

# Uvod u programiranje

- predavanja -

prosinac 2020.

---

## 22. Standardna biblioteka

- 3. dio -

# Standardna biblioteka

`<stdio.h>`

*Formatted Input and Output*

# scanf, fscanf

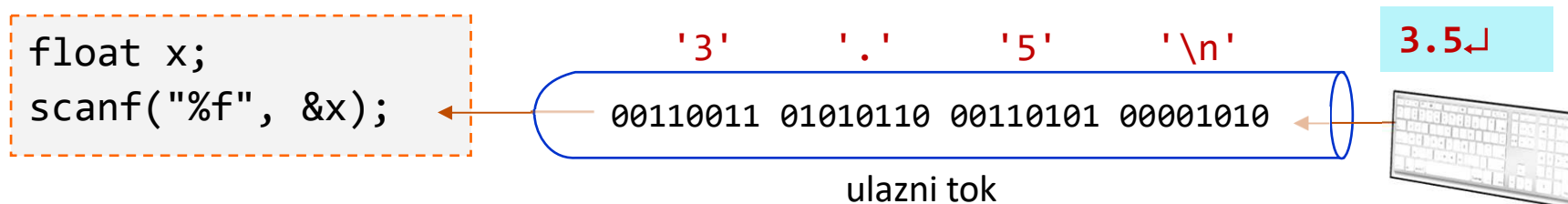
<stdio.h>

```
int scanf(const char *format, ...);  
int fscanf(FILE *stream, const char *format, ...);
```

- `scanf(format, ...) ≡ fscanf(stdin, format, ...)`
  - čitanje znakova iz standardnog ulaza (`scanf`) ili zadanog ulaznog *toka* (`fscanf`), u skladu s formatom kojim se definiraju
    - dopušteni oblik ulaza
    - vrste *konverzija* koje nad ulazom treba obaviti da bi se dobile vrijednosti točno određenog tipa
  - vrijednosti dobivene *konverzijom* znakova s ulaza redom se upisuju u memoriju na mjesta određena preostalim argumentima (pokazivačima na objekte)
    - funkcija vraća broj vrijednosti koje je uspjela upisati u objekte na koje pokazuju argumenti
- Opisane su tek najvažnije mogućnosti funkcije `scanf`.
  - Detaljniji opis funkcije `scanf` može se pronaći u gotovo svakom C priručniku.

## Zašto se govori o *konverziji*?

- funkcija iz ulaznog toka čita **znakove** (bajtove od kojih svaki sadrži po jednu ASCII vrijednost znaka)



- obavlja konverziju **pročitanih znakova** u podatak odgovarajućeg tipa (tip podatka određen je konverzijskom specifikacijom). U konkretnom slučaju, realni broj standardne preciznosti, IEEE 754
  - 01000000 01100000 00000000 00000000
- rezultat dobiven konverzijom upisuje na mjesto u memoriji na koje pokazuje argument &x, dakle u varijablu x
- budući da se konverzija obavlja prema specifikacijama iz formata
  - scanf, printf i slične funkcije se nazivaju funkcije za formatirani ulaz/izlaz

## Format za funkciju scanf

- format za funkciju scanf može sadržavati
  - konverzijske specifikacije
    - npr. %d, %f
  - bijele praznine, odnosno *bjelina* (*whitespace*)
    - znak praznine (*space*, ASCII 32<sub>10</sub>) ili vodoravni tabulator (*horizontal tab*, ASCII 11<sub>10</sub>)
  - ostale (*non-whitespace*) znakove

# Konverzijska specifikacija

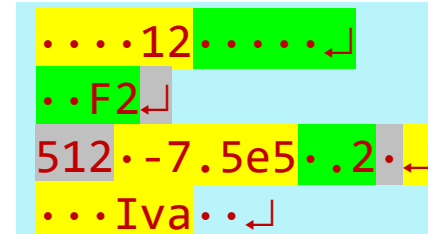
- Opći oblik `%[*][širina][modifikator]specifikator`

opcionalni modifikator	specifikator	konverzija u tip
	c	znak
h - short l - long ll - long long	d	cijeli broj
	u	cijeli broj bez predznaka
	o	cijeli broj (iz oktalnog zapisa)
	x	cijeli broj (iz heksadekadskog zapisa)
l - double L - long double	f	realni broj
	s, [...]	niz znakova

