. datoteka ne postoji u trenutku pokretanja programa), ispisati poruku "Nije uspjelo otvaranje ulaz.txt" i prekinuti program uz status 10

  - datoteku `ulaz.txt` treba u radnoj mapi kreirati editorom (npr. editorom Notepad)

Primjer sadržaja datoteke `ulaz.txt`

```
Ovu datoteku smo napisali pomocu
obicnog editora kakav se koristi
za pisanje C programa.

Npr. Notepad ili Notepad++.
```

Rješenje

```
#include <stdio.h>
#include <stdlib.h>
int main(void) {
   FILE *tokUlaz = NULL;
   tokUlaz = fopen("ulaz.txt", "r");
   if (tokUlaz == NULL) {
      printf("Nije uspjelo otvaranje ulaz.txt");
      exit(10);
   }

   FILE *tokIzlaz = fopen("izlaz.txt", "w");
   int c;
   while ((c = getc(tokUlaz)) != EOF) {
      putchar(c);  // ili putc(c, stdout);
      putc(c, tokIzlaz);
   }
   fclose(tokUlaz);
   fclose(tokIzlaz);

   return 0;
}
```

Primjer

- Programski zadatak
  - Iz datoteke `cijeli.txt` čitati cijele brojeve, svaki pročitani broj pomnožiti realnim brojem 0.5 (standardne preciznosti) te rezultat, svaki u svom retku, upisati u datoteku `realni.txt`. Pri pisanju realnih brojeva koristiti konverzijsku specifikaciju `%5.1f`

Primjer sadržaja datoteke `cijeli.txt`

```
-12 15
   -3+8
 7
```

Primjer sadržaja datoteke `realni.txt`

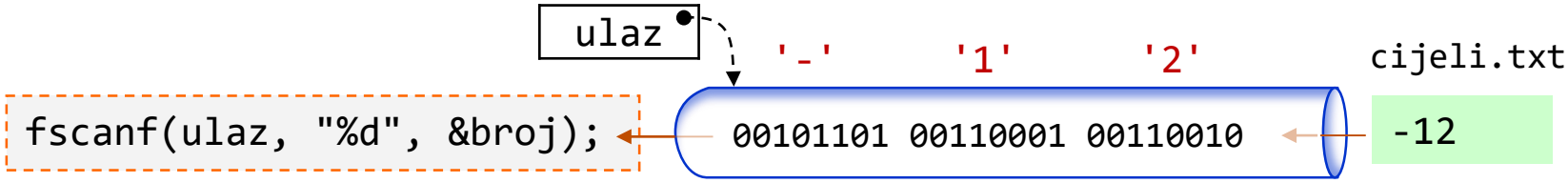
```
 -6.0
  7.5
-1.5
 4.0
 3.5
```

Rješenje

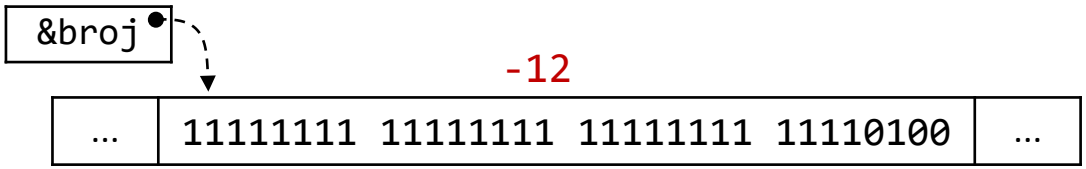
```
#include <stdio.h>

int main(void) {
   FILE *ulaz = fopen("cijeli.txt", "r");
   FILE *izlaz = fopen("realni.txt", "w");

   int broj;
   float realniBroj;
   while (fscanf(ulaz, "%d", &broj) == 1) {
      realniBroj = broj * 0.5f;
      fprintf(izlaz, "%5.1f\n", realniBroj);
   }

   fclose(ulaz);
   fclose(izlaz);

   return 0;
}
```

Tekstne datoteke

- Što se točno u prethodnom primjeru dešava pri čitanju iz ulaznog toka?



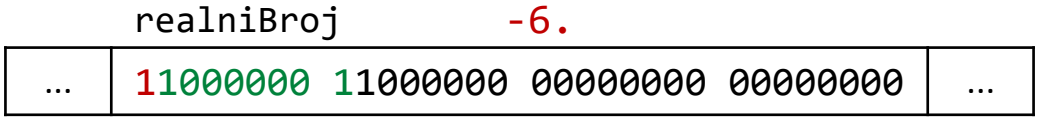
  - obavlja se konverzija pročitanih **znakova** u podatak odgovarajućeg tipa (prema konverzijskoj specifikaciji)
  - rezultat dobiven konverzijom upisuje se na mjesto u memoriji na koje pokazuje argument `&broj`, dakle u varijablu `broj`



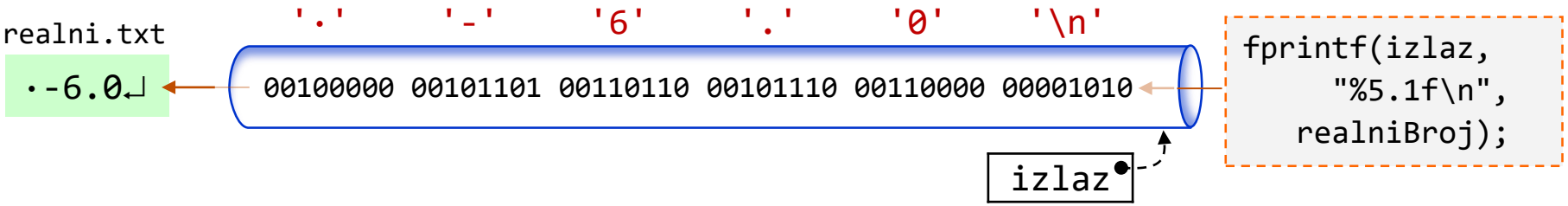
  - s obzirom da je u datoteci pohranjen tekst, da se konverzija u binarni oblik obavlja "prema formatu", ovakve datoteke nazivaju se *tekstne* ili *formatirane* datoteke
    - sadržaj takvih datoteka moguće je pregledavati i uređivati editorom

Tekstne datoteke

- Simetrično, pri pisanju u izlazni tok



  - obavlja se konverzija binarnog sadržaja varijable `realniBroj` u niz znakova (njihovih ASCII vrijednosti) koje će se pohraniti u datoteku `realni.txt`



