Programiranje 2
11. predavanje

Saša Singer

singer@math.hr
web.math.hr/~singer

PMF – Matematički odsjek, Zagreb

Sadržaj predavanja

- Ulaz i izlaz podataka:
  - Funkcija `scanf`.
  - Funkcija `printf`.

Informacije

Konzultacije (službeno):

- petak, 12–14 sati, ili — po dogovoru.

Programiranje 2 je u kolokvijskom razredu C3.

- Drugi kolokvij: petak, 1. 6., u 15 sati,
- Popravni kolokvij: petak, 15. 6., u 15 sati.

Uputa: "izbjegnite" popravni — obavite to ranije!

Ne zaboravite, "žive" su i domaće zadaće na adresi

`http://degiorgi.math.hr/prog2/ku/`

Dodatni bodovi "čekaju na vas".

Informacije — podsjetnik na kolokvijima

Napomena o podsjetniku = "šalabahteru" na službenom webu:

- nemojte ga nositi na kolokvij — to je zabranjeno!
- Dobit ćete ga na kolokviju.

Razlog: Nećemo "obrađene" šalce — s "osobnim dodacima".

Informacije — ulaz i izlaz

Na mom webu, pod dodatnim materijalima za Prog1 i Prog2, nalazi se tekst

- `in_out.pdf` (7 stranica, 59 kB),

koji sadrži detaljan opis funkcija

- za formatirani ulaz i izlaz podataka.

Najveći dio teksta govori o funkcijama `fprintf` i `fscanf`. U imenima ovih funkcija,

- prvo slovo `f` dolazi od riječi "file" (datoteka), a
- zadnje slovo `f` dolazi od "formatted" (formatirani).

Ove funkcije, u principu, rade za bilo koju datoteku, a

- izlazna ili ulazna datoteka se zadaje kao argument.

Informacije — ulaz i izlaz (nastavak)

Funkcije `printf` i `scanf` (bez prvog slova `f`)

- rade na standardnim datotekama za izlaz, odnosno ulaz,

i zato se datoteka ne zadaje. To je jedina razlika!

Veza između osnovne funkcije (s prvim slovom `f`) i funkcije za standardnu datoteku dana je na kraju opisa osnovne funkcije.

Osnova za tekst je

- Dodatak B iz knjige KR2.

Tamo je opis svih funkcija iz standardne C biblioteke.

Međutim, "prijevod" nije doslovan. Neki dijelovi su prošireni i

- popravljen je opis `fscanf` (original nije skroz korektan).

Lijepo molim, ako uočite "tipfelere" — javite mi!

Formati za funkciju `scanf`

Funkcija `scanf`

Funkcija `scanf` služi formatiranom učitavanju podataka sa standardnog ulaza (`stdin`). Opća forma poziva funkcije je

```
scanf(kontrolni_string, arg_1,
                arg_2, ..., arg_n)
```

Prvi argument `kontrolni_string` je konstantni znakovni niz (string) koji sadrži informacije o vrijednostima koje se učitavaju u argumente `arg_1`, ..., `arg_n`.

Sastoji se od "običnih" znakova i posebnih grupa znakova, tzv. specifikacija ili oznaka konverzije (ili pretvaranja).

Svaka oznaka konverzije pridružena je po jednom sljedećem argumentu — onim redom, kako pišu.

Funkcija `scanf` *(nastavak)*

Svaka oznaka (grupa znakova) konverzije započinje znakom postotka (`%`), a na kraju dolazi `znak konverzije` koji upućuje na tip podatka koji se učitava. Na primjer, `%c` ili `%d`.