- npr. `%hd`: konverzijska specifikacija za konverziju u short `int`
- konverzijske specifikacije nalažu sljedeće
  - preskoči sve eventualne bjeline na ulazu (osim za specifikator c i [...])
  - čitaj dok god znakovi na ulazu odgovaraju specifikatoru

# Primjer

```
short i; unsigned int j; int k;  
float x; double y;  
char c; char niz[20];  
scanf("%hd%x%d", &i, &j, &k);  
scanf("%f%lf", &x, &y);  
scanf("%c%s", &c, niz);
```



- %hd preskače 4 praznine, čita znakove 12, pretvara ih u short 12
- %x preskače 5 praznina, novi red, 2 praznine, čita znakove F2, pretvara ih u unsigned int 242
- %d preskače novi red, čita znakove 512 i pretvara ih u int 512
- %f preskače prazninu, čita znakove -7.5e5 i pretvara ih u float  $-7.5 \cdot 10^5$
- %lf preskače prazninu, čita znakove .2 i pretvara ih u double 0.2
- %c ne preskače ništa, čita prazninu i upisuje njezinu ASCII vrijednost u varijablu c
- %s preskače novi red, 3 praznine, čita znakove Iva, upisuje ih u polje niz i dodaje '\0'
- dvije praznine i znak za novi red ostaju na ulazu, nepročitani

## Specifikator [*znakovi*]

- konverzijske specifikacije %[*znakovi*] i %[<sup>^</sup>*znakovi*] slične su specifikaciji %s po tome što čitaju znakove s ulaza i pohranjuju ih u niz znakova kojeg na kraju terminiraju znakom '`\0`', uz sljedeće razlike:
  - ne preskaču bjeline na ulazu
  - %[*znakovi*] čita sve znakove s ulaza koji se nalaze u *znakovi*
  - %[<sup>^</sup>*znakovi*] čita sve znakove s ulaza koji se **ne** nalaze u *znakovi*
- primjeri specifikacija
  - %[abcdefABCDEF0123456789]
    - učitava znakove dok god se na ulazu nalaze heksadekadske znamenke
  - %[`\n`]
    - učitava znakove dok god se na ulazu nalaze praznine ili oznake novog retka
  - %[<sup>^</sup>`\n!`]
    - učitava znakove dok god se na ulazu ne pojavi oznaka novog retka ili znak uskličnik



## Primjer

```
char niz1[40], niz2[40];  
char niz3[40], niz4[40];  
scanf("%[ABCDEF0123456789]", niz1);  
scanf("%[^\\n!]", niz2);  
scanf("%[.!\n]", niz3);  
scanf("%[^.]", niz4);
```

```
C00Fa....!..  
.....  
.Mali.auto  
dobro.vози.Narocito.  
u.gradu.
```

- `%[ABCDEF0123456789]` čita znakove C00F, upisuje u `niz1` i terminira `niz1`
- `%[^\\n!]` čita znakove a...., upisuje u `niz2` i terminira `niz2`
- `%[.!\n]` čita znakove !..  
....., upisuje u `niz3` i terminira `niz3`
- `%[^.]` čita znakove Mali.auto.  
dobro.vози, upisuje u `niz4` i terminira `niz4`
- znakovi .Narocito.  
u.gradu. ostaju na ulazu, nepročitani

# Konverzijska specifikacija

`%[*][širina][modifikator]specifikator`

- opcionalna širina (pozitivni cijeli broj)
  - čitaj najviše zadani broj znakova

```
int i, j;  
float x;  
scanf("%3d%5d", &i, &j);  
scanf("%5f", &x);
```

12345...3.1415926↵

- `%3d` čita znakove 123 i pretvara ih u `int` 123
- `%5d` čita znakove 45 i pretvara ih u `int` 45
- `%5f` preskače 3 praznine, čita znakove 3.141 i pretvara ih u `float` 3.141
- znakovi 5926↵ ostaju na ulazu, nepročitani