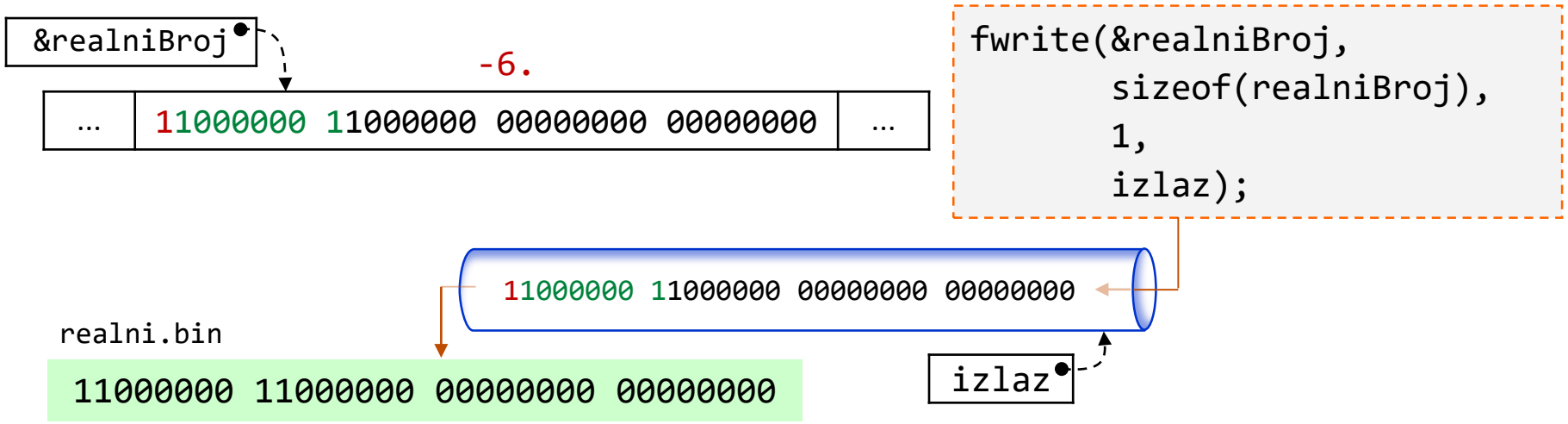
  - rezultat je tekstna (formatirana) datoteka koja sadrži ASCII vrijednosti znakova

Tekstne datoteke

- Tokovi za tekstne datoteke imaju karakteristike i koriste se na isti način kao tokovi `stdin`, `stdout`, `stderr`
  - npr. kada se tok standardni izlaz preusmjeri u datoteku, dobije se tekstna datoteka
  - funkcije koje se koriste za obavljanje operacija u tekstnim datotekama
    - `fprintf`, `fscanf`
    - `getc`, `ungetc`, `putc`
    - `fgets`, `fputs`

Binarne datoteke

- Sadržaj memorije također je moguće pisati ili čitati iz datoteke u binarnom obliku, dakle *bez konverzije prema formatu*



  - s obzirom da je u datoteci pohranjen *binarni* sadržaj, da se pri čitanju/pisanju *ne obavlja konverzija prema formatu*, ovakve datoteke nazivaju se *binarne* ili *neformatirane* datoteke
    - sadržaj takvih datoteka nije moguće pregledavati i uređivati (običnim) editorom

fwrite <stdio.h>

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

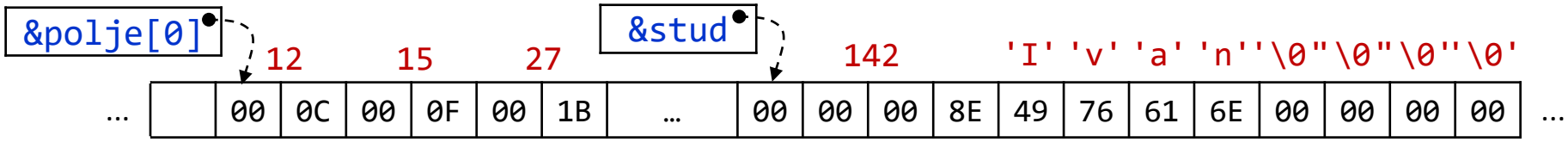
- u tok (u datoteku) na kojeg pokazuje `stream` upisuje se sadržaj memorije na kojeg pokazuje ptr
  - `size`: veličina pojedinačnog objekta koji se upisuje (u bajtovima)
  - `nmemb`: broj objekata koji se upisuje
- ukupna veličina memorije koja se upisuje je veličine `size·nmemb` bajtova
- rezultat funkcije
  - broj objekata koji je uspješno upisan

Primjer

- U binarnu datoteku `podaci.bin` upisati sadržaj sljedećeg polja i strukture

```
short polje[] = {12, 15, 27};
struct osoba_s {
   int rbr;
   char ime[7 + 1];
};
struct osoba_s stud = {142, "Ivan"};
```

Podsjetnik: članovi jednog polja, **ali i članovi jedne strukture**, uvijek su smješteni u kontinuiranom području memorije, redom jedan član neposredno iza drugog.




```
FILE *bin = fopen("podaci.bin", "wb");
fwrite(&polje[0], sizeof(short), 3, bin);     ili sizeof(polje[0])
ili fwrite(&polje[0], sizeof(polje), 1, bin);

fwrite(&stud, sizeof(stud), 1, bin);     ... fclose...
```

`podaci.bin`

```
00 0C 00 0F 00 1B 00 00 00 8E 49 76 61 6E 00 00 00 00
```

fread <stdio.h>

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

- iz toka (iz datoteke) na kojeg pokazuje `stream` sadržaj se čita i upisuje u memoriju na mjesto na koje pokazuje ptr
  - `size`: veličina jednog objekta koji se čita (u bajtovima)
  - `nmemb`: broj objekata koji se čita
- ukupna veličina memorije koja se čita je veličine `size·nmemb` bajtova
- rezultat funkcije
  - broj objekata koji je uspješno pročitan