Najčešće korišteni znakovi konverzije su:

`znak konverzije`	`tip podatka koji se učitava`
`d`	decimalni cijeli broj (`int`)
`i`	decimalni, heksadecimalni ili oktalni cijeli broj (`int`)
`u`	cijeli broj bez predznaka (`unsigned int`)
⋮	⋮

Funkcija `scanf` *(nastavak)*

znak konverzije	tip podatka koji se učitava
⋮	⋮
`o`	oktalni cijeli broj (`int`)
`x`	heksadecimalni cijeli broj (`int`)
`e, f, g`	broj s pomičnom točkom (`float`)
`c`	jedan znak (`char`)
`s`	string (`char *`)
`p`	pokazivač (`void *`)

Ispred znaka konverzije može doći modifikator duljine tipa:

- `h` — skraćuje tip (`h` = "half"),
- `l` ili `L` — "malo" ili "jako" produljuje tip (`l` = "long").

Učitavanje cijelih brojeva

Cijeli brojevi mogu biti učitani kao decimalni (`%d`), ili kao decimalni, oktalni i heksadecimalni (`%i`). Znak konverzije `i` interpretira ulazni podatak kao oktalni broj ako mu prethodi nula, a kao heksadecimalan broj ako mu prethodi `0x` ili `0X`.

Primjer. Ako komad programa

```
int x, y, z;
...
scanf("%i %i %i", &x, &y, &z);
```

učitava ulaznu liniju:

```
13  015    0Xd
```

onda je u `x`, `y` i `z` učitana ista vrijednost 13 (decimalno).

Učitavanje cijelih brojeva (nastavak)

Cijeli brojevi u oktalnom i heksadecimalnom zapisu mogu se čitati i pomoću znakova konverzije `o`, odnosno, `x`. Ti znakovi konverzije ne zahtijevaju da oktalna konstanta započinje nulom, a heksadecimalna s `0x` ili `0X`.

Primjer.

```
int x, y, z;
...
scanf("%d %o %x", &x, &y, &z);
```

ispravno čita ulazne podatke:

```
13  15  d
```

i svim varijablama pridružuje vrijednost 13 (decimalno).

Učitavanje cijelih brojeva (nastavak)

Podatak tipa `unsigned` učitavamo znakom konverzije `u`.

Znakovi konverzije `d`, `i`, `o`, `u`, `x` mogu (moraju) dobiti

- prefiks `h` — ako je argument pokazivač na `short`,
- prefiks `l` — ako je argument pokazivač na `long`.

Primjer.

```
int x;
short y;
long z;
...
scanf("%d %hd %ld", &x, &y, &z);
```

učitava tri decimalna cijela broja i sprema ih u varijable tipa `int`, `short` i `long`.

Učitavanje realnih brojeva

Znakovi konverzije `e`, `f` i `g` služe za učitavanje varijable tipa `float`. Ako se učitava vrijednost u varijablu tipa `double` treba koristiti prefiks `l` (`le`, `lf` ili `lg`).

Primjer.

```
float x;
double y;
...
scanf("%f %lg", &x, &y);
```

Prefiks `L` koristi se učitavanje realne vrijednosti u varijablu tipa `long double` (ako postoji).

Maksimalna širina ulaznog polja

Uz svaki znak konverzije može se zadati i maksimalna širina ulaznog polja koje će se učitati — tako da se ispred znaka konverzije stavi broj koji zadaje tu maksimalnu širinu polja.

Primjer.

`%3d` učitava cijeli broj s najviše tri znamenke.

`%11s` učitava najviše 11 znakova stringa (bitno u praksi).

- Ako podatak sadrži manje dozvoljenih znakova od zadane maksimalne širine polja, čitanje staje ispred prvog nedozvoljenog znaka (na pr. bjelina — za brojeve).
- Ako "podatak" ima više (>) dozvoljenih znakova od zadanog, pripadno polje znakova za konverziju "skraćuje" se na zadani broj — "višak" znakova ostaje za naknadno čitanje.

Razmaci (praznine) u kontrolnom stringu

Oznake konverzije mogu biti odijeljene razmacima:

```
scanf("%f %d", &x, &i);
```

Taj razmak ("praznina") ima za posljedicu preskakanje svih bjelina na ulazu — do početka novog ulaznog polja.