# Konverzijska specifikacija

%[\*][širina][modifikator]specifikator

- instrukcija funkciji scanf da treba suspendirati konverziju i pridruživanje. Znakovi će se pročitati u skladu s ostatkom navedene specifikacije, ali se konverzija neće obaviti
  - to znači da za dotičnu specifikaciju ne treba navesti pripadni pokazivač
  - korisno ako neke znakove na ulazu treba samo preskočiti

```
int i; float x;  
scanf("%*[#]%d", &i);  
scanf("%*[.\n#C]%f", &x);
```

```
#####150·C↵  
#####9.65·paskala↵
```

- %\*[#] čita znakove ##### i zanemaruje ih
- %d čita znakove 150 i pretvara ih u int 150
- %\*[.\n#C] čita znakove ·C##### i zanemaruje ih
- %f čita znakove 9.65 i pretvara ih u float 9.65
- znakovi ·paskala↵ ostaju na ulazu, nepročitani

## Bjeline i ostali znakovi navedeni u formatu

- Bjelina u formatu
  - instrukcija za funkciju `scanf` da treba preskočiti sve sljedeće bjeline na ulazu (*space*, *horizontal tab* i *newline*) dok god se ne pojavi neki *non-whitespace* znak
- Ostali (*non-whitespace*) znakovi u formatu
  - svaki *non-whitespace* znak zahtijeva da se na ulazu pojavi upravo takav znak

## Napomena uz konverzijsku specifikaciju %c

- specifikacija %c ne preskače bjeline na ulazu. Stoga, ako se namjerava učitati samo znak koji nije bjelina, potrebno je prvo pomoću bjeline navedene u formatu preskočiti bjeline na ulazu

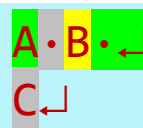
```
char c1, c2, c3;  
scanf("%c%c%c", &c1, &c2, &c3);
```

- %c čita znak A i upisuje ga u c1
- %c čita znak praznine i upisuje ga u c2
- %c čita znak B i upisuje ga u c3



```
char c1, c2, c3;  
scanf("·%c·%c·%c", &c1, &c2, &c3);
```

- u formatu nema što preskočiti na ulazu
- %c čita znak A i upisuje ga u c1
- u formatu preskače znak · na ulazu
- %c čita znak B i upisuje ga u c2
- u formatu preskače znakove ·↵ na ulazu
- %c čita znak C i upisuje ga u c3



## Napomena uz konverzijsku specifikaciju %s

- specifikacija %s prestaje učitavati niz znakova kada naiđe na prvu bjelinu na ulazu

```
char niz1[40], niz2[40], niz3[40];  
scanf("%20s", niz1);  
scanf("%s", niz2);  
scanf("%3s", niz3);
```

..Ana·Marija↵  
...Ivana·↵

- %20s preskače znakove · · , čita znakove Ana, upisuje ih u niz1 i terminira niz
- %s preskače znak · , čita znakove Marija, upisuje ih u niz2 i terminira niz
- %3s preskače znakove ↵· · · , čita znakove Iva, upisuje ih u niz3 i terminira niz
- znakovi na ·↵ ostaju na ulazu, nepročitani

## Napomena uz konverzijsku specifikaciju %s

- niz znakova koji sadrži bjeline može se pročitati pomoću %[...]

```
char niz1[40], niz2[40];  
scanf("%[^\\n]*c", niz1);  
scanf("%[^\\n]*c", niz2);
```

```
..Ana·Marija↵  
...Ivana·↵
```

- %[^\\n] čita znakove ..Ana·Marija, upisuje ih u niz1 i terminira ga. Zaustavlja se ispred \\n.
- %\*c preskače znak \\n
- %[^\\n] čita znakove ...Ivana·, upisuje ih u niz2 i terminira ga. Zaustavlja se ispred \\n.
- %\*c preskače znak \\n

## Prijevremeni prekid izvršavanja funkcije

- ako znak na ulazu ne odgovara konverzijskoj specifikaciji, funkcija se prijevremeno prekida. Ostatak ulaza, uključujući znak koji nije odgovarao konverzijskoj specifikaciji, ostaje nepročit.

```
int n, rez; float x, y;  
rez = scanf("%f%d%f", &x, &n, &y);
```