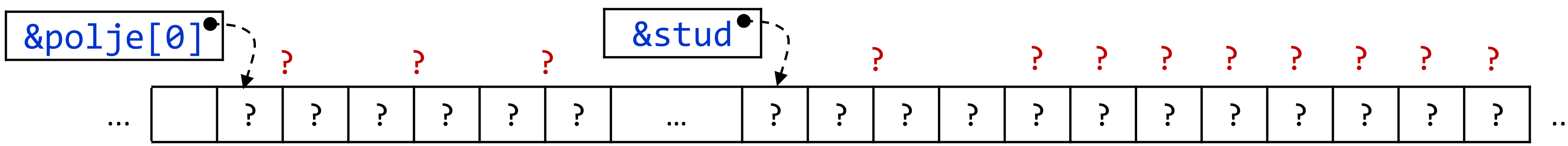
Primjer

podaci.bin

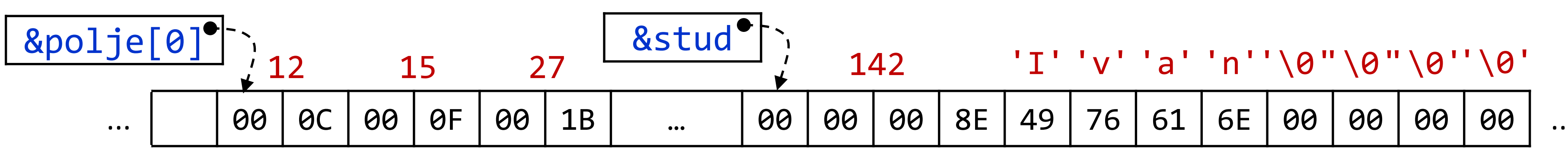
`00 0C 00 0F 00 1B 00 00 00 8E 49 76 61 6E 00 00 00 00`

- Pročitati sadržaj iz binarne datoteke `podaci.bin`
  - naravno, moramo unaprijed znati točnu strukturu podataka u datoteci

```
short polje[3];
struct osoba_s {
   int rbr;
   char ime[7 + 1];
};
struct osoba_s stud;
```




```
FILE *bin = fopen("podaci.bin", "rb");
fread(&polje[0], sizeof(short), 3, bin); ili sizeof(polje[0])
ili fread(&polje[0], sizeof(polje), 1, bin);

fread(&stud, sizeof(stud), 1, bin);     ... fclose...
```



Primjer

- Programski zadatak
  - Sadržaj postojeće tekstne datoteke `bodovi.txt` prepisati u novu binarnu datoteku `bodovi.bin`
  - Ime i prezime ne sadrže praznine, niti jedno nije dulje od 8 znakova. Broj bodova je cijeli broj manji od 100 000

Primjer sadržaja datoteke `bodovi.txt`

```
Iva Pek 156↵
Ante Horvat 12↵
```

```
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 0A
```

  - svaki zapis datoteke `bodovi.bin` sadrži: niz znakova *ime* (7+1 znak), niz znakova *prezime* (7+1 znak) i cijeli broj *broj bodova* (int)

Zapis datoteke (*record*): skup susjednih podataka unutar datoteke koji se obrađuje kao cjelina.

Primjer sadržaja datoteke `bodovi.bin`

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

Rješenje

```
...
struct ispit_s {
   char ime[7 + 1];
   char prez[7 + 1];
   int brBod;
} ispit;

FILE *ulTok = fopen("bodovi.txt", "r");
FILE *izTok = fopen("bodovi.bin", "wb");

while (fscanf(ulTok, "%s %s %d",
              ispit.ime, ispit.prez, &ispit.brBod) == 3) {
   fwrite(&ispit, sizeof(ispit), 1, izTok);
}

fclose(ulTok);
fclose(izTok);
...
```

Primjer

- Programski zadatak
  - Napisati program kojim će se stvoriti nova binarna datoteka `tocke.bin` koja sadržava točno 10^8 točaka (~1.5 GB). Svaka točka pohranjena je kao sljedeća struktura:

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

  - Koordinate točaka generirati generatorom pseudoslučajnih brojeva. Koordinate trebaju biti realni brojevi iz intervala [0, 100]

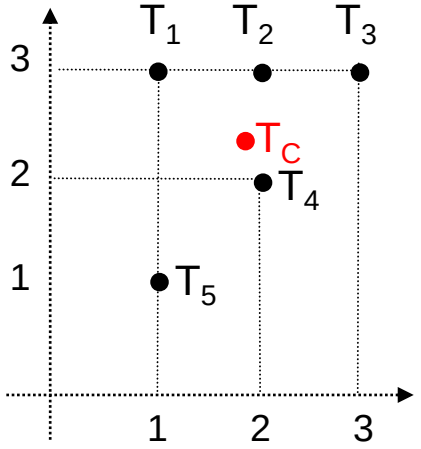
Rješenje

```
...
#define BROJ_TOCAKA 100000000

int main(void) {
   struct tocka_s {
      double x;
      double y;
   } tocka;

   FILE *izTok = fopen("tocke.bin", "wb");
   srand((unsigned)time(NULL));

   for (int i = 0; i < BROJ_TOCAKA; ++i) {
      tocka.x = (float)rand() / RAND_MAX * 100.f;
      tocka.y = (float)rand() / RAND_MAX * 100.f;
      fwrite(&tocka, sizeof(tocka), 1, izTok);
   }
   fclose(izTok);

   return 0;
}
```

Primjer

- Programski zadatak
  - Napisati program kojim će se za svaku grupu od po 100 000 točaka iz postojeće datoteke tocke.bin (iz prethodnog zadatka) 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.

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




```
   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>

#define TOCAKA_U_GRUPI 100000
#define BROJ_GRUPA 1000

int main(void) {
   struct tocka_s {
      double x;
      double y;
   };

   FILE *ulTok = fopen("tocke.bin", "rb");
```

Rješenje (2. dio)

varijanta s čitanjem *točka po točka* (10^8 čitanja po 16 bajtova)

```
   struct tocka_s tocka;

   for (int grupa = 0; grupa < BROJ_GRUPA; ++grupa) {
      float xCent = 0.f, yCent = 0.f;
      for (int rbrToc = 0; rbrToc < TOCAKA_U_GRUPI; ++rbrToc) {
         fread(&tocka, sizeof(struct tocka_s), 1, ulTok);
         xCent += tocka.x;
         yCent += tocka.y;
      }

      xCent /= TOCAKA_U_GRUPI;
      yCent /= TOCAKA_U_GRUPI;
      printf("%4d. grupa: %5.2f %5.2f\n", grupa + 1, xCent, yCent);
   }

   fclose(ulTok);

   return 0;
}
```

