fprintf

Funkcija `fprintf` služi formatiranom ispisu podataka. Opći oblik (prototip) funkcije je

```
int fprintf(FILE *fp, const char *format, ...)
```

Funkcija `fprintf` pretvara (konvertira) i piše izlaz u datoteku `fp` pod kontrolom formata zadanog stringom `format`. Vrijednost koju funkcija vraća je broj ispisanih znakova, ako je ispis obavljen bez greške, ili negativan broj, ako je došlo do greške.

String za formatiranje sadrži dvije vrste objekata:

1. obične znakove, koji se samo kopiraju u izlaznu datoteku, i
2. oznake za konverziju, s tim da svaka od njih radi pretvaranje i ispis sljedećeg po redu argumenta funkcije `fprintf`.

Svaka oznaka za konverziju počinje znakom `%` i završava nekim znakom za konverziju (v. tablicu na kraju opisa funkcije). Između ta dva znaka mogu se nalaziti, redom:

- **Zastavice** (u bilo kojem međusobnom poretku), koje modificiraju specifikacije pretvaranja:
  - `-` označava lijevo pozicioniranje konvertiranog argumenta u polju za ispis.
  - `+` označava da će broj uvijek biti ispisan s predznakom.
  - ␣ (praznina): ako prvi znak (nakon pretvorbe) nije predznak, dodat će se praznina (razmak, blank) na početak.
  - `0` kod numeričkih konverzija, označava dopunjenje polja za ispis do širine polja vodećim nulama.
  - `#` označava alternativnu formu izlaza za pojedine znakove konverzije.
    - Za `o`, prva znamenka bit će nula.
    - Za `x`, odnosno `X`, dodat će znakove `0x`, odnosno `0X`, na početak rezultata različitog od nule. Ako je rezultat nula, neće učiniti ništa.
    - Za `e`, `E`, `f`, `g` i `G`, izlaz će uvijek imati decimalnu točku, a za `g` i `G`, nule na kraju decimalnog broja (koje bi se mogle brisati) će se ispisati.
- **Broj** koji specificira minimalnu širinu polja za ispis, tj. minimalni broj ispisanih znakova. Konvertirani argument bit će ispisan u polju barem te širine, ili širem, ako je potrebno. Ako konvertirani argument ima manje znakova od zadane minimalne širine, onda će on biti pozicioniran desno u polju za ispis, a s lijeve strane bit će dopunjen znakovima za dopunu (osim ako je prisutna zastavica za lijevo pozicioniranje, pa se dopunjuje s desne strane). Standardni znak za dopunjenje do zadane širine polja za ispis je praznina, ali može biti i `0`, ako je postavljena zastavica za ispis vodećih nula.

- **Točka**, koja razdvaja širinu polja za ispis od preciznosti.
- **Broj**, tzv. preciznost, sa sljedećim značenjima, ovisno o vrsti konverzije:
  - Ako se ispisuje string, označava maksimalni broj znakova koji će biti ispisani iz tog stringa.
  - Kod konverzija `e`, `E` ili `f`, označava broj decimala koje će biti ispisane iza decimalne točke.
  - Kod konverzija `g` ili `G`, označava broj značajnih znamenki koje će biti ispisane.
  - Ako se ispisuje cijeli broj, označava minimalni broj znamenki koje će biti ispisane, s tim da će se dodati vodeće nule ako je broj koji se ispisuje "prekratak".
- **Modifikator duljine**, koji može biti `h`, `l` (malo slovo l) ili `L`, a odnosi se na tip (ispisa) vrijednosti sljedećeg argumenta za konverziju:

  `h` označava da argument treba ispisati kao `short` ili `unsigned short`.

  `l` označava da je argument tipa `long` ili `unsigned long`.

  `L` označava da je argument tipa `long double`.

Širina ispisa i/ili preciznost mogu se zamijeniti znakom `*`. U tom slučaju, pripadna vrijednost računa se "dinamički" (u trenutku nailaska na znak `*`) — konvertiranjem prvog sljedećeg nekonvertiranog argumenta, koji mora biti tipa `int`. Dobivena vrijednost "uvrštava" se tog trena umjesto odgovarajućeg znaka `*` u formatu, a zatim se dalje "čita" format. Zato se argumenti za znakove `*` pišu prije argumenta na kojeg se odnosi cijela oznaka konverzije, jer se tim redom radi konverzija argumenata.

Znak konverzije, u načelu, određuje tip vrijednosti argumenta koji se konvertira i način konverzije, a piše se na kraju oznake za konverziju. Ako iza znaka `%` ne slijedi znak za konverziju, ponašanje nije definirano.

