Uvod u programiranje

- predavanja -

siječanj 2021.

23. Datoteke

- 1. dio -

Datoteke

Uvod

Memorija računala

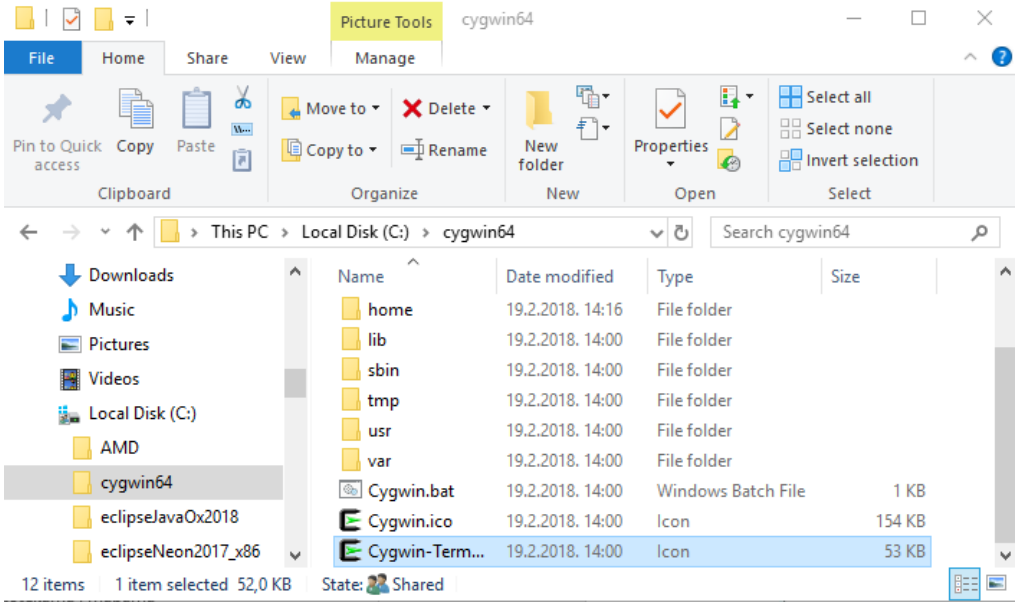
- Primarna
  - privremena (sadržaj se gubi po gubitku napajanja), relativno skupa, manjeg kapaciteta, brža
  - RAM (*Random Access Memory*)
- Sekundarna, tercijarna i off-line
  - trajna (sadržaj ostaje sačuvan po gubitku napajanja), relativno jeftina, većeg kapaciteta, sporija
  - sekundarna memorija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

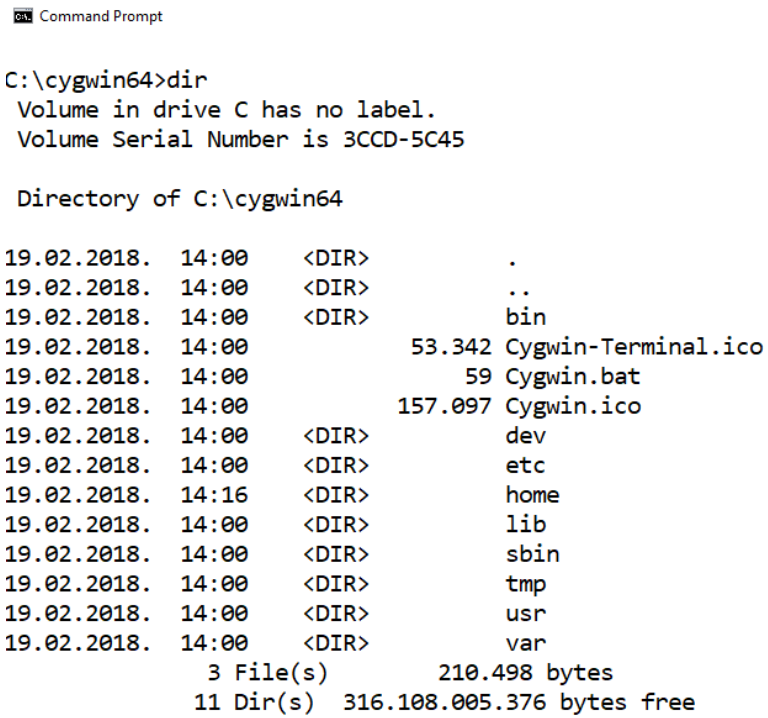
Sekundarna, tercijarna i off-line memorija