Kod čitanja vrijednosti brojevnih tipova i stringova,

- eventualne bjeline ispred polja se automatski preskaču.

U kontrolnom stringu, pripadne oznake konverzije možemo pisati razdvojeno razmacima (kao u primjeru `"%f %d"`), ili nerazdvojeno (kao `"%f%d"`). Oba zapisa rade jednako.

To ne vrijedi za znakove konverzije `c` i `[`. Tamo razmak ispred oznake ima značenje (v. malo kasnije).

Drugi znakovi u kontrolnom stringu

U kontrolnom stringu, osim razmaka i oznaka konverzije, mogu se pojaviti i drugi znakovi. Njima moraju odgovarati posve isti znakovi na ulazu — vrši se "sparivanje".

Primjer. Ako realan i cijeli broj učitavamo naredbom

```
scanf("%f,%d", &x, &i);
```

onda ulazni podaci moraju biti oblika, na pr.

```
1.456, 8
```

bez bjeline između prvog broja i zareza.

Tek sljedeća oznaka konverzije `%d` preskače sve eventualne bjeline na ulazu ispred "svog polja" (drugog broja).

Formatiranje i konverzija ulaza (nastavak)

Ako se želi dozvoliti bjelina prije zareza, potrebno je koristiti naredbu

```
scanf("%f ,%d", &x, &i);
```

Razmak nakon oznake `%f` preskače sve eventualne bjeline na ulazu ispred zareza.

Dakle, za razmak u kontrolnom stringu, također, vrijedi "sparivanje". Za razliku od ostalih "običnih" znakova, s tim razmakom se na ulazu

- sparuje bilo koji niz bjelina (onih 6 dozvoljenih znakova), s tim da taj niz smije biti i prazan.

Učitavanje znakovnih nizova — %s

Znak konverzije `s` učitava niz znakova (string). Vodeće bjeline na ulazu se preskaču. Niz završava (najdalje) ispred prve sljedeće bjeline u ulaznom nizu znakova. Iza posljednjeg učitanog znaka, u string se automatski dodaje nul-znak (`\0`).

Primjer. Čitanje jedne "riječi" (bez bjelina) i jednog broja

```
char string[128];
int x;
...
scanf("%s %d", string, &x);
```

Ime polja je sinonim za pokazivač na prvi element polja.

Zato se ispred varijable `string` ne stavlja adresni operator.

Učitavanje znakovnih nizova — `%[...]`

Oznakom konverzije `%s` nije moguće učitati niz znakova koji sadrži bjeline, jer bjeline služe kao oznaka za kraj polja.

Za učitavanje nizova znakova koji uključuju i bjeline koristimo uglate zagrade kao znak konverzije — oznaka je `%[...]`.

- Unutar uglatih zagrada upisuje se niz znakova.
- Funkcija `scanf` će, u pripadni argument, učitati najdulji niz znakova s ulaza koji se sastoji samo od znakova navedenih unutar uglatih zagrada.
- Učitavanje završava ispred prvog znaka na ulazu koji nije naveden u uglatim zagradama. Na kraj učitanog niza dodaje se nul-znak (`\0`).
- Vodeće bjeline se ne preskaču.

Učitavanje znakovnih nizova — %[...]

Primjer. Naredba

```
char linija[128];
...
scanf(" %[ ABCDEFGHIJKLMNOPRSTUVWXYZ]", linija);
```

učitava najdulji niz znakova sastavljen samo od velikih slova i razmaka.

- Prije `%[` ostavljen je jedan razmak koji govori funkciji `scanf` da preskoči sve bjeline ispred znakovnog niza.
- To je nužno ako smo prije imali poziv `scanf` funkcije, koji nije učitao bjelinu kojom završava prethodno polje u ulaznom nizu (na pr., završni znak za prijelaz u novi red).

Učitavanje znakovnih nizova — %[...]

Primjer. Naredba bez tog razmaka na početku

```
scanf("%[ ABCDEFGHIJKLMNOPRSTUVWXYZ]", linija);
```

započela bi čitanje na tom znaku za prijelaz u novi red (`\n`).