U sljedećoj tablici dani su znakovi za konverziju i njihova značenja — osnovni tip argumenta na kojeg se pojedini znak odnosi i u što se taj argument konvertira.

`d`, `i` — `int`;
Dekadski zapis cijelog broja s predznakom.

`o` — `int`;
Oktalni zapis broja bez predznaka i vodeće nule.

`x`, `X` — `int`;
Heksadecimalni zapis broja bez predznaka i vodećih znakova `0x`, odnosno `0X`. Ako je `x` u formatu, slovčane znamenke pišu se malim slovima `abcdef`, a ako je `X` u formatu, onda velikim slovima `ABCDEF`.

`u` — `unsigned int`;
Dekadski zapis cijelog broja bez predznaka.

`c` — `int`;
Jedan jedini znak, nakon konverzije u `unsigned char`.

`s` — `char *`;
Iz stringa se redom ispisuju znakovi sve dok se ne dostigne znak `'\0'`, ili dok broj ispisanih znakova ne dostigne zadanu preciznost.

`f` — `double`;
Decimalni zapis broja oblika $[-]mmm.ddd$, gdje je broj decimala d zadan preciznošću. Ako preciznost nije zadana, uzima se 6 (tzv. default). Ako je preciznost 0, onda se ne ispisuje decimalna točka.

`e`, `E` — `double`;
Decimalni zapis oblika $[-]m.dddddd$ `e`$\pm xx$ ili $[-]m.dddddd$ `E`$\pm xx$, gdje je broj decimala d zadan preciznošću. Ako preciznost nije zadana, uzima se 6 (default). Ako je preciznost 0, onda se ne ispisuje decimalna točka.

`g`, `G` — `double`;
Ako je eksponent manji od -4 ili veći ili jednak preciznosti, koristi se `%e` ili `%E`. U suprotnom, koristi se `%f`. Završne nule iza decimalne točke se ne ispisuju. Decimalna točka se, također, ne ispisuje ako iza nje nema decimalnih znamenki različitih od nule.

`p` — `void *`;
Ispisuje argument kao pokazivač (reprezentacija na izlazu ovisi o implementaciji).

`n` — `int *`;
Broj znakova koji je do tog trenutka ispisan ovim pozivom funkcije piše se (tj. sprema) u pripadni argument. Niti jedan argument se ne konvertira.

`%` — Niti jedan argument se ne konvertira, piše se doslovno znak `%`.

Funkcija `printf` služi formatiranom ispisu podataka na standardnu izlaznu datoteku `stdout`. Prototip funkcije je

```
int printf(const char *format, ...)
```

Poziv `printf(...)` je ekvivalentan pozivu `fprintf(stdout, ...)`.

Funkcija `sprintf` služi formatiranom ispisu podataka u string, a ne u datoteku. Prototip funkcije je

```
int sprintf(char *s, const char *format, ...)
```

Funkcija `sprintf` radi isto što i `printf`, osim što se izlaz piše u string `s` i završava znakom `'\0'`. String `s` mora biti dovoljno dugačak da cijeli rezultat stane u njega. Funkcija vraća broj ispisanih znakova, bez završnog znaka `'\0'`.

fscanf

Funkcija `fscanf` služi formatiranom čitanju podataka. Prototip funkcije je

```
int fscanf(FILE *fp, const char *format, ...)
```

Funkcija `fscanf` čita iz datoteke `fp` pod kontrolom formata zadanog stringom `format` i dodjeljuje konvertirane vrijednosti odgovarajućim argumentima, redom kako slijede iza formata. Svaki od tih argumenata **mora** biti pokazivač (na vrijednost odgovarajućeg tipa).

Funkcija `fscanf` normalno završava posao kad je iscrpila cijeli format. Čitanje može završiti i ranije, ako se prilikom čitanja naiđe na kraj datoteke ili dođe do greške. Funkcija vraća `EOF` ako je **prije prve** konverzije došlo do kraja datoteke ili do greške. U protivnom, funkcija vraća **nenegativan** broj ulaznih cjelina znakova (tzv. polja) koje su uspješno konvertirane u vrijednosti i dodjeljene argumentima (tj. broj dodjeljenih vrijednosti). Taj broj može biti i 0, ako sljedeći znak na ulazu **ne** odgovara **prvoj** specifikaciji u format stringu (na primjer, sljedeći znak na ulazu je zarez, a čita se cijeli broj). Sljedeći poziv funckije `fscanf` nastavlja čitanje odmah nakon zadnjeg ranije konvertiranog znaka.