Rješenje (2. dio)

varijanta s čitanjem grupa od po 100 000 točaka (1 000 čitanja po 1 600 000 bajtova)

```
   struct tocka_s skup[TOCAKA_U_GRUPI];

   for (int grupa = 0; grupa < BROJ_GRUPA; ++grupa) {
      fread(&skup[0], sizeof(struct tocka_s), TOCAKA_U_GRUPI, ulTok);

      float xCent = 0.f, yCent = 0.f;
      for (int rbrToc = 0; rbrToc < TOCAKA_U_GRUPI; ++rbrToc) {
         xCent += skup[rbrToc].x;
         yCent += skup[rbrToc].y;
      }

      xCent /= TOCAKA_U_GRUPI;
      yCent /= TOCAKA_U_GRUPI;
      printf("%4d. grupa: %5.2f %5.2f\n", grupa + 1, xCent, yCent);
   }

   fclose(ulTok);

   return 0;
}
```

Datoteke

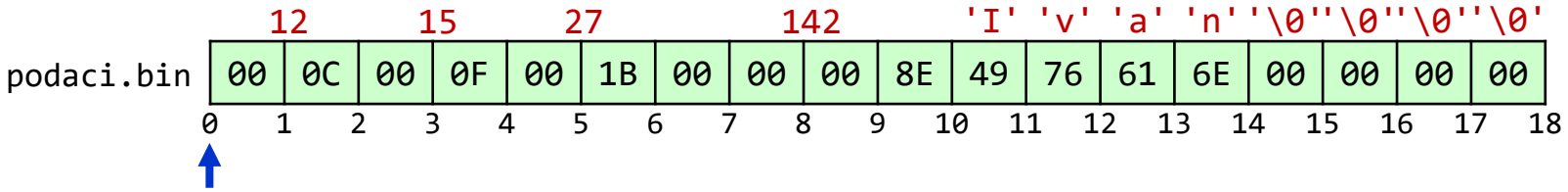
Indikator pozicije

Indikator pozicije u datoteci

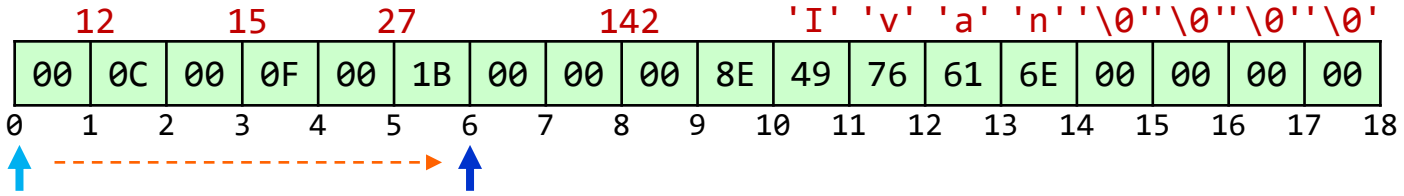
```
short polje[3];
struct osoba_s {
   int rbr;
   char ime[7 + 1];
};
struct osoba_s stud;
```

- Podaci se uvijek čitaju ili pišu počevši od mjesta u datoteci na koje trenutačno pokazuje indikator pozicije (*file position indicator*)

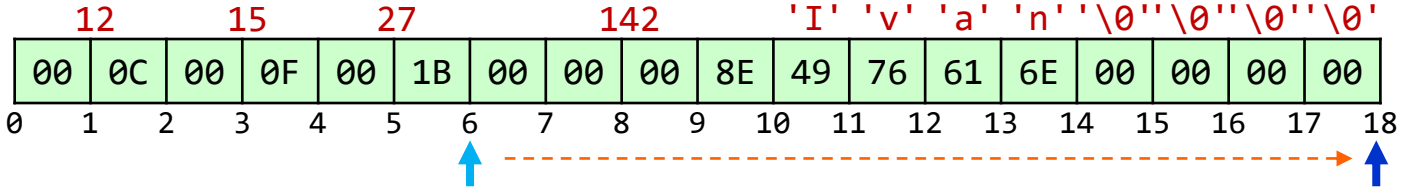
```
FILE *bin = fopen("podaci.bin", "rb");
```




```
fread(&polje[0], sizeof(polje), 1, bin);
```




```
fread(&stud, sizeof(stud), 1, bin);
```

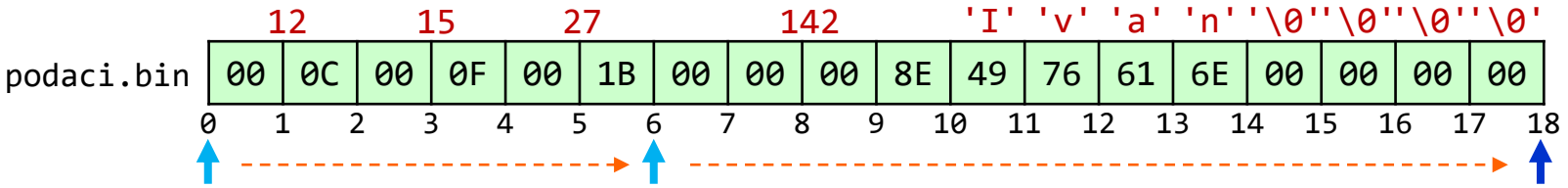


ftell <stdio.h>

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

- rezultat funkcije: vrijednost indikatora pozicije za tok `stream`, tj. udaljenost od početka datoteke za koju je otvoren tok na kojeg pokazuje `stream`, izražena u broju bajtova
  - ako je indikator pozicije na samom početku datoteke, rezultat je `0L`
  - u slučaju pogreške, funkcija vraća `-1L`

```
FILE *bin = fopen("podaci.bin", "rb");
```




```
printf("%ld", ftell(bin));                    0
fread(&polje[0], sizeof(polje), 1, bin);
printf("%ld", ftell(bin));                    6
fread(&stud, sizeof(stud), 1, bin);
printf("%ld", ftell(bin));                    18
```

fseek <stdio.h>

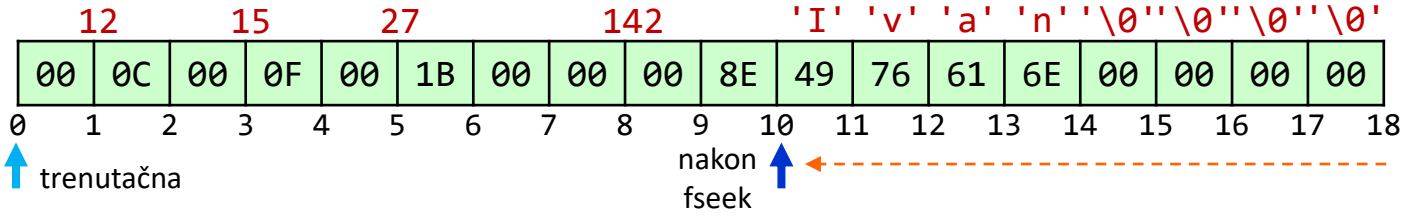
```
int fseek(FILE *stream, long offset, int whence);
SEEK_SET              makro:  u odnosu na početak datoteke
SEEK_CUR                      u odnosu na trenutačnu poziciju
SEEK_END                      u odnosu na kraj datoteke
```