- 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*)



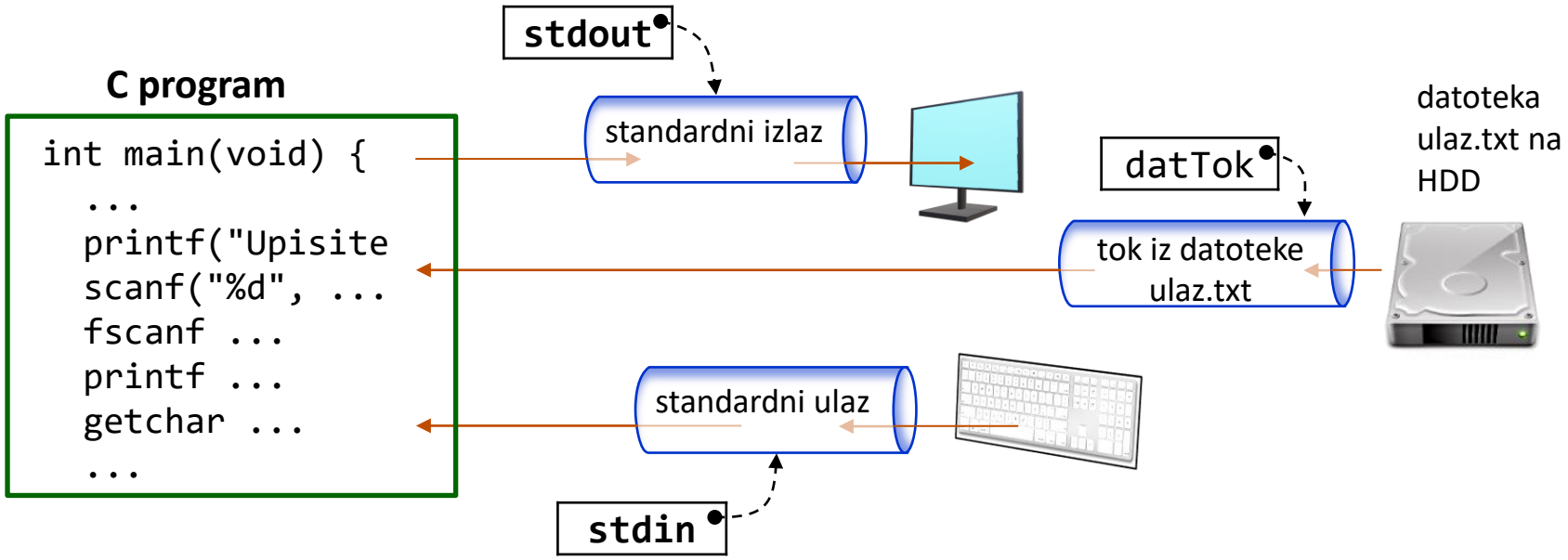


Datoteke i mape

- 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 `stderr`)

