

Ugrađene funkcije

Ugrađene matematičke funkcije

```
#include <math.h> ← mjesto gdje su prototipovi  
  
int abs (int x); ← prototipovi  
long labs (long x); |x| za long int  
double fabs (double x); |x| za double  
double sin (double x); sin x  
double cos (double x); cos x  
double tan (double x); tan x  
double asin (double x); arcsin x  
double acos (double x); arccos x  
double atan (double x); arctan x
```

Ugrađene matematičke funkcije

<code>double sinh (double x);</code>	sh x
<code>double cosh (double x);</code>	ch x
<code>double tanh (double x);</code>	th x
<code>double exp (double x);</code>	e^x
<code>double log (double x);</code>	ln x
<code>double log10 (double x);</code>	log x
<code>double pow (double x, double y);</code>	x^y
<code>double sqrt (double x);</code>	\sqrt{x}
<code>double fmod (double x, double y);</code>	x MOD y
<code>double ceil (double x);</code>	zaokr. na gore
<code>double floor (double x);</code>	zaokr. na dolje

ceil vraća najmanji cijeli broj koji je veći ili jednak x
floor vraća najveći cijeli broj koji je manji ili jednak x

3

Ugrađene posebne funkcije iz <stdlib.h>

```
#include <stdlib.h>
void exit (int status);
```

`exit(x)` trenutno prekida izvođenje programa i pozivajućem programu (operacijskom sustavu) vraća vrijednost x

`exit(x)` unutar main funkcije je ekvivalentno s `return x`

4

Ugrađene posebne funkcije iz <stdlib.h>

Generiranje pseudoslučajnih brojeva

```
void srand (unsigned int seed);
```

- inicijalizira generator pseudoslučajnih brojeva. Za isti *seed* dobit će se uvijek isti niz pseudoslučajnih brojeva

```
int rand (void);    [0, RAND_MAX] ≡ [0, 32767]
```

- uzastopnim pozivanjem ove funkcije dobiva se niz pseudoslučajnih brojeva iz intervala [0, **RAND_MAX**]
- macro **RAND_MAX** određuje najveću vrijednost koju vraća funkcija **rand**. Definiran je u <stdlib.h>. U VS C prevodiocu iznosi 32767.

5

Ugrađene posebne funkcije iz <stdlib.h>

```
int rand (void) → [0, 32767]
```

Kako jednoliko preslikati cijele brojeve x iz intervala $[a, b]$ u interval $[c, d]$?

$$y = \underbrace{(x - a) / (b - a + 1) * (d - c + 1)}_{\text{skaliranje}} + \underbrace{c}_{\text{translacija}}$$

6

Primjer: Načiniti funkciju koja simulira bacanje kocke. Baciti kocku zadani broj puta. Ispisati frekvenciju pojavljivanja svih brojeva.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int kocka () {
    float y = (float) rand() / (RAND_MAX+1) * 6 + 1;
    return (int)y;
}
int main () {
    int brojac[6] = {0};
    int i, n;
    printf ("RAND_MAX = %d\n", RAND_MAX);
    printf ("Unesite broj bacanja kocke >");
    scanf ("%d", &n);
    /* time() vraća broj sekundi od ponoći, 1. siječnja 1970. (GMT)*/
    srand ((unsigned) time(NULL));
```

← vraća time_t

7

Primjer: Bacanje kocke, nastavak

```
for (i = 0; i < n; i++) {
    ++brojac[kocka()-1];
}
for (i = 0; i < 6; i++) {
    printf ("%d %5d\n", i+1, brojac[i]);
}
return 0;
}
```

Primjer izvođenja programa:

```
RAND_MAX = 32767
Unesite broj bacanja kocke >100000
1 166579
2 166446
3 166802
4 167009
5 166714
6 166450
```

8

Razlika između konstantnog znakovnog niza i niza znakova

- Niz znakova: jednodimenzionalno polje znakova s '\0':

```
#define DULJINA_NIZA 8
char ime_niza[DULJINA_NIZA + 1];
```

isto {

```
char ime1[5] = {'I', 'v', 'a', 'n', '\0'};
```

polje čiji su elementi inicijalizirani na 'I', 'v', 'a', 'n', '\0'

```
char ime1[] = "Ivan";
```

```
char *ime2 = "Ana";
```

pokazivač koji pokazuje na konstantni znakovni niz "Ana"

9

Razlika između konstantnog znakovnog niza i niza znakova

- **Primjer:**

```
char *ime = " ";
ime[2] = 'A'; ili *(ime+2) = 'A'; /*nije dopušteno*/
```

ime pokazuje na konstantni znakovni niz čiji se sadržaj ne smije mijenjati

- **Primjer:**