- indikator pozicije za tok `stream` pomiče na poziciju koja je za `offset` bajtova udaljena od:
  - početka datoteke (za `whence` = `SEEK_SET`)
  - tekuće pozicije (za `whence` = `SEEK_CUR`)
  - kraja datoteke (za `whence` = `SEEK_END`)
- rezultat funkcije:
  - u slučaju pogreške vraća cijeli broj različit od nule, inače nulu

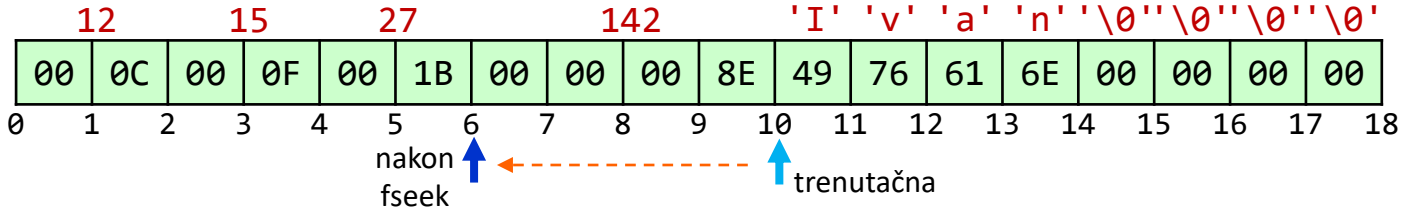
Primjer

```
short polje[3];
struct osoba_s {
   int rbr;
   char ime[7 + 1];
};
struct osoba_s stud;
```

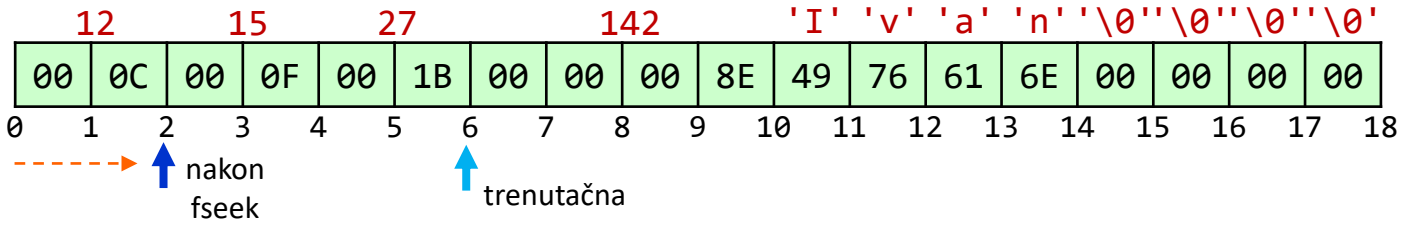
```
fseek(bin, -8L, SEEK_END);
```




```
fseek(bin, -1L * sizeof(stud.rbr), SEEK_CUR);
```




```
fseek(bin, (long)sizeof(polje[0]), SEEK_SET);
```



Datoteke

Slijedni i direktni pristup zapisima u datoteci

Slijedni pristup zapisima u datoteci

- zapisima se pristupa redom, slijedno
  - jedan po jedan zapis čita se od početka prema kraju datoteke, dok se ne pročitaju svi zapisi ili dok se ne pročita jedan ili više zapisa koje je potrebno obraditi
    - kolokvijalno: slijedna obrada datoteke
  - slijedni pristup podacima je primjenjiv u svim datotekama (neovisno imaju li fiksne duljina zapisa i na koji način su zapisi pozicionirani u datoteci), ali nije uvijek dovoljno efikasan
    - slijedni pristup se koristi kada se trebaju obraditi svi ili većina zapisa u datoteci ili kada direktni pristup do zapisa nije moguć

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

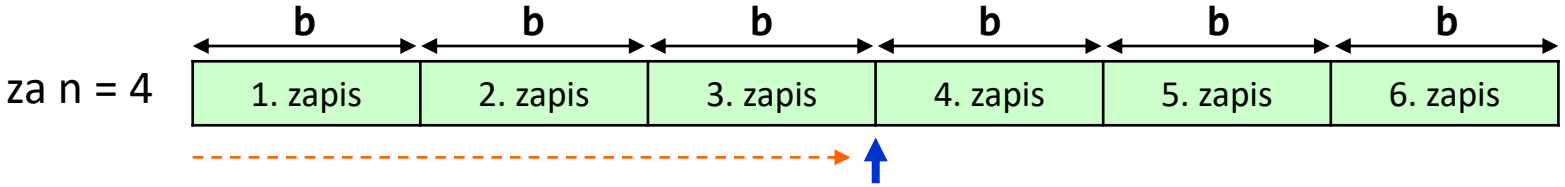
```
fscanf(tok, "%s %f", ime, &bodovi);   čitanje 1. zapisa
fscanf(tok, "%s %f", ime, &bodovi);   čitanje 2. zapisa
fscanf(tok, "%s %f", ime, &bodovi);   čitanje 3. zapisa
```

    - nije moguće pročitati treći zapis bez da se pročitaju prethodni

Direktni pristup zapisima u datoteci

- zapisima se pristupa izravno, direktno, na temelju rednog broja zapisa (kolokvijalno: direktna obrada datoteke)
  - zapisi u datoteci **moraju** biti fiksne duljine (npr. `b` bajtova)
  - iz rednog broja zapisa, npr. `n` (prvi zapis u datoteci ima redni broj 1), izračuna se pozicija na kojoj se u datoteci nalazi dotični zapis
  - indikator pozicije se pomakne na početak tog zapisa