Format string uobičajeno sadrži oznake (specifikacije) za konverziju, koje služe za upravljanje interpretacijom znakova na ulazu. Preciznije, string za formatiranje može sadržavati:

- Praznine ili tab-ove, koji se (uglavnom) ignoriraju, a stvarno služe za preskakanje bjelina do prve ne-bjeline, tamo gdje se to **ne** radi automatski (na pr. kod običnih znakova ili `%c` konverzija).
- Obične znakove (različite od `%`). Za svaki takav znak se očekuje da se doslovno podudara sa sljedećim znakom iz ulazne datoteke (koji se onda i učitava). U protivnom, ako nema podudaranja, smatra se da je došlo do greške prilikom čitanja i čitanje se prekida. Ovo odgovara doslovnom čitanju znakova, a slično je doslovnom pisanju kod `fprintf`.
- Oznake za konverziju, koje počinju znakom `%`, a završavaju nekim znakom za konverziju (v. tablicu na kraju opisa funkcije). Između ta dva znaka mogu (ali, ne moraju) se nalaziti, redom:
  - Znak `*`, koji **zabranjuje** dodjeljivanje konvertirane vrijednosti sljedećem argumentu.
  - **Broj** koji označava maksimalnu širinu ulaznog polja za čitanje.

– Znakovi `h`, `l` ili `L`, koji zadaju veličinu tipa za konvertiranu vrijednost (v. malo dalje za preciznije značenje).

Oznaka konverzije određuje način konverzije sljedećeg (još nepročitanog) polja na ulazu. Nakon čitanja i konverzije, obično se rezultat sprema u varijablu na koju pokazuje odgovarajući argument — to je prvi sljedeći kojem još nije dodjeljena vrijednost. Međutim, ako je dodjeljivanje zabranjeno znakom `*`, kao u `%*s`, onda se pripadno ulazno polje naprosto preskače, bez dodjeljivanja rezultata.

Prilikom čitanja, **bjelinom** se smatraju sljedeći znakovi: praznina (blank), horizontalni tabulator `'\t'`, znak za prijelaz u novi red `'\n'`, znak za pomak na početak reda `'\r'` (carriage return), vertikalni tabulator `'\v'` i znak za prijelaz na novu stranicu `'\f'` (formfeed).

Kod većine konverzija, prvo se automatski **preskaču** sve bjeline na početku do tada nepročitanog ulaza, i tek prva sljedeća ne-bjelina započinje ulazno polje za konverziju. To vrijedi kod svih numeričkih konverzija i čitanja stringova (znak `s`), a izuzetak su samo čitanje znakova (znak `c`) i "filtrirano" čitanje stringova (znak `[` kao prvi znak za konverziju). Preskakanje početnih bjelina može se dobiti i u tim slučajevima, navođenjem praznine (ili tab-a) u formatu, ispred odgovarajuće oznake za konverziju. Isto vrijedi i za obične znakove u formatu.

Preskakanje vodećih bjelina povlači da će `fscanf` čitati i preko kraja reda da nađe svoj ulaz, jer se znak `'\n'` smatra bjelinom.

Ulazno polje za konverziju, u principu, počinje prvim sljedećim nepročitanim znakom, nakon eventualnog preskakanja vodećih bjelina.

Nakon toga se, ovisno o znaku za konverziju, čita (i, po potrebi, konvertira) **najdulji** mogući niz znakova koji odgovara dozvoljenom obliku ulaza za taj znak konverzije, sve do maksimalne širine polja, ako je zadana, ili do prvog znaka koji **nije** dozvoljen u ulaznom polju (taj znak se onda **ne** učitava, već ostaje kao prvi sljedeći nepročitani znak). To znači da kod svih numeričkih konverzija i čitanja stringova (znak `s`), ulazno polje ide najdalje do znaka ispred **prve** sljedeće bjeline.

Znak za konverziju zadaje interpretaciju ulaznog polja i dozvoljeni oblik ulaza. Pripadni argument mora biti **pokazivač** na odgovarajući tip. U sljedećoj tablici dani su znakovi za konverziju i njihova značenja — osnovni tip argumenta na kojeg se pojedini znak odnosi i što se (s ulaza) konvertira u taj argument.

`d` — `int *`;
Dekadski zapis cijelog broja (dozvoljen predznak).

`i` — `int *`;
Cijeli broj u zapisu kao cjelobrojna konstanta u C-u. Zapis je oktalni, ako je vodeća znamenka `0`, ili heksadecimalni, ako su vodeći znakovi `0x` ili `0X`. U protivnom, zapis je dekadski.

`o` — `int *`;
Oktalni zapis cijelog broja (sa ili bez vodeće nule).

`u` — `unsigned int *`;
Dekadski zapis cijelog broja bez predznaka.

`x` — `int *`;
Heksadecimalni zapis cijelog broja (sa ili bez vodećih znakova `0x` ili `0X`).