```
char *ime;
char polje[3+1] = " ";
polje[2] = 'A'; /*dopušteno*/
ime = polje; ili ime = &polje[0];
ime[2] = 'A'; ili *(ime+2) = 'A'; /*dopušteno*/
```

ime pokazuje na prvi element polja znakova (čiji je elemente dopušteno mijenjati)

10

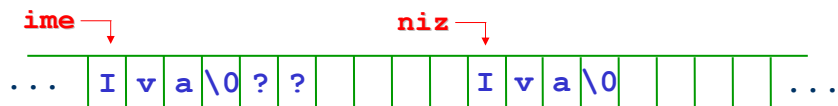
Ugrađene funkcije za operacije nad nizovima znakova `strcpy` <string.h>

- Kopiranje niza znakova (kopira src u dest, uključujući \0, vraća dest)
`char *strcpy(char *dest, const char *src);`

```
char ime[5+1];  
char niz[] = "Iva";    ili char *niz = "Iva";
```



```
strcpy (ime, niz);    ili strcpy (ime, "Iva");
```



Što bi se dogodilo da je polje definirano ovako: `char ime[2] ?`

11

Prvi argument funkcije `strcpy` ne smije biti konstantni znakovni niz

- **Primjer:**

```
char *ime = "    ";  
char niz[] = "Iva";    ili char *niz = "Iva";  
strcpy (ime, niz);    /* nije dopušteno */
```

`ime` pokazuje na konstantni znakovni niz čiji se sadržaj ne smije mijenjati

- **Primjer:**

```
char *ime;  
char polje[3+1];  
char niz[] = "Iva";    ili char *niz = "Iva";  
strcpy (polje, niz);    /* dopušteno */  
ime = polje;    ili ime = &polje[0];  
strcpy (ime, niz);    /* dopušteno */
```

`ime` pokazuje na prvi element polja znakova (čiji je elemente dopušteno mijenjati)

12

Ugrađene funkcije za operacije nad nizovima znakova `strncpy` `<string.h>`

- Kopiranje **dijela** niza znakova: kopira se maxlen znakova iz niza src u niz dest (to znači da se \0 možda neće uspjeti kopirati!). Ako je maxlen veći od duljine niza koji se kopira, u dest se dodaju \0 znakovi dok se ne dospije do duljine maxlen. Funkcija vraća dest.

```
char *strncpy(char *dest,
              const char *src,
              size_t maxlen);
```

↓ rez

```
char rez[10];           → ... ? ? ? ? ? ? ? ? ...
strncpy(rez, "Ana", 2); → ... A n ? ? ? ? ? ? ? ...
strncpy(rez, "Ana", 3); → ... A n a ? ? ? ? ? ? ? ...
strncpy(rez, "Ana", 4); → ... A n a \0 ? ? ? ? ? ? ...
strncpy(rez, "Ana", 6); → ... A n a \0 \0 \0 ? ? ? ? ...
```

13

Ugrađene funkcije za operacije nad nizovima znakova `strcat` `<string.h>`

- Konkatenacija (nadovezivanje) nizova znakova: na kraj niza dest dodaje (kopira) sve znakove iz niza src i \0. Funkcija vraća dest.

```
char *strcat(char *dest, const char *src);

char ime[7+1];
char niz[] = "Iva";
strcpy(ime, "Ana");
```

ime → niz →

```
... | A | n | a | \0 | ? | ? | ? | ? | | | I | v | a | \0 | | | | | ...
```

```
strcat(ime, niz);
```

ime → niz →

```
... | A | n | a | I | v | a | \0 | ? | | | I | v | a | \0 | | | | | ...
```

14

Ugrađene funkcije za operacije nad nizovima znakova `strlen` `<string.h>`

- Duljina niza: vraća broj znakova u nizu. Ne broji `\0`.

```
size_t strlen(const char *s);
```

```
char niz[] = "Pero";
```

```
char *p = "Ana";
```

```
char polje[3] = {'I', 'v', 'a'};
```

```
strlen(niz);    → 4
```

```
strlen(p);     → 3
```

```
strlen(polje); → nije poznato gdje se u memoriji nalazi \0
```

15

Ugrađene funkcije za operacije nad nizovima znakova `strcmp` `<string.h>`

- Usporedba nizova: leksikografski uspoređuje nizove `s1` i `s2`.
 - vraća 0 ako su nizovi jednaki
 - vraća cijeli broj < 0 ako je `s1 < s2`
 - vraća cijeli broj > 0 ako je `s1 > s2`

```
int strcmp(const char *s1, const char *s2);
```

```
strcmp("abcd", "abrd");    → -1
```

```
strcmp("abc", "abcd");     → -1
```

```
strcmp("abcd", "abc");     → 1
```

```
strcmp("abcd", "abcc");    → 1
```

```
strcmp("aBc", "abc");      → -1
```

```
strcmp("aBc", "aBc");     → 0
```

16

Ugrađene funkcije za operacije nad nizovima znakova `strncmp` `<string.h>`

- Usporedba nizova: leksikografski uspoređuje **najviše** maxlen znakova u nizovima s1 i s2.
 - vraća 0 ako su podnizovi jednaki
 - vraća cijeli broj < 0 ako je podniz s1 < podniz s2
 - vraća cijeli broj > 0 ako je podniz s1 > podniz s2

```
int strncmp(const char *s1,
            const char *s2,
            size_t maxlen);
```

```
strncmp("abcd", "abrd", 2);    → 0
strncmp("abc", "abcd", 5);     → -1
strncmp("abcd", "abc", 4);     → 1
strncmp("bcd", "abc", 1);      → 1
```

17

Ugrađene funkcije za operacije nad nizovima znakova `strchr` `<string.h>`

- Traženje znaka unutar niza: vraća **pokazivač** na prvi znak vrijednosti c unutar niza znakova s. Ako takav znak ne postoji, vraća NULL.