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



  - zapis se zatim pročita (ili se zapiše nova vrijednost zapisa)
- Primjer: pomak indikatora pozicije na početak n-tog zapisa datoteke u kojoj zapisi odgovaraju strukturi `zapis_s`

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

Direktni pristup zapisima u datoteci

- mogućnost direktnog pristupa n-tom zapisu u datoteci praktično je iskoristiva samo onda kada redni broj zapisa odgovara nekom ključu potrage ili se iz ključa potrage može izračunati, npr:
  - zapis o osobi s matičnim brojem n uvijek je smješten kao n-ti zapis u datoteci (ključ potrage u ovom slučaju je matični broj osobe)
- moguće je da će neki zapisi biti "prazni", npr.
  - ako su u datoteku upisani podaci o 100 osoba, a ne postoje osobe s matičnim brojevima 2, 17, 33, 34

- Primjer (kada se redni broj zapisa može *izračunati* iz ključa potrage):
  - ako datoteka sadrži zapise o mjestima (poštanski broj i naziv mjesta, a raspon poštanskih brojeva je od 10 000 do 60 000), kako omogućiti direktni pristup do zapisa na temelju zadanog poštanskog broja?

Primjer

- Programski zadatak
  - svaki zapis tekstne datoteke `drzave.txt` sadrži numeričku šifru, kraticu naziva i naziv države (međusobno su odvojeni znakom praznine). Šifra države je pozitivni cijeli broj, a naziv države nije dulji od 40 znakova
  - napisati program kojim će se na zaslon ispisati:
    - broj država koje su upisane u datoteku
    - najveća šifra države

`drzave.txt`

```
1 PR Puerto Rico↵
248 AX Aland Islands↵
3 AL Albania↵
24 AO Angola↵
...
897 TG Togo↵
898 PN Pitcairn↵
894 ZM Zambia↵
79 ZW Zimbabwe↵
895 CK Cook Islands↵
732 EH Western Sahara↵
```

- Analiza
  - potrebno je pročitati sve zapise datoteke
    - prikladno je koristiti slijedni pristup zapisima

Rješenje

```
...
int sifDrz, maxSif, brojac = 0;

FILE *tok = fopen("drzave.txt", "r");

while (fscanf(tok, "%d %*s %*[^\n]", &sifDrz) == 1) {
   ++brojac;
   if (brojac == 1 || sifDrz > maxSif) {
      maxSif = sifDrz;
   }
}
printf("Broj zapisa = %d, najveca sifra = %d", brojac, maxSif);

fclose(tok);
...
```

Primjer

- Programski zadatak
  - s tipkovnice učitati jedan cijeli broj. Ako u datoteci `drzave.txt` postoji država sa šifrom koja odgovara učitanom broju, na zaslon ispisati naziv te države. Inače, ispisati poruku "Nema drzave s tom sifrom"

`drzave.txt`

```
1 PR Puerto Rico↵
248 AX Aland Islands↵
3 AL Albania↵
24 AO Angola↵
...
897 TG Togo↵
898 PN Pitcairn↵
894 ZM Zambia↵
79 ZW Zimbabwe↵
895 CK Cook Islands↵
732 EH Western Sahara↵
```

- Analiza
  - jedini način kako pristupiti traženom zapisu jest redom čitati zapis po zapis datoteke i pročitanu šifru uspoređivati s traženom šifrom
  - ponavljati dok se ne pronađe zapis s odgovarajućom šifrom ili dok se ne pročitaju svi zapisi
  - postupak nije efikasan (slijedna obrada), ali u ovom slučaju jedini moguć

Rješenje

```
...
int trazimSifru, sifDrz;
char naz[40 + 1];

FILE *tok = fopen("drzave.txt", "r");

printf("Upisite sifru drzave > ");
scanf("%d", &trazimSifru);

int procitano;
do {
   procitano = fscanf(tok, "%d %*s %[^\n]", &sifDrz, naz);
} while (sifDrz != trazimSifru && procitano == 2);

if (sifDrz == trazimSifru && procitano == 2) {
   printf("%s\n", naz);
} else {
   printf("Nema drzave s tom sifrom\n");
}

fclose(tok);
...
```

Primjer

- Programski zadatak
  - Vrlo slično prethodnom zadatku: s tipkovnice učitavati po jedan cijeli broj dok god upisani cijeli broj odgovara šifri neke od država u datoteci `drzave.txt`. Ako postoji država sa šifrom koja odgovara učitanom broju, na zaslon ispisati naziv te države, inače, ispisati poruku "Nema drzave s tom sifrom" i prekinuti daljnje učitavanje cijelih brojeva.

`drzave.txt`

```
1 PR Puerto Rico↵
248 AX Aland Islands↵
3 AL Albania↵
24 AO Angola↵
...
897 TG Togo↵
898 PN Pitcairn↵
894 ZM Zambia↵
79 ZW Zimbabwe↵
895 CK Cook Islands↵
732 EH Western Sahara↵
```

Neispravno rješenje

```
...
FILE *tok = fopen("drzave.txt", "r");

while (1 == 1) {
   printf("Upisite sifru drzave > ");
   scanf("%d", &trazimSifru);

   int procitano;
   do {
      procitano = fscanf(tok, "%d %*s %[^\n]", &sifDrz, naz);
   } while (sifDrz != trazimSifru && procitano == 2);

   if (sifDrz == trazimSifru && procitano == 2) {
      printf("%s\n", naz);
   } else {
      printf("Nema drzave s tom sifrom\n");
      break;
   }
}
...
```

Nakon pretrage datoteke za jednu šifru učitanu s tipkovnice, indikator pozicije u toku **tok** ostao je iza pronađenog zapisa ili na kraju datoteke (ako zapis s odgovarajućom šifrom nije pronađen). To znači da pretraga za sljedeću šifru države učitanu s tipkovnice ne bi počela od prvog zapisa u datoteci.