- Budući da on nije naveden unutar uglatih zagrada, odmah bi završila čitanje ulaznih podataka.

Broj učitanih znakova bio bi nula!

Posljedica. Željena `linija` ne bi bila učitana, tj. dobili bismo prazan string — koji sadrži samo `\0`.

Učitavanje znakovnih nizova — %[^...]

S uglatim zagradama možemo koristiti i sintaksu "negacije"

```
scanf(" %[^niz znakova]", linija);
```

Sada se u odgovarajući argument učitava najdulji mogući niz znakova sastavljen od bilo kojih znakova — osim onih koji se nalaze u uglatim zagradama iza znaka ^.

Primjer. Cijelu liniju bez znaka za prijelaz u novi red (\n) možemo učitati naredbom

```
scanf(" %[^\n]", linija);
```

Na kraj učitanog niza znakova bit će dodan \0, a ispred %[namjerno je ostavljen razmak, zato da se preskoče sve prethodne bjeline.

Opasnost kod čitanja znakovnih nizova

Kod formatiranog čitanja stringova postoji ista opasnost kao i kod funkcije `gets`. Ako ne navedemo maksimalnu širinu polja,

- može doći do "prepunjenja" stringa.

Zato uvijek treba navesti maksimalnu širinu polja,

- tako da svi učitani znakovi stanu u string, zajedno s `\0` koji se dodaje na kraj stringa.

Primjer.

```
char str_1[16], str_2[33], str_3[80];
...
scanf("%15s", str_1);
scanf("%32[A-Z]", str_2);
scanf("%79[^\n]", str_3);
```

Učitavanje pojedinačnih znakova

Znak konverzije `c` učitava jedan znak u varijablu, bez obzira je li on bjelina ili ne (nema preskakanja bjelina).

- Ako želimo preskakanje bjelina, na pr. zato da preskočimo znak za prijelaz u novi red koji je ostao nakon prethodnog poziva funkcije `scanf`, treba staviti jedan razmak ispred oznake konverzije `%c`.
- Kontrolni niz `" %c%c%c"` — učitava vrijednosti tri znaka. Prvo preskače sve bjeline (zbog razmaka ispred prve `%c` oznake), a zatim čita tri uzastopna znaka.

  Dakle, prvi učitani znak nije bjelina, a preostala dva mogu biti bjeline.
- Ako želimo čitati samo znakove različite od bjelina, treba koristiti `" %c %c %c"`, ili `%c` zamijeniti s `%1s`.

Prefiks *

Neki podatak u ulaznom nizu moguće je preskočiti i ne pridružiti ga odgovarajućoj varijabli. To se radi tako da se znaku konverzije doda prefiks `*` — odmah iza znaka `%`.

Primjer.

```
scanf(" %s %*d %d", ime, &n);
```

korektno čita drugi podatak po oznaci `%d`. Zbog prefiksa `*`, neće se izvršiti pridruživanje te vrijednosti varijabli `n`. Taj podatak, tj. pripadno polje znakova se preskače (zanemaruje).

Treći podatak bit će normalno pridružen varijabli `n`.

Svrha: Preskakanje "kolona" u tablicama!

Primjer 1 za `scanf`

Primjer. Dio programa koji čita i piše podatke (v. `p_sc_04.c`)

```
double x;  char c;  int i;

scanf("%lg%c%d", &x, &c, &i);

printf("x = %g, c = '%c', i = %d\n", x, c, i);
```

Za ulaz:

```
17.19x17
```

dobivamo izlaz:

```
x = 17.19, c = 'x', i = 17
```

Primjer 2 za `scanf`

Primjer. Dio programa koji čita i piše podatke (v. `p_sc_05.c`)

```
int i;  float x;  char s[50];

scanf("%2d%f%*d %[0-9]", &i, &x, s);

printf("i = %d, x = %f, s = %s\n", i, x, s);
```