..1.5..62.50↵

- %f preskače .., čita znakove 1.5, pretvara u float 1.5, upisuje u x
- %d preskače znakove .., čita znakove 62, pretvara u int 62, upisuje u n
- %f čita znakove .50, pretvara u float 0.5, upisuje u y
- znak ↵ ostaju na ulazu, nepročitani. Funkcija je vratila 3  $\Rightarrow$  rez = 3

```
int n, rez; float x, y;  
rez = scanf("%f%d%f", &x, &n, &y);
```

..1.5..62.50↵

- %f preskače .., čita znakove 1.5, pretvara u float 1.5, upisuje u x
- %d preskače znakove .., točka nije u skladu s %d, vraća se na stdin
- znakovi .62.50↵ ostaju na ulazu, nepročitani. Funkcija vraća 1  $\Rightarrow$  rez = 1



# Primjeri

```
int i, j;  
float x, y, z;  
scanf("%d%d.%f.%f.%f.", &i, &j, &x, &y, &z);
```

...38  
...-15.012...  
..24+5e2...-8

- %d preskače ... , čita znakove 38, pretvara u int 38, upisuje u i
- %d preskače znakove ↵... , čita znakove -15, pretvara u int -15, upisuje u j
- • u formatu bi preskočila bjeline kad bi ih bilo na ulazu. Ne radi ništa.
- %f bi preskočio bjeline kad bi ih bilo na ulazu. Čita znakove .012, pretvara u float 0.012, upisuje u x
- • u formatu preskače .....↵..
- %f bi preskočio bjeline kad bi ih bilo na ulazu. Čita znakove 24, pretvara u float 24.0, upisuje u y
- • u formatu bi preskočila bjeline kad bi ih bilo na ulazu. Ne radi ništa.
- %f bi preskočio bjeline kad bi ih bilo. Čita +5e2, pretvara u float 500.0, upisuje u z
- • u formatu preskače ...
- znakovi -8↵ ostaju na ulazu, nepročitani. Funkcija vraća 5

# Primjeri

```
int d, m, g, rez;  
rez = scanf("Datum:%d.%d.%d", &d, &m, &g);
```

Datum: 15.1.2017.↵

- Datum: preskače Datum:
- %d preskače . , čita 15, pretvara i upisuje u d
- . iz formata preskače . na ulazu ... itd. Na kraju, znakovi .↵ ostaju na ulazu, nepročitani

```
rez = scanf("Datum:%d.%d.%d", &d, &m, &g);
```

Datum: : 15.1.2017.↵

- Datum: ne odgovara znakovima na ulazu. Funkcija se prijevremeno prekida, vraća 0
- znakovi : 15.1.2017. ostaju na ulazu, nepročitani
- za popravak bi bilo dovoljno u formatu iza Datum staviti jednu prazninu

```
rez = scanf("Datum: :%d.%d.%d", &d, &m, &g);
```

Datum: 15.1.2017.↵

- . u formatu bi preskočila bjeline kad bi ih bilo na ulazu. Ne radi ništa.
- svi cijeli brojevi će se uspjeti pročitati. Funkcija vraća 3
- na kraju, znakovi .↵ ostaju na ulazu, nepročitani

# printf, fprintf

<stdio.h>

```
int printf(const char *format, ...);  
int fprintf(FILE *stream, const char *format, ...);
```

- `printf(format, ...) ≡ fprintf(stdout, format, ...)`
- pisanje znakova na standardni izlaz (`printf`) ili zadani izlazni *tok* (`fprintf`), u skladu sa zadanim formatom
- vrijednosti argumenata navedenih iza formata formatiraju se u skladu s konverzijskim specifikacijama
  - ostali znakovi koji se nalaze u formatu ispisuju se nepromijenjeni
- funkcija vraća broj znakova koje je ispisala ili EOF ako se pri ispisu dogodi pogreška

- Opisane su tek najvažnije mogućnosti funkcije `printf`.
- Detaljniji opis funkcije `printf` može se pronaći u gotovo svakom C priručniku.