Ispravno rješenje

```
...
FILE *tok = fopen("drzave.txt", "r");

while (1 == 1) {
   printf("Upisite sifru drzave > ");
   scanf("%d", &trazimSifru);

   int procitano;
   do {
      procitano = fscanf(tok, "%d %*s %[^\n]", &sifDrz, naz);
   } while (sifDrz != trazimSifru && procitano == 2);

   if (sifDrz == trazimSifru && procitano == 2) {
      printf("%s\n", naz);

      fseek(tok, 0L, SEEK_SET);     // indikator pozicije vrati na početak datoteke
   } else {
      printf("Nema drzave s tom sifrom\n");
      break;
   }
}
...
```

Nakon pretrage datoteke za jednu šifru učitanu s tipkovnice, indikator pozicije u toku **tok** vraća se na početak toka, kako bi pretraga za sljedeću šifru države učitanu s tipkovnice mogla početi od prvog zapisa u datoteci.

Primjer

drzave.txt

```
1 PR Puerto Rico↵
248 AX Aland Islands↵
3 AL Albania↵
...
```

- Programski zadatak
  - zapise iz tekstne datoteke `drzave.txt` treba prepisati u binarnu datoteku `drzave.bin`
  - svaki zapis datoteke `drzave.bin` treba sadržavati
    - šifru (int), dvoslovnu kraticu (2+1 znak) i naziv države (40+1 znak)
    - redni broj zapisa u datoteci `drzave.bin` mora odgovarati šifri države

0 →	1	PR	Puerto Rico
48 →	0000	000	0000000000000000
96 →	3	AL	Albania
144 →	...		
42912 →	895	CK	Cook Islands
42960 →	0000	000	0000000000000000
43008 →	897	TG	Togo
43056 →	898	PN	Pitcairn
43104 →			

- prazni zapisi (npr. ne postoji država sa šifrom 2 ili država sa šifrom 896) sadrže vrijednosti 0. Ako se iz datoteke pročita zapis kojem je šifra nula, to znači da takvog zapisa "nema".
- kako onda "obrisati" zapis? Vrijednost šifre postaviti na nulu
- ako bi se u datoteku na odgovarajuću poziciju upisao npr. zapis s rednim brojem 900, sustav bi prostor zapisa s rednim brojem 899 sam popunio nulama

Rješenje

```
...
struct drz_s {
   int sifDrz;
   char krat[2 + 1];
   char naz[40 + 1];
} drzava;

FILE *ulaz = fopen("drzave.txt", "r");
FILE *izlaz = fopen("drzave.bin", "wb");

while (fscanf(ulaz, "%d %s %[^\n]",
              &drzava.sifDrz, drzava.krat, drzava.naz) == 3) {
   fseek(izlaz, (long)(drzava.sifDrz - 1) * sizeof(drzava), SEEK_SET);
   fwrite(&drzava, sizeof(drzava), 1, izlaz);
}

fclose(ulaz);
fclose(izlaz);
...
```

Primjer

- Programski zadatak
  - svaki zapis datoteke `drzave.bin` sadrži
    - šifru (int), dvoslovnu kraticu (2+1 znak) i naziv države (40+1 znak)
    - redni broj zapisa u datoteci `drzave.bin` odgovara šifri države
  - s tipkovnice učitati jedan cijeli broj. Ako u datoteci `drzave.bin` postoji država sa šifrom koja odgovara učitanom broju, na zaslon ispisati naziv te države. Inače, ispisati poruku "Nema drzave s tom sifrom"

- Analiza
  - iz šifre države lako je izračunati poziciju (redni broj bajta) na kojoj započinje zapis o toj državi (ako takav zapis postoji). Koristiti direktni pristup zapisu
    - u ovom primjeru bilo bi **vrlo pogrešno** koristiti slijedni pristup zapisu
  - postaviti indikator pozicije na početak zapisa i pročitati ga. Ako je pročitana šifra jednaka nuli, tada zapisa s traženom šifrom u datoteci nema

Rješenje

U binarnim datotekama treba čitati (također i pisati) uvijek cijeli zapis, koristeći pri tome strukturu. Bilo bi pogrešno čitanje obaviti ovako:

```
fread(&drzava.sifDrz, sizeof(drzava.sifDrz), 1, tok);
fread(drzava.krat, sizeof(drzava.krat), 1, tok);
fread(drzava.naz, sizeof(drzava.naz), 1, tok);
```

```
...
struct drz_s {
   int sifDrz;
   char krat[2 + 1];
   char naz[40 + 1];
} drzava;
int trazimSifru;

scanf("%d", &trazimSifru);

FILE *tok = fopen("drzave.bin", "rb");
fseek(tok, (long)(trazimSifru - 1) * sizeof(drzava), SEEK_SET);
fread(&drzava, sizeof(drzava), 1, tok);

if (drzava.sifDrz == trazimSifru) {
   printf("%s", drzava.naz);
} else {
   printf("Nema drzave s tom sifrom");
}

fclose(tok);
...
```

Primjer

- Programski zadatak
  - svaki zapis datoteke `drzave.bin` sadrži
    - šifru (int), dvoslovnu kraticu (2+1 znak) i naziv države (40+1 znak)
    - redni broj zapisa u datoteci `drzave.bin` odgovara šifri države
  - s tipkovnice učitati niz od dva znaka. Ako u datoteci `drzave.bin` postoji država s kraticom koja odgovara učitanom nizu, na zaslon ispisati šifru i naziv te države. Inače, ispisati poruku "Nema drzave s tom kraticom"

- Analiza
  - ključ potrage ovdje je kratica države i iz nje nije moguće odrediti redni broj zapisa, pa posljedično niti poziciju zapisa u datoteci. Iako neefikasno, jedini način na koji se zadatak može riješiti jest koristiti slijedni pristup
    - redom čitati zapise i uspoređivati sa zadanom kraticom

Rješenje

```
...
struct drz_s {
   int sifDrz;
   char krat[2 + 1];
   char naz[40 + 1];
} drzava;
char trazimKrat[2 + 1];
scanf("%s", trazimKrat);

FILE *tok = fopen("drzave.bin", "rb");
while (fread(&drzava, sizeof(drzava), 1, tok) == 1) {
   if (drzava.sifDrz != 0 && strcmp(drzava.krat, trazimKrat) == 0) {
      break;
   }
}

if (strcmp(drzava.krat, trazimKrat) == 0) {
   printf("%d %s", drzava.sifDrz, drzava.naz);
} else {
   printf("Nema drzave s tom kraticom");
}

fclose(tok);
```

