Uvod u računarstvo
9. predavanje

Saša Singer

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

PMF – Matematički odjel, Zagreb

Sadržaj predavanja

- Ulaz i izlaz podataka:
  - Funkcije `getchar` i `putchar`.
  - Funkcije za testiranje znakova.
  - Funkcije `gets` i `puts`.
  - Funkcija `scanf`.
  - Funkcija `printf`.

Ulaz i izlaz

Funkcije za ulaz/izlaz

U standardnoj ulazno–izlaznoj biblioteci postoje sljedeće funkcije za ulaz/izlaz podataka (za standardne ulazne, odnosno, izlazne datoteke `stdin`, `stdout`):

- `getchar`, `putchar` — za znakove,
- `gets`, `puts` — za stringove,
- `scanf` i `printf` — za formatirani ulaz/izlaz.

Program koji koristi neku od tih funkcija

- mora uključiti datoteku zaglavlja `<stdio.h>`.

Funkcije `getchar` *i* `putchar`

```
int getchar(void);
int putchar(int);
```

Funkcija `getchar` čita jedan znak sa standardnog ulaza (tipično tipkovnice). Funkcija nema argumenata pa je sintaksa poziva:

```
c_var = getchar();
```

Funkcije `getchar` *i* `putchar` *(nastavak)*

Funkcija `putchar` šalje jedan znak na standardni izlaz (tipično ekran). Ona uzima jedan argument (znak koji treba ispisati) i vraća cjelobrojnu vrijednost. Najčešće poziv funkcije ima oblik

```
putchar(c_var);
```

pri čemu se vraćena vrijednost ignorira.

Funkcije `getchar` *i* `putchar` *(nastavak)*

Kada funkcija `getchar` naiđe na kraj ulaznih podataka vraća vrijednost `EOF` (skraćeno od engl. `End of File`).

`EOF` je simbolička konstanta definirana u `<stdio.h>` koja signalizira kraj datoteke, odnosno, kraj ulaznih podataka (ulaz je tretiran kao datoteka `stdin`).

Konstanta `EOF` mora se razlikovati od znakova iz sustava znakova koje računalo koristi. Stoga funkcija `getchar` ne vraća vrijednost tipa `char`, već vrijednost tipa `int`, što daje dovoljno prostora za kodiranje konstante `EOF` (obično -1).

Isto tako `putchar` uzima vrijednost tipa `int` i vraća vrijednost tipa `int`. Vraćena vrijednost je znak koji je ispisan ili `EOF` ako ispis znaka nije uspio.

Funkcije `getchar` *i* `putchar` *(nastavak)*

Primjer: program koji kopira znak po znak s ulaza na izlaz i pritom sva slova pretvara u velika.

U datoteci zaglavlja `<ctype.h>` deklarirana je funkcija `toupper` koja pretvara mala slova u velika, a sve druge znakove ostavlja na miru.

Funkcije `getchar` *i* `putchar` *(nastavak)*

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

int main(void) {
    int c;

    while ((c = getchar()) != EOF)
        putchar(toupper(c));

    return 0;
}
```

Pitanje: Što se događa ako piše samo `putchar(c);` ?

Kratko o stringovima

String je niz znakova koji završava tzv. nul znakom `'\0'` (oznaka za kraj).

Primjer. Primijetite razliku između sljedećih deklaracija:

```
char niz_znakova[5] = {'h', 'e', 'l', 'l', 'o'};

char string[6] = {'h', 'e', 'l', 'l', 'o', '\0'};
char string[] = "hello";
```

pri čemu u prvoj na kraj niza znakova ne dolazi simbol `'\0'`.

Funkcije gets *i* puts