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`

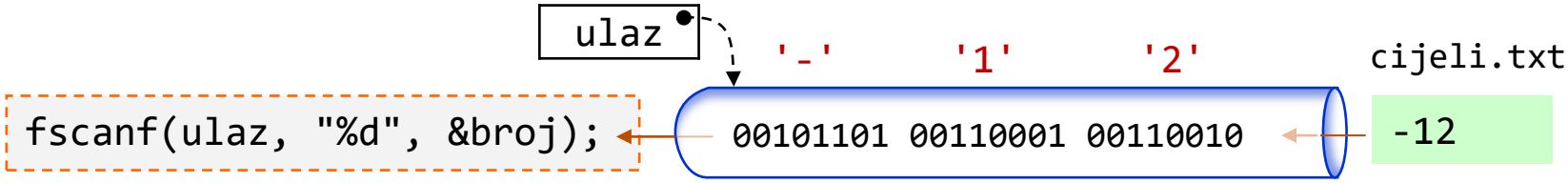
```
 -6.0
  7.5
-1.5
 4.0
 3.5
```

Rješenje

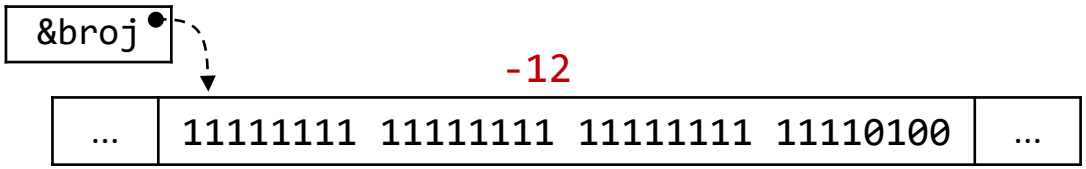
```
#include <stdio.h>

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?



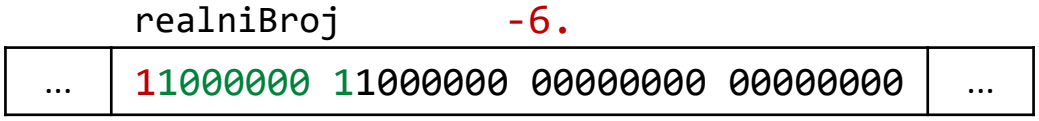
  - obavlja se konverzija pročitanih **znakova** u podatak odgovarajućeg tipa (prema konverzijskoj specifikaciji)
  - rezultat dobiven konverzijom upisuje se na mjesto u memoriji na koje pokazuje argument `&broj`, dakle u varijablu `broj`



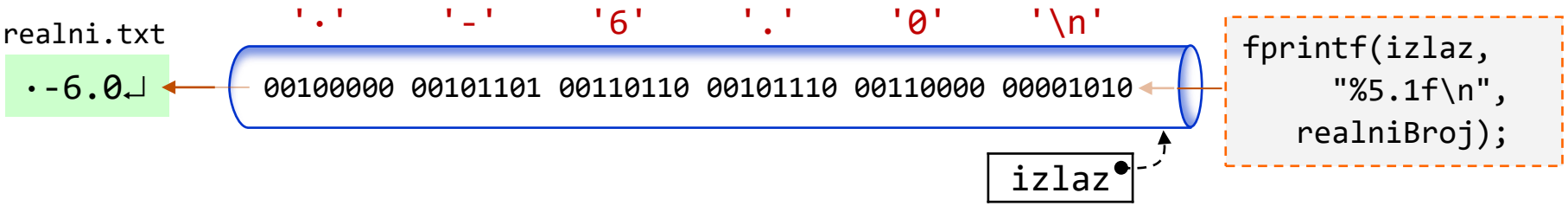
  - s obzirom da je u datoteci pohranjen tekst, da se konverzija u binarni oblik obavlja "prema formatu", ovakve datoteke nazivaju se *tekstne* ili *formatirane* datoteke
    - sadržaj takvih datoteka moguće je pregledavati i uređivati editorom

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`



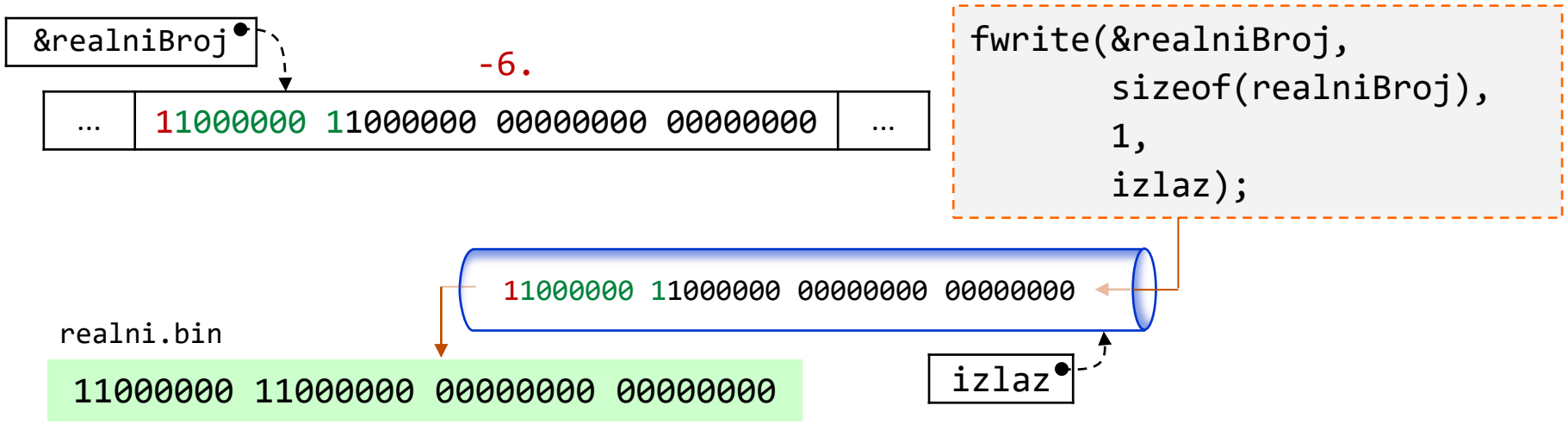
  - rezultat je tekstna (formatirana) datoteka koja sadrži ASCII vrijednosti znakova

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);
```