`c` — `char *`;
Čitaju se sljedeći znakovi na ulazu i redom stavljaju u zadano polje, do broja zadanog maksimalnom širinom ulaznog polja. Ako maksimalna širina ulaznog polja nije zadana, uzima se širina 1 (default), tj. čita se jedan znak. Na kraju polja se **ne dodaje** znak `'\0'`. Ovdje **nema** uobičajenog preskakanja vodećih bjelina. Ako se želi pročitati sljedeća ne-bjelina, može se koristiti `%1s` (ali to dodaje i znak `'\0'` na kraj stringa). Bolje je dodati prazninu ispred oznake konverzije `%c`.

`s` — `char *`;
Čita se niz znakova (string) sastavljen od ne-bjelina i redom sprema u zadano polje. String se piše bez navodnika, tj. svi navodnici su dio stringa. Na kraj učitanog stringa dodaje se znak `'\0'`. Polje na koje pokazuje argument mora biti dovoljno dugo za cijeli učitani niz i završni znak `'\0'`.

`e`, `f`, `g` — `float *`;
Dekadski zapis realnog broja (u floating–point prikazu). Dozvoljeni oblik zapisa na ulazu je: znak za predznak (kojeg ne mora biti), niz znamenki koji može (ali ne mora) sadržavati decimalnu točku, i polje za eksponent (kojeg ne mora biti), a sadrži znak `e` ili `E`, iza kojeg slijedi cijeli broj (koji smije imati predznak).

`p` — `void *`;
Vrijednost pokazivača — onako kako bi ju ispisao `printf("%p")`.

`n` — `int *`;
Broj znakova koji je do tog trenutka pročitan ovim pozivom funkcije piše se (tj. sprema) u pripadni argument. Ništa se ne čita na ulazu. Broj konvertiranih vrijednosti se ne povećava.

`[...]` — `char *`;
Učitava najdulji neprazan string ulaznih znakova koji se **nalaze** u skupu znakova navedenom između uglatih zagrada. Taj skup smije sadržavati prazninu i ostale posebne znakove. Na kraj učitanog stringa dodaje se znak `'\0'`. Oznaka `[]...]` uključuje znak `]` u skup. Ovdje **nema** uobičajenog preskakanja vodećih bjelina.

`[^...]` — `char *`;
Učitava najdulji neprazan string ulaznih znakova koji se **ne nalaze** u skupu znakova navedenom između uglatih zagrada. Na kraj učitanog stringa dodaje se znak `'\0'`. Oznaka `[^]...]` uključuje znak `]` u skup. Ovdje **nema** uobičajenog preskakanja vodećih bjelina.

`%` — Doslovno obični znak `%` u formatu (tj. očekuje se znak `%` na ulazu). Nema konverzije i nema dodjeljivanja vrijednosti argumentima.

Kod svih numeričkih konverzija **mora** se korektno zadati stvarna veličina tipa za konvertiranu vrijednost — prema tipu na kojeg pokazuje odgovarajući argument. Zadaje se znakovima `h`, `l` (malo slovo l) ili `L` u oznaci za konverziju, ispred znaka konverzije. Dozvoljene modifikacije tipa su:

- Kod konverzija `d`, `i`, `n`, `o`, `u` i `x`, ako argument **nije** pokazivač na `int` (ili `unsigned int` za `u`), onda ispred znaka mora stajati:
  - `h` — ako je argument pokazivač na odgovarajući `short` tip.
  - `l` — ako je argument pokazivač na odgovarajući `long` tip.
- Kod konverzija `e`, `f` i `g`, ako argument **nije** pokazivač na `float`, onda ispred znaka mora stajati:
  - `l` — ako je argument pokazivač na `double`.
  - `L` — ako je argument pokazivač na `long double`.

Tipična greška je čitanje po `f` (za `float`) i spremanje u `double`, a korektno čitanje i spremanje **mora** biti po `lf`.

Funkcija `scanf` služi formatiranom čitanju podataka iz standardne ulazne datoteke `stdin`. Prototip funkcije je

```
int scanf(const char *format, ...)
```

Poziv `scanf(...)` je ekvivalentan pozivu `fscanf(stdin, ...)`.

Funkcija `sscanf` služi formatiranom čitanju podataka iz stringa, a ne iz datoteke. Prototip funkcije je

```
int sscanf(char *s, const char *format, ...)
```

Poziv `sscanf(s, ...)` radi isto što i `scanf(...)`, osim što se ulazni znakovi čitaju iz stringa `s`. Znak `'\0'` za kraj stringa ovdje ima istu ulogu kao i kraj datoteke kod `scanf`.