```
char *gets(char *s);
int puts(const char *s);
```

Funkcije `gets` i `puts` služe čitanju i pisanju znakovnih nizova (stringova).

Funkcija `gets` čita znakovni niz sa standardnog ulaza (tipkovnice), sve dok ne naiđe na kraj linije `'\n'` koji zamjenjuje nul znakom `'\0'`.

Funkcija vraća pokazivač na `char` koji pokazuje na učitani znakovni niz ili `NULL` ako se došlo do kraja ulaznih podataka ili se javila greška. Simbolička konstanta `NULL` definirana je `<stdio.h>`.

Funkcije `gets` *i* `puts` *(nastavak)*

Funkcija `puts` uzima kao argument znakovni niz koji će biti ispisan na standardnom izlazu.

Funkcija vraća cijeli broj (tipa `int`). Ta vrijednost je:

- broj ispisanih znakova (nenegativan) ako je ispis uspio,
- a `EOF` ako nije.

Prije ispisa `puts` zamjenjuje nul znak `'\0'` (na kraju stringa) znakom `'\n'` za kraj reda.

Funkcije `gets` *i* `puts` *(nastavak)*

Primjer: program koji kopira liniju po liniju ulaza na izlaz.

```
#include <stdio.h>

int main(void) {
    char red[128];

    while (gets(red) != NULL)
        puts(red);

    return 0;
}
```

Funkcije `gets` *i* `puts` *(nastavak)*

Osnovni nedostatak funkcije `gets` je u tome što

- nije moguće odrediti maksimalni broj znakova koji će biti učitan.

Ako je broj znakova na ulazu veći od dimenzije polja koje je argument funkcije `gets`, doći će do greške ("gazimo" po memoriji).

Stoga je bolje umjesto `gets` koristiti funkciju `fgets` (bit će opisana kasnije).

Funkcija `scanf`

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

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

gdje je `kontrolni_string` konstantni znakovni niz koji sadrži informacije o vrijednostima koje se učitavaju u argumente `arg_1`, `...` , `arg_n`.

Kontrolni znakovni niz (string) je konstantan znakovni niz koji se sastoji od individualnih grupa znakova konverzije pri čemu je svakom argumentu pridružena jedna grupa.

Funkcija `scanf` *(nastavak)*

Svaka grupa znakova konverzije započinje znakom postotka (`%`) kojeg slijedi `znak konverzije` koji upućuje na tip podatka koji se učitava. Na pr. `%c` ili `%d`.

Najčešće korišteni znakovi konverzije su:

`znak konverzije`	`tip podatka koji se učitava`
`c`	jedan znak (`char`)
`d`	decimalni cijeli broj (`int`)
`e, f, g`	broj s pomičnim zarezom (`float`)
`h`	kratak cijeli broj (`short`)
⋮	⋮

Funkcija `scanf` *(nastavak)*

`znak konverzije`	`tip podatka koji se učitava`
⋮	⋮
`i`	decimalni, heksadecimalni ili oktalni cijeli broj (`int`)
`o`	oktalni cijeli broj (`int`)
`u`	cijeli broj bez predznaka (`unsigned int`)
`x`	heksadecimalni cijeli broj (`int`)
`s`	string (`char *`)
`p`	pokazivač (`void *`)

Funkcija `scanf` (nastavak)

- Unutar kontrolnog niza znakova grupe kontrolnih znakova mogu se nastavljati jedna na drugu bez razmaka ili mogu biti odvojene bjelinama (prazno mjesto, tabulator, prijelaz u novu liniju). Bjeline će u ulaznim podacima biti učitane i ignorirane.
- Argumenti funkcije `scanf` mogu biti samo pokazivači na varijable. Ako podatak treba učitati u neku varijablu, onda `scanf` uzima kao argument adresu te varijable.
- Podaci koje `scanf` čita dolaze sa standardnog ulaza, što je tipično tipkovnica. Ako se unosi više podataka oni moraju biti separirani bjelinama, što uključuje i prijelaz u novi red (koji se računa kao bjelina). Numerički podaci na ulazu moraju imati isti oblik kao i numeričke konstante.

Učitavanje cijelih brojeva

Cijeli brojevi mogu biti učitani kao decimalni (`%d`), ili kao 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: 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 vrijednost 13 (decimalno).

Učitavanje cijelih brojeva (nastavak)

Cijeli brojevi u oktalnom i heksadecimalnom zapisu mogu se upisivati i pomoću znakova konverzije `%o` i `%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 učitavamo u varijablu tipa `unsigned` znakom konverzije `%u`.
Znakovi konverzije `d`, `i`, `o`, `u`, `x` mogu dobiti prefiks `h` ako je argument pokazivač na `short` te 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 konvertira ih u varijable tipa `int`, `short` i `long`.

Učitavanje realnih brojeva

Znakovi konverzije `e`, `f` i `g` služe za učitavnje 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 ako je argument pointer na `long double`.

Formatiranje i konverzija ulaza

- Funkcija `scanf` dijeli niz znakova na ulazu u polja znakova odvojena bjelinama.
- Polje znakova (u principu) ne sadrži bjeline.
- Svako polje znakova interpretira se prema odgovarajućem znaku konverzije i pripadna vrijednost se upisuje u varijablu na koju pokazuje odgovarajući argument funkcije.
- Svaki znak konverzije (u principu) učitava jedno ulazno polje.

Formatiranje i konverzija ulaza (nastavak)

Primjer:

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

Prvi znak konverzije `%f` učitava i konvertira prvo polje znakova. Pritom se eventualne bjeline na početku preskaču. Prvo polje znakova završava bjelinom koju `%f` ne učitava.

Drugi znak konverzije `%d` preskače sve bjeline koje odjeljuju prvo polje znakova od drugog i učitava (i konvertira) drugo polje znakova.

Bjeline u kontrolnom stringu

Znakovi konverzije mogu biti odijeljeni bjelinama:

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

Ta bjelina ima za posljedicu preskakanje svih bjelina na ulazu do početka novog ulaznog polja.

Stoga je pisanje znakova konverzije u kontrolnom znakovnom nizu razdvojeno bjelinama (kao u primjeru `"%f %d"`) ili nerazdvojeno (kao `"%f%d"`) posve ekvivalentno.

To ne vrijedi za znakove konverzije `%c` i `[` (v. malo kasnije).

Drugi znakovi u kontrolnom stringu

U kontrolnom znakovnom nizu mogu se pojaviti i drugi znakovi osim bjelina i znakova konverzije. Njima moraju odgovarati posve isti znakovi na ulazu.

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 sljedeći znak 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);
```