# Konverzijska specifikacija

- Opći oblik %[znak][širina][.preciznost][modifikator]specifikator

opcionalni modifikator	specifikator	konverzija iz tipa i oblik ispisa
	c	znak
h - short l - long ll - long long	d	cijeli broj
	u	cijeli broj bez predznaka
	o	cijeli broj, oktalna notacija
	x, X	cijeli broj, heksadekadska notacija (a-f ili A-F)
l - double L - long double	f	realni broj bez prikaza eksponenta
	e, E	realni broj s eksponentom, ispis e ili E
	g, G	realni broj s eksponentom ili bez, ispis e ili E
	s	niz znakova
	p	pokazivač

# Konverzijska specifikacija

`%[znak][širina][.preciznost][modifikator]specifikator`

- širina (pozitivni cijeli broj) određuje najmanju širinu polja za ispis
  - za zadanu širinu  $n$ , ispisat će se najmanje  $n$  znakova
  - ako je  $n$  veći od potrebne širine podatka, podatak se pozicionira desno unutar polja ispisa širine  $n$ , s vodećim prazninama
  - ako je podatak širi od  $n$ , ili ako širina nije zadana, podatak će se ispisati u širini koja je potrebna za ispis tog podatka
- preciznost (pozitivni cijeli broj)
  - za  $e$ ,  $E$ ,  $f$ : određuje broj znamenki iza decimalne točke
  - za ostale specifikatore ima drugačije značenje, ovdje se preciznost za te specifikatore neće razmatrati
  - ako se preciznost ne zada, koristi se preciznost po definiciji (npr. za  $e$ ,  $E$ ,  $f$ , to je šest znamenki iza decimalne točke)

# Konverzijska specifikacija

%[znak][širina][.preciznost][modifikator]specifikator

- pruža dodatne mogućnosti prilagodbe ispisa, npr.
  - ispis vodećih nula
  - ispis predznaka i za pozitivne brojeve
  - lijevo pozicioniranje u polju ispisa

```
printf("%5d\n", 25);  
printf("%05d\n", 25);  
printf("%+5d\n", 25);  
printf("%-5d\n", 25);
```

```
...25↵  
00025↵  
..+25↵  
25...↵
```

## Primjer

```
float x = 321.f, y = 1.234e-7f, z = 7.65432e9f;
printf("|%f|%f|%f|\n", x, y, z);
printf("|%10f|%10f|%10f|\n", x, y, z);
printf("|%10.4f|%10.4f|%10.4f|\n", x, y, z);
printf("|%.4f|%.4f|%.4f|\n", x, y, z);
printf("|%3.1f|%3.1f|%3.1f|\n", x, y, z);
printf("|%13.11f|%13.11f|%13.11f|\n", x, y, z);
```

```
| 321.000000|0.000000|7654320128.000000|
| 321.000000|..0.000000|7654320128.000000|
|..321.0000|....0.0000|7654320128.0000|
| 321.0000|0.0000|7654320128.0000|
| 321.0|0.0|7654320128.0|
| 321.00000000000|0.00000012340|7654320128.00000000000|
```

## Primjer

```
float x = 321.f, y = 1.234e-7f, z = 7.65432e9f;  
printf("|%e|%e|%e|\n", x, y, z);  
printf("|%15e|%15e|%15e|\n", x, y, z);  
printf("|%15.2E|%15.2E|%15.2E|\n", x, y, z);
```

```
|3.210000e+002|1.234000e-007|7.654320e+009|  
|..3.210000e+002|..1.234000e-007|..7.654320e+009|  
|.....3.21E+002|.....1.23E-007|.....7.65E+009|
```



## Primjer

```
float x = 321.f, y = 1.234e-7f, z = 7.65432e9f;  
printf("|%g|%g|%g|\n", x, y, z);  
printf("|%15G|%15G|%15G|\n", x, y, z);
```

```
|321|1.234e-007|7.65432e+009|  
|.....321|.....1.234E-007|...7.65432E+009|
```

## Primjer

```
char *s1 = "Ana ";  
char *s2 = " Iva";  
char *s3 = "Ana-Marija";  
printf("|%s|%s|%s|\n", s1, s2, s3);  
printf("|%12s|%12s|%12s|\n", s1, s2, s3);  
printf("|%6s|%6s|%6s|\n", s1, s2, s3);
```

```
|Ana | Iva|Ana-Marija|  
|.....Ana |.....Iva|..Ana-Marija|  
|..Ana·|...Iva|Ana-Marija|
```