Za ulaz:

```
56789 0123 56a72
```

dobivamo izlaz:

```
i = 56, x = 789.000000, s = 56
```

Formati za funkciju `printf`

Funkcija `printf`

Funkcija `printf` služi za formatirani ispis podataka na standardnom izlazu (`stdout`). Opća forma poziva funkcije je

```
printf(kontrolni_string, arg_1,
                arg_2, ..., arg_n)
```

Prvi argument `kontrolni_string` je konstantni znakovni niz (string) koji sadrži informaciju o formatiranju ispisa vrijednosti argumenata `arg_1`, ..., `arg_n`.

Kontrolni string (ili "format–string") ima posve istu formu i vrlo sličnu funkciju kao kod funkcije `scanf`.

Ostali argumenti `arg_1`, ..., `arg_n` su, općenito, izrazi.

Funkcija `printf` *(nastavak)*

Najčešće korišteni znakovi konverzije su:

znak konverzije	tip podatka koji se ispisuje
`d, i`	decimalni cijeli broj (`int`)
`u`	decimalni cijeli broj bez predznaka (`unsigned int`)
`o`	oktalni cijeli broj (`int`)
`x, X`	heksadecimalni cijeli broj (`int`)
`e, f, g`	broj s pomičnom točkom (`double`)
`c`	jedan znak (`char`)
`s`	string (`char *`)
`p`	pokazivač (`void *`)
`%`	nema konverzije, ispiši znak `%`

Funkcija `printf` — *primjer*

Primjer. Dio programa (v. `p_pr_00.c`)

```
int n = 13;
printf("%%10d\n", n);
```

ispisuje izlaz od jednog reda teksta:

```
%10d
```

Razlog: `%%` nije "prava" oznaka konverzije, već

- "nalog" za doslovni ispis jednog znaka `%`.

Dakle, cijeli format-string se "doslovno" ispisuje (nema oznaka konverzija). I još dobijem upozorenje od prevoditelja da

- format–string završava prije argumenta `n`.

Konverzije tipova kod `printf`

Pri pozivu funkcije `printf` može doći do konverzije tipova:

- Argumenti tipa `float` uvijek se pretvaraju u `double`.
- Argumenti tipa `char` i `short` mogu biti pretvoreni u tip `int`, ako tako piše u oznaci konverzije (primjeri slijede).

Zbog toga, znak konverzije:

- `%f` — ispisuje vrijednosti tipa `float` i `double`,
- `%d` — može ispisati vrijednosti tipa `int`, `char` i `short`.

Slično vrijedi i za ostale "realne", odnosno, "cjelobrojne" znakove konverzije.

Zato, oprez s tipovima.

- Nemojte ignorirati upozorenja prevoditelja, jer ne mora raditi dobro.

Ispis znaka

Jedan znak možemo ispisati na dva načina:

- kao "običan" znak — `%c`, i
- kao cijeli broj — `%d` (uz pretvaranje tipova).

Primjer. Dio programa (v. `p_pr_02.c`)

```
char c = '1';
printf("c(char) = %c, c(int) = %d\n", c, c);
```

ispisuje

```
c(char) = 1, c(int) = 49
```

ako računalo koristi ASCII skup znakova — broj `49` je ASCII kôd znaka `'1'`.

Oktalni i heksadecimalni ispis

Pomoću znakova konverzije `%o` i `%x` (ili `%X`), ispisuju se cijeli brojevi u oktalnom i heksadecimalnom obliku, i to:

- bez predznaka i bez vodeće nule, odnosno, `0x` (ili `0X`).

Ako želimo da za broj različit od nule

- oktalni ispis ima vodeći znak `0`, odnosno,
- heksadecimalni ispis ima vodeće znakove `0x` (ili `0X`),

onda treba koristiti tzv. alternativnu formu ispisa.

- Dobiva se "zastavicom" (engl. "flag") `#`, koju treba napisati odmah iza znaka `%`.