u kojoj bjelina nakon `%f` preskače sve eventualne bjeline na ulazu ispred zareza.

Učitavanje znakovnih nizova — %s

Znak konverzije %s učitava niz znakova (string). Niz završava prvom bjelinom u ulaznom nizu znakova. Iza posljednjeg učitanog znaka automatski se dodaje nul-znak (\0).

Primjer:

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

Budući da se svako polje kao argument funkcije interpretira kao pokazivač na prvi element polja, ispred varijable string ne stavlja se adresni operator.

Učitavanje znakovnih nizova — %[...]

Znakom 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 %[...].

- Unutar uglatih zagrada upisuje se niz znakova.
- Funkcija scanf će učitati u pripadni argument najveći niz znakova s ulaza koji se sastoji od znakova navedenih unutar uglatih zagrada.
- Učitavanje završava prvi znak na ulazu koji nije naveden u uglatim zagradama i 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 najveći niz znakova sastavljen od velikih slova i razmaka.

- Prije `%[` ostavljen je jedan razmak koji govori funkciji `scanf` da preskoči sve bjeline koje prethode znakovnom nizu.
- To je nužno ako smo već imali poziv `scanf` funkcije, jer ona ostavlja završni znak prijelaza u novi red u ulaznom nizu (ne učita ga).

Učitavanje znakovnih nizova — %[...]

Naredba

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

bi pročitala prethodni znak prijelaza u novi red i budući da on nije unutar uglatih zagrada, završila bi čitanje ulaznih podataka i `linija` ne bi bila učitana.

Učitavanje znakovnih nizova — %[^...]

S uglatim zagradama možemo koristiti sintaksu

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

Sada se u odgovarajući argument učitava najveći mogući niz znakova sastavljen od svih znakova osim onih koji se nalaze u uglatim zagradama.

Primjer: cijelu liniju bez znaka za prijelaz u novi red možemo učitati pomoću naredbe

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

Na kraj učitanog niza znakova bit će dodan `\0`, a ispred `%[` mora biti ostavljeno prazno mjesto kako bi bile preskočene sve prethodne bjeline.

Učitavanje pojedinačnih znakova

Znak konverzije `c` učitava jedan znak u varijablu bez obzira je li on bjelina ili ne.