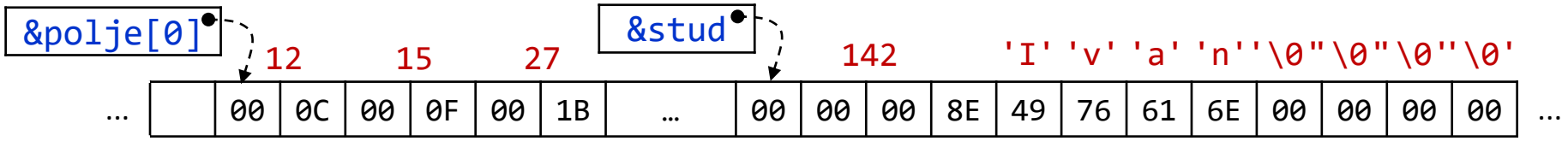
- u tok (u datoteku) na kojeg pokazuje `stream` upisuje se sadržaj memorije na kojeg pokazuje ptr
  - `size`: veličina pojedinačnog objekta koji se upisuje (u bajtovima)
  - `nmemb`: broj objekata koji se upisuje
- ukupna veličina memorije koja se upisuje je veličine `size·nmemb` bajtova
- rezultat funkcije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

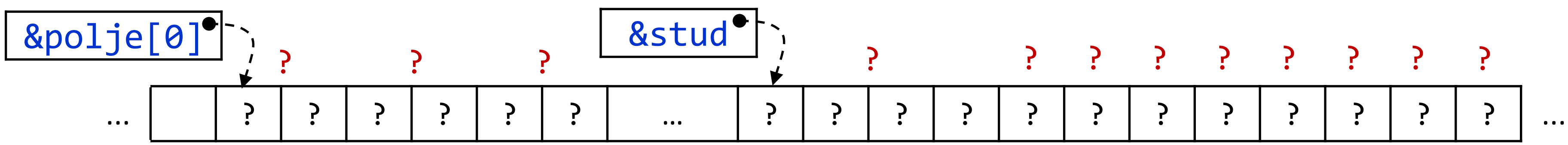
Primjer

podaci.bin

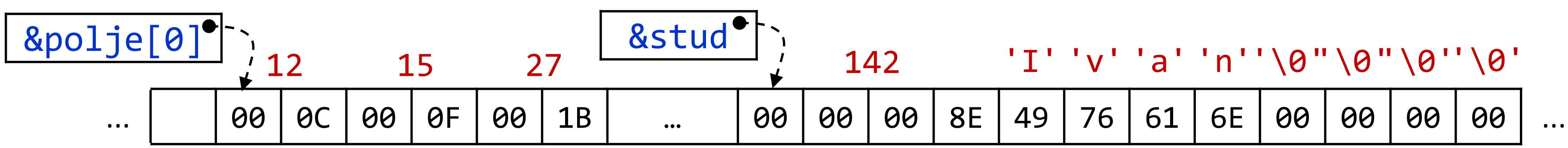
`00 0C 00 0F 00 1B 00 00 00 8E 49 76 61 6E 00 00 00 00`

- 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]

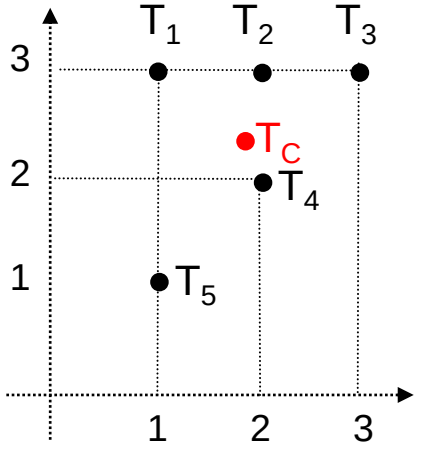
Rješenje

```
...
#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;
}
```