Oktalni i heksadecimalni ispis — primjer

Primjer. Dio programa (v. p_pr_03.c)

```
int i = 64;

printf("i(dec) = %d  = %i\n", i, i);
printf("i(oct) = %o  = %#o\n", i, i);
printf("i(hex) = %x  = %#x\n", i, i);
```

ispisuje

```
i(dec) = 64  = 64
i(oct) = 100  = 0100
i(hex) = 40  = 0x40
```

Oktalni i heksadecimalni ispis — primjer (nast.)

Primjer. Ako istu stvar (bez `%i` i `#`) napravimo za `i = -3` (v. `p_pr_04.c`), dobivamo

- `i(dec) = -3`,
- `i(oct) = 37777777775`,
- `i(hex) = fffffffd`.

Objašnjenje: sadržaj lokacije `i` na kojoj je spremljen `-3` konvertira se u oktalni, odnosno, heksadecimalni zapis, ali bez predznaka.

Ispis je isti kao da tu lokaciju

- interpretiramo po bitovima ("binarno"),

odnosno, kao cijeli broj bez predznaka.

Modifikatori tipa za `short` *i* `long`

Promjena duljine osnovnog tipa zadaje se

- modifikatorom tipa u odgovarajućoj oznaci konverzije,

koji se piše ispred znaka konverzije (tj. kao prefiks).

Za cjelobrojne tipove modifikatori tipa su:

- `h` — označava da je argument tipa `short` ili `unsigned short`. Ako ga ne napišemo, dolazi do pretvaranja tipa u `int` ili `unsigned int`.
- `l` — označava da je argument tipa `long` ili `unsigned long`. Ovdje nema pretvaranja tipova, tj. treba napisati modifikator tipa (osim ako su `int` i `long` isti, pa stvar radi slučajno).

Na nekim sustavima postoji i `ll` za `long long` (kad ga ima).

Ispis realnih brojeva

Brojeve tipa `float` i `double` možemo ispisivati pomoću znakova konverzije `%f`, `%g` i `%e`.

- `%f` — broj se ispisuje bez eksponenta.
- `%e` — broj se ispisuje s eksponentom.
- `%g` — način ispisa (s eksponentom ili bez njega) ovisi o vrijednosti koja se ispisuje.
  - Ako je eksponent manji od -4 ili dovoljno velik, koristi se `%e`. U protivnom, koristi se `%f`.
  - Završne nule iza decimalne točke se ne ispisuju.

Za ispis brojeva tipa `long double` (ako postoje) koristimo prefiks (modifikator duljine) `L`.

- Pripadne specifikacije konverzije su `%Le`, `%Lf`, `%Lg`.

Ispis realnih brojeva (nastavak)

Primjer. Dio programa (v. p_pr_07.c)

```
double x = 12345.678;

printf("x(f) = %f\n", x);
printf("x(e) = %e\n", x);
printf("x(g) = %g\n", x);
```

ispisuje

```
x(f) = 12345.678000
x(e) = 1.234568e+004
x(g) = 12345.7
```

Za %f i %e imamo 6 decimala, a %g daje 6 vodećih znamenki.

Minimalna širina ispisa

Uz svaki znak konverzije moguće je zadati minimalnu širinu ispisa, tj. minimalni broj znakova u ispisu, tako da se

- ispred znaka konverzije stavi odgovarajući broj.

Primjer.

- `%3d` — ispisuje cijeli broj s najmanje 3 znaka.
- `%9s` — ispisuje najmanje 9 znakova stringa.

Ako podatak treba:

- manje znakova od zadane minimalne širine polja, bit će slijeva dopunjen razmacima do zadane širine (osim ako nije zadano drugačije dopunjavanje — "zastavicama").
- više znakova od minimalne širine ispisa, bit će ispisan sa svim potrebnim znakovima.

Preciznost ispisa realnih brojeva

Pored minimalne širine, moguće je zadati i preciznost ispisa. Kod realnih brojeva, preciznost je

- (najveći) broj decimala (za `%f` i `%e`), odnosno, vodećih znamenki (za `%g`), koje će biti ispisane.