- Ako je prvi znak konverzije `c` potrebno je ispred njega staviti jednu bjelinu kako ne bi pročitao znak za prijelaz u novi red koji je ostao nakon prethodnog poziva funkcije `scanf`.
- Kontrolni niz `" %c%c%c"` čita tri znaka. Počet će s prvim znakom koji nije bjelina (zbog bjeline ispred prvog `%c` znaka) i pročitat će tri uzastopna znaka bili oni bjeline ili ne.
- Ako želimo čitati samo znakove bez bjelina treba koristiti `" %c %c %c"` ili `%c` zamijeniti s `%1s`.

Prefiks *

Neki podatak u listi moguće je preskočiti i ne pridružiti ga odgovarajućoj varijabli. To se radi tako da se znaku konverzije doda prefiks `*`.

Primjer:

```
scanf(" %s %*d %f", linija, &n, &x);
```

neće izvršiti pridruživanje drugog podatka varijabli `n`. On će biti preskočen, a treći podatak bit će normalno pridružen varijabli `x`.

Maksimalna širina ulaznog polja

Uz svaki kontrolni znak može se zadati maksimalna širina ulaznog polja koje će se učitati — tako da se ispred kontrolnog znaka stavi broj koji određuje širinu polja.

Primjer:

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

`%11c` učitava najviše 11 znakova.

- Ako podatak sadrži manje znakova od zadane maksimalne širine polja on se učita samo do prve bjeline.
- Ako podatak ima više znamenaka od maksimalne širine polja, "višak" znamenaka bit će učitan sljedećim konverzijskim znakom ili sljedećom `scanf` funkcijom.

Povratna vrijednost funkcije `scanf`

Funkcija `scanf` vraća broj uspješno učitanih podataka ili `EOF`.

Primjer: učitavanje brojeva većih ili jednakih od nule.

```
int n;

while (scanf("%d",&n) == 1 && n >= 0)
{
  // radi nesto s brojem
 }
```

`while` petlja se prekida ako je učitan negativan broj ili ako unos broja nije uspio.

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)
```

gdje je `kontrolni_string` konstantan 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)*

Kontrolni string sadrži dvije vrste objekata:

- obične znakove, koji se doslovno prepisuju (kopiraju) pri ispisu na izlaznu datoteku,
- specifikacije konverzije. To su grupe znakova koje počinju znakom %, a završavaju nekim znakom konverzije. Između ovih znakova može biti još znakova, s posebnim značenjima.

Svaka specifikacija konverzije vrši pretvaranje i ispis sljedećeg po redu (još neispisanog) argumenta u tom pozivu `printf`.

- Prvo se izračuna vrijednost tog argumenta,
- a zatim se, po pravilima konverzije, ta vrijednost pretvara u niz znakova, koji se onda ispisuje.

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`	heksadecimalni cijeli broj (`int`)
`e, f, g`	broj s pomičnim zarezom (`double`)
`c`	jedan znak (`char`)
`s`	string (`char *`)
`p`	pokazivač (`void *`)
`%`	nema konverzije, ispiši znak `%`

Funkcija `printf` *(nastavak)*

Uočiti: ako treba ispisati znak `%`, onda unutar kontrolnog znakovnog niza na tom mjestu treba staviti `%%`.

Funkcija `printf` vraća broj ispisanih znakova (nenegativan) ili `EOF`, ako je došlo do greške.

Pri pozivu funkcije `printf` dolazi do konverzije tipova:

- argumenti tipa `char` i `short` konvertiraju se u tip `int`,
- a argumenti tipa `float` u `double`.

Zbog toga, znak konverzije:

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

Funkcija `printf` — *primjer*

Argumenti funkcije `printf` (iza format–stringa) su izrazi, tj. mogu biti konstante, varijable, složeniji izrazi ili polja.

Primjer: (v. `printf_1.c`)

```
double x = 2.0;
...
printf("x=%f, y=%f\n", x, sqrt(x));
```

ispisuje jedan red teksta (znak `x` je prvi znak u redu):