## Preusmjeravanje toka *standardni izlaz*

- Standardni izlazni tok može se preusmjeriti u trenutku pokretanja programa (na razini operacijskog sustava)

```
...  
int main(void) {  
    ...  
    printf("Upisite n > ");  
    scanf("%d", &n);  
    if (n < 1 || n > 46) {  
        printf("Prevelik/premalen n");  
        exit(1);  
    }  
    for (i = 1; i <= n; i = i + 1) {  
        ...  
        printf("%d\n", fib_i);  
    }  
    ...  
}
```

```
C:\upro> prog > fib.txt  
10
```

```
fib.txt Upisite n > 1  
1  
2  
3  
5  
...
```

```
C:\upro> prog > fib.txt  
50
```

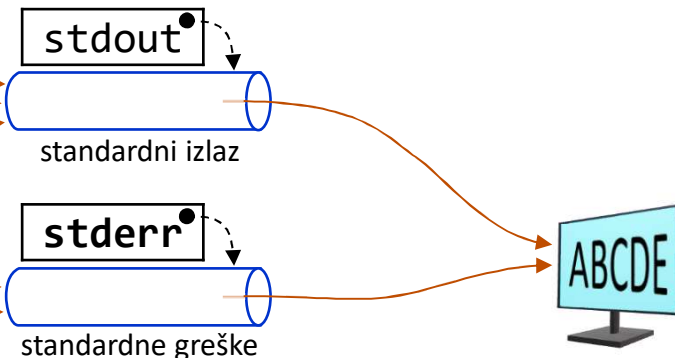
```
fib.txt Upisite n > Prevelik/premalen n
```

# Tok standardne greške

- pored tokova standardni ulaz i standardni izlaz, u trenutku pokretanja programa automatski se otvara i tok *standardne greške*
  - slično eksternim varijablama `stdin` i `stdout`, eksterna varijabla `stderr` je tipa pokazivač na tok *standardne greške*

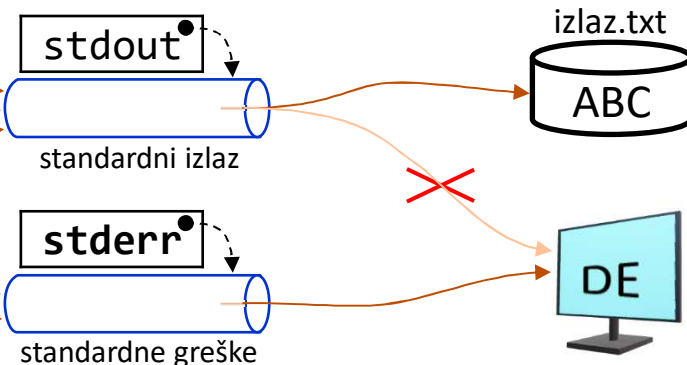
```
C:\upro> prog
```

```
putchar('A');  
printf("B");  
fprintf(stdout, "C");  
putc('D', stderr);  
fprintf(stderr, "E");
```



```
C:\upro> prog > izlaz.txt
```

```
putchar('A');  
printf("B");  
fprintf(stdout, "C");  
putc('D', stderr);  
fprintf(stderr, "E");
```



## Korištenje toka *standardna greška*

- Ako neki sadržaj na zaslon treba ispisati bez obzira na eventualno preusmjeravanje standardnog izlaza, treba koristiti izlazni tok standardne greške

```
...  
int main(void) {  
    ...  
    fprintf(stderr, "Upisite n > ");  
    scanf("%d", &n);  
    if (n < 1 || n > 46) {  
        fprintf(stderr, "Prevelik/premalen n");  
        exit(1);  
    }  
    for (i = 1; i <= n; i = i + 1) {  
        ...  
        printf("%d\n", a_i);  
    }  
    ...  
}
```

```
C:\upro> prog > fib.txt  
Upisite n > 10
```

fib.txt

```
1  
1  
2  
3  
5  
...
```

```
C:\upro> prog > fib.txt  
Upisite n > 50  
Prevelik/premalen n
```

fib.txt

# Preusmjeravanje toka *standardni ulaz*

- Preusmjeravanje standardnog ulaza

```
...  
int main(void) {  
    ...  
    printf("Upisite n > ");  
    scanf("%d", &n);  
    if (n < 1 || n > 46) {  
        printf("Prevelik/premalen n");  
        exit(1);  
    }  
    for (i = 1; i <= n; i = i + 1) {  
        ...  
        printf("%d\n", fib_i);  
    }  
    ...  
}
```

ulaz.txt

10

```
C:\upro> prog < ulaz.txt  
Upisite n > 1
```

```
1  
2  
3  
5  
...
```