Sintaksa:

- `%a.bf` ili `%a.be` ili `%a.bg`, gdje je
  - `a` — minimalna širina ispisa,
  - `b` — preciznost.

Primjer.

- `%7.3e` — znači ispis u `e` formatu s najmanje 7 znakova, pri čemu su najviše 3 znamenke iza decimalne točke.

Ispis bez specificirane preciznosti $\Longrightarrow$ preciznost $= 6$.

Preciznost ispisa realnih brojeva (nastavak)

Primjer. Ispis broja π na razne načine (v. `p_pr_10.c`)

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

int main(void) {
    double pi = 4.0 * atan(1.0);
    printf("%5f, %10.5f, %5.10f, %5.4f\n",
           pi, pi, pi, pi);
    return 0;
}
```

Rezultat ispisa je (zaokruživanjem na zadani broj decimala):

```
3.141593,    3.14159, 3.1415926536, 3.1416
```

Dinamičko zadavanje širine i preciznosti

Širinu i preciznost ispisa moguće je odrediti dinamički — u trenutku izvođenja programa, tako da se

- iznos širine ili preciznosti u formatu zamijeni znakom *.

Na pripadnom mjestu u listi argumenata, koje odgovara tom znaku *, mora biti

- cjelobrojni izraz — obično, varijabla.

Trenutna vrijednost tog argumenta određuje širinu, odnosno, preciznost, tj.

- "uvrštava" se, tog trena, umjesto znaka *.

Vrijednost tog argumenta se ne ispisuje (argument se "potroši" na supstituciju umjesto *) i ide se dalje, na sljedeći argument.

Dinamičko zadavanje širine i preciznosti (nast.)

Primjer. Opet, ispis broja π na razne načine (v. p_pr_11.c)

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

int main(void) {
    double pi = 4.0 * atan(1.0);  int i = 10;
    printf("%*f, %*.*f, %5.*f\n",
           11, pi, 16, 14, pi, i, pi);
    return 0;
}
```

ispisuje

```
3.141593, 3.14159265358979, 3.1415926536
```

Ispis znakovnih nizova

Znak konverzije `%s` služi za ispis znakovnih nizova (stringova). Ispisuje sve znakove u stringu dok ne dođe do nul-znaka `\0`, kojeg ne ispisuje.

Minimalna širina polja i preciznost mogu se koristiti i kod `%s` konverzije.

- Preciznost je maksimalni broj znakova koji smije biti ispisan.

Na primjer,

- `%5.12s` — specificira da će biti ispisano minimalno 5 znakova (dopunjenih vodećim razmacima, ako treba, do zadanih 5 znakova), a maksimalno 12 znakova.
- Ako string ima više od 12 znakova, "višak" neće biti ispisan (već samo prvih 12 znakova).

Sažetak o zastavicama

Zastavice služe za

- modificiranje standardnog ponašanja znakova konverzije.

Pišu se odmah iza znaka `%`, smije ih biti i više, i mogu biti napisane u bilo kojem međusobnom poretku.

- `-` označava lijevo pozicioniranje konvertiranog argumenta u polju za ispis.
- `+` označava da će broj uvijek biti ispisan s predznakom.
- `␣` (razmak ili praznina): ako prvi znak (nakon pretvorbe) nije predznak, dodat će se razmak (praznina, blank) na početak.
- `0` kod numeričkih konverzija, označava dopunjenje polja za ispis (do širine polja) vodećim nulama, a ne razmacima.

Sažetak o zastavicama (nastavak)

`#` označava alternativnu formu ispisa 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` — ispisani broj će uvijek imati decimalnu točku.

  Dodatno, za `g` i `G` — nule na kraju decimalnog broja (koje bi se mogle brisati) će se ispisati.

Primjer. Pogledajte programe od `p_pr_12.c` do `p_pr_16.c`.