```
x=2.000000, y=1.414214
```

Svi znakovi koji nisu dio specifikacije konverzije ispisani su točno onako kako su uneseni u kontrolnom znakovnom nizu: `"x=%f, y=%f\n"`

.

Funkcija `printf` — *primjer*

Znak možemo ispisati kao cijeli broj (`%d`) i kao jedan znak (`%c`).

Primjer:

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

ispisuje

```
c(int)=119, c(char)=w
```

ako računalo koristi ASCII skup znakova (broj 119 je ASCII kôd znaka "w").

Oktalni i heksadecimalni ispis

Pomoću znakova konverzije `%o` i `%x` cijeli brojevi ispisuju se u oktalnom i heksadecimalnom obliku

- bez predznaka i bez vodeće nule, odnosno, `0X`.

Primjer: (v. `printf_2.c`)

```
short i = 64;
...
printf("i(okt)=%o: i(hex)=%x: i(dec)=%d\n",
       i, i, i);
```

ispisuje

```
i(okt)=100: i(hex)=40: i(dec)=64
```

Oktalni i heksadecimalni ispis (nastavak)

Probajte isti program za `i = -3` (može tipa `short` ili `int`).

Dobijem (v. `printf_3.c` i `printf_4.c`):

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

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.

Ispis brojeva tipa `long`

Izrazi tipa `long` ispisuju se pomoću prefiksa `l`.

Primjer:

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

long i = LONG_MAX;
int main(void) {
    printf("i(okt) = %lo\n", i);
    printf("i(hex) = %lx\n", i);
    printf("i(dec) = %ld\n", i);

    return 0;
}
```

Ispis brojeva tipa `long` (nastavak)

Program, ovisno o računalu na kojem se izvršava, može ispisati:

```
i(okt) = 17777777777
i(hex) = 7fffffff
i(dec) = 2147483647
```

I ja dobijem isto na Intelovom compileru na IA–32 (v. `printf_5.c`).

Simbolička konstanta `LONG_MAX` definirana je u datoteci zaglavlja `<limits.h>` i predstavlja najveći broj tipa `long`.

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.

Za ispis brojeva tipa `long double` koristimo prefiks `L`. Pripadne specifikacije konverzije su `%Le`, `%Lf`, `%Lg`.

Ispis realnih brojeva (nastavak)

Primjer: dio programa (v. printf_6.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
```

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 bjelinama do zadane širine (osim ako nije zadano drugačije dopunjavanje).
- više znakova od minimalne širine ispisa, bit će ispisan sa svim potrebnim znakovima.

Minimalna širina ispisa (nastavak)

Primjer:

```
double x = 1.2;
...
printf("%1g\n%3g\n%5g\n", x, x, x);
```

ispisuje

```
1.2
1.2
  1.2
```

Prva dva ispisa imaju točno 3 znaka u svom redu, dok treći ima točno pet znakova, tj. ima dvije vodeće bjeline.

Preciznost ispisa realnih brojeva

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

- (najveći) broj decimala koje će biti ispisane.

Sintaksa:

- `%a.bf` ili `%a.bg` ili `%a.be`, 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 daje šest decimala.

Preciznost ispisa realnih brojeva (nastavak)

Primjer: ispis broja π na razne načine (v. `printf_7.c`).

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

int main(void) {
    double pi = 4.0 * atan(1.0);
    printf("%5f  %5.5f  %5.10f\n", pi, pi, pi);
    return 0;
}
```

Rezultat ispisa je (zaokruživanjem na zadani broj decimala):

```
3.141593  3.14159  3.1415926536
```

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.

Dinamičko zadavanje širine i preciznosti (nast.)

Primjer: (v. printf_8.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.

Primjer:

```
char naslov[] = "Programski jezik C";
...
printf("%s\n", naslov);
```

ispisuje

```
Programski jezik C
```

i prelazi u novi red, zbog `\n` iza `%s`.

Ispis znakovnih nizova (nastavak)

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 bjelinama ako treba), a maksimalno 12 znakova.
- Ako string ima više od 12 znakova, "višak" neće biti ispisan (već samo prvih 12 znakova).

Ispis znakovnih nizova (nastavak)

Primjer:

```
char naslov[] = "Programski jezik C";
...
printf("%.16s\n", naslov);
```

ispisuje

```
Programski jezik
```

Zadnji znak `k` je i zadnji znak u tom redu.