# Preusmjeravanje toka *standardni ulaz i izlaz*

- Istovremeno preusmjeravanje standardnog ulaza i izlaza

```
...  
int main(void) {  
    ...  
    printf("Upisite n > ");  
    scanf("%d", &n);  
    if (n < 1 || n > 46) {  
        printf("Prevelik/premalen n");  
        exit(1);  
    }  
    for (i = 1; i <= n; i = i + 1) {  
        ...  
        printf("%d\n", fib_i);  
    }  
    ...  
}
```

ulaz.txt

10

```
C:\upro> prog < ulaz.txt > fib.txt↵
```

fib.txt

```
Upisite n > 1  
1  
2  
3  
5  
...
```

## sscanf, sprintf

<stdio.h>

```
int sscanf(const char* buffer, const char* format, ...);  
int sprintf(char *buffer, const char *format, ...);
```

- identične funkcijama fscanf i fprintf, osim po sljedećem: umjesto toka, kao izvor, odnosno destinaciju, funkcije koriste niz znakova.
  - sscanf je korisna u slučajevima kada je niz znakova koji se nalazi na ulazu potrebno formatirano pročitati više nego jednom
  - sprintf automatski terminira niz znakom '\0'. Funkcija je korisna onda kada u *niz znakova* treba upisati nešto u skladu sa zadanim formatom, a taj niz se neće ispisivati u neki tok ili će se ispisivati tek kasnije



# Primjer

- Programski zadatak
  - s tipkovnice učitati niz znakova koji zajedno s eventualno učitanim oznakom novog retka ne smije biti dulji od 20 znakova
  - ako učitani niz sadrži podniz OCT ili HEX, tada od početka niza pročitati oktalni, odnosno heksadekadski broj, te ga ispisati kao dekadski broj. Ako niz ne sadrži podniz OCT ili HEX ili učitavanje broja s početka niza ne uspije, ispisati poruku "Neispravan ulaz"

```
Upisite niz > 12OCT↵  
Ucitan je broj 10↵
```

```
Upisite niz > 1eM HEX↵  
Ucitan je broj 30↵
```

```
Upisite niz > M1e HEX↵  
Neispravan ulaz
```

```
Upisite niz > 219 hex ↵  
Neispravan ulaz
```

# Rješenje

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

int main(void) {
    char niz[20 + 1];
    unsigned int broj;
    int procitanoBrojeva = 0;
    printf("Upisite niz > ");
    fgets(niz, 20 + 1, stdin);
    if (strstr(niz, "OCT") != NULL)
        procitanoBrojeva = sscanf(niz, "%o", &broj);
    else if (strstr(niz, "HEX") != NULL)
        procitanoBrojeva = sscanf(niz, "%x", &broj);
    if (procitanoBrojeva == 1)
        printf("Ucitan je broj %d\n", broj);
    else
        printf("Neispravan ulaz\n");

    return 0;
}
```

## Primjer

- Funkcije `sprintf` i `scanf` ne smiju se koristiti bez razloga!
  - ako program s tipkovnice treba samo pročitati, a zatim na zaslon ispisati tri cijela broja, *vrlo loše* je napisati:

```
char ulaz[80], izlaz[80];  
int m, n, k;  
fgets(ulaz, 80, stdin);  
sscanf(ulaz, "%d %d %d", &m, &n, &k);  
sprintf(izlaz, "%d %d %d", m, n, k);  
puts(izlaz);
```

NEISPRAVNO

- umjesto:

```
int m, n, k;  
scanf("%d %d %d", &m, &n, &k);  
printf("%d %d %d", m, n, k);
```

ISPRAVNO