Neispravno rješenje

```
…
…

for (int i = 0; i < 100000; ++i) {
   fseek(tok, (long)(i - 1) * sizeof(drzava), SEEK_SET);
   fread(&drzava, sizeof(drzava), 1, tok);
   if (drzava.sifDrz != 0 && strcmp(drzava.krat, trazimKrat) == 0) {
      break;
   }
}
if (strcmp(drzava.krat, trazimKrat) == 0) {
   printf("%d %s", drzava.sifDrz, drzava.naz);
} else {
   printf("Nema drzave s tom kraticom");
}
fclose(tok);
```

Primjer

- Programski zadatak
  - u tekstnoj datoteci `kupljeno.txt` upisani su podaci o kupljenim artiklima. Zapis datoteke sadrži šifru artikla (4 znamenke) i broj kupljenih komada tog artikla (2 znamenke)

```
1012 12↵
1151 2↵
```

  - zapis binarne datoteke `artikli.bin` 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:

```
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 = NULL, *art = NULL;

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

   short sifArt, kolicina;
   float suma = 0;

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

Rješenje (2. dio)

```
   while (fscanf(kup, "%hd %hd", &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("%-20s %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

- Programski zadatak
  - u tekstnu datoteku `ulaz.txt`, koja se nalazi u mapi `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 tekstnu datoteku `izlaz.txt` u mapi `c:/tmp` prepisati podatke o osobama čije prezime sadrži slovo `r`. Primjer sadržaja datoteke prikazan je ovdje:

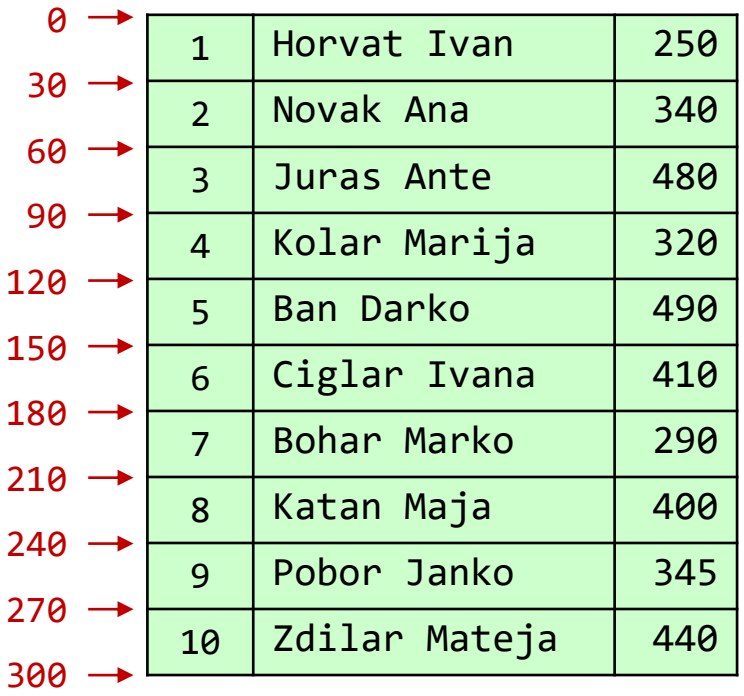
```
101 Vurnek↵
412 Voras↵
```

Rješenje

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

int main(void) {
   int mbr;
   char prez[15 + 1];

   FILE *ulTok = fopen("c:/tmp/ulaz.txt", "r");
   FILE *izTok = fopen("c:/tmp/izlaz.txt", "w");

   while (fscanf(ulTok, "%d %[^\n]", &mbr, prez) == 2) {
      if (strchr(prez, 'r') != NULL) {
         fprintf(izTok, "%d %s\n", mbr, prez);
      }
   }

   fclose (ulTok);
   fclose (izTok);

   return 0;
}
```

Primjer

- Programski zadatak
  - u binarnoj datoteci `bodovi.bin` 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.

Rješenje (1. dio)

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

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

   FILE *dUlIzl = fopen("bodovi", "r+b");  čitanje i pisanje

   srand((unsigned)time(NULL));

   int mbr;
   mbr = rand() % 10 + 1;
```

Rješenje (2. dio)

```
   fseek(dUlIzl, (long)sizeof(zapis) * (mbr - 1), SEEK_SET);
   fread(&zapis, sizeof(zapis), 1, dUlIzl);
   // povecaj broj bodova (ali ne na vise od 500)
   zapis.brBod *= 1.1;
   if (zapis.brBod > 500)
      zapis.brBod = 500;
   // indikator pozicije vrati na pocetak zapisa!
   fseek(dUlIzl, -1L * sizeof(zapis), SEEK_CUR);
   // zapisi sadrzaj cijele strukture u datoteku
   fwrite(&zapis, sizeof(zapis), 1, dUlIzl);
   fclose(dUlIzl);
   return 0;
}
```

Ulazno/izlazni tok

- u prethodnom primjeru isti tok se koristio i za operacije čitanja i za operacije pisanja. Takav tok se naziva *ulazno/izlazni* tok
  - operacije čitanja i pisanja u ulazno/izlaznom toku smiju se koristiti naizmjence, ali
  - kada se u jednom toku nakon operacija čitanja žele početi obavljati operacije pisanja (ili obratno), tada se između operacija čitanja i operacija pisanja mora obaviti barem jedan poziv funkcije `fseek` ili barem jedan poziv funkcije `fflush`

Napomena

Prevodilac gcc na operacijskom sustavu Windows trenutačno po ovom pitanju još nije u potpunosti usklađen sa standardom: između operacija čitanja i pisanja (ili obrnuto) treba obaviti barem jedan poziv funkcije `fseek` (a ne po izboru ili `fseek` ili `fflush`).

fflush <stdio.h>

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

- sadržaj međuspremnika toka zapisuje u sekundarnu memoriju (trajno pohranjuje)
  - radi efikasnog korištenja sekundarne memorije (koja je spora), pisanjem u tok podaci se prvo upisuju u međuspremnik u primarnoj memoriji, a tek zatim (u nekom kasnijem trenutku) se upisuju u sekundarnu memoriju (trajno pohranjuju)
  - to znači: ako program završi prije nego je obavljen `fflush` za neki tok, tada postoji mogućnost da neki od podataka koji jesu upisani u tok možda ipak neće biti zapisani u sekundarnu memoriju (i time će biti izgubljeni)
  - pozivom funkcije `fflush` osigurava se da su svi podaci do tog trenutka upisani u tok, također upisani i u sekundarnu memoriju
  - (funkcija `fclose` automatski obavlja `fflush` prije nego se tok zatvori)