```
char *strchr(const char *s, int c);
```

```
char niz[] = "Pero";
char *p = "Ana-Marija";
printf("%c", *strchr(niz, 'r')); → r
printf("%s", strchr(p, 'a'));   → a-Marija

strchr(p, 'Z');                → vraća NULL
```

18

Ugrađene funkcije za operacije nad nizovima znakova `strstr` `<string.h>`

- Traženje podniza unutar niza: u nizu s1 pronalazi prvi podniz koji je jednak nizu s2 i vraća pokazivač na prvi znak tog podniza. Ako odgovarajući podniz ne postoji, vraća NULL.

```
char *strstr(const char *s1, const char *s2);
```

```
char niz[] = "Nigdar ni tak bilo da ni nekak bilo";  
printf("%s", strstr(niz, "ni"));
```

→ ni tak bilo da ni nekak bilo

```
strstr(niz, "Tak"); → vraća NULL
```

19

Funkcije nad znakom `<ctype.h>`

- Pretvorba u veliko slovo

```
int toupper(int ch);
```

```
printf("%c", toupper('r')); → R
```

```
printf("%c", toupper('R')); → R
```

```
printf("%c", toupper('3')); → 3
```

- Pretvorba u malo slovo

```
int tolower(int ch);
```

```
printf("%c", tolower('R')); → r
```

```
printf("%c", tolower('r')); → r
```

```
printf("%c", tolower('3')); → 3
```

20

Macro nad znakom

`<ctype.h>`

↪ logička vrijednost

`int isdigit(int c);` znamenka (0-9)

u `<ctype.h>` je definiran macro

```
#define isdigit(c) ((c) >= '0' && (c) <= '9')
```

`int isalpha(int c);` slovo (A-Z ili a-z)

`int isalnum(int c);` slovo (A-Z ili a-z) ili znamenka (0-9)

`int isprint(int c);` znak koji se može ispisati (0x20-0x7E)

`int iscntrl(int c);` kontrolni znak (0x00-0x1F ili 0x7F)

`int isspace(int c);` praznina

`int islower(int c);` slovo (a-z)

`int isupper(int c);` slovo (A-Z)

21

Izdvajanje rečenica iz unešenog teksta

može se preskočiti

Zadatak: Učitati s tipkovnice stranicu teksta (veličine do 4096 znakova). Tekst se sastoji od rečenica odvojenih točkama. Unos teksta se završava unosom riječi GOTOVO. U programu treba ispisati tekst tako da svaka rečenica počne u novom retku.

22

Rješenje - deklaracijski blok

Recenice

može se preskočiti

```
#include <string.h>
#include <stdio.h>
#define MAXTEXT 4096
#define MAXRIJEC 80
#define MAXRECEN 512
int main () {
    char tekst[MAXTEXT+1];
    char rijec[MAXRIJEC+1];
    char recenica[MAXRECEN+1];
    char *pocrec, *tocka;
    int lrijec, ltekst;
```

23

Rješenje - postupak

/* Učitavanje ulaznih podataka */

može se preskočiti

```
strcpy (tekst, "");
ltekst = 0;
while (1) {
    scanf ("%s", rijec);
    if (strcmp (rijec, "GOTOVO") == 0) break;
    lrijec = strlen (rijec);
    if (ltekst + lrijec + 1 > MAXTEXT)break;
    strcat (tekst, rijec);
    strcat (tekst, " ");
    ltekst += lrijec + 1;
}
```

24

Primjer zadavanja teksta

može se preskočiti

```
prva recenica.   nova↵  
recenica.       GOTOVO↵
```

Nakon unosa tekst će biti pohranjen u memoriji računala na slijedeći način:

p	r	v	a	.	r	e	c	e	n	i	c	a	.	n	o	v	a	.	r	e	c	e	n	i	c	a	.	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

25

Rješenje - analiza niza i ispis rezultata

može se preskočiti

```
/* analiza niza i ispis */  
pocrec = tekst;  
do {  
    tocka = strchr (pocrec, '.');  
    if (!tocka) {  
        if (strlen (pocrec) > 0) {  
            printf ("%s\n", pocrec);  
        }  
    } else {  
        strncpy (recenica, pocrec, tocka-pocrec+1);  
        recenica[tocka-pocrec+1] = '\0';  
        printf ("%s\n", recenica);  
        pocrec = tocka+2;  
    }  
} while (tocka);  
  
return 0;  
}
```

26

Prikaz promjene pokazivača

može se preskočiti

p r v a r e c e n i c a . n o v a r e c e n i c a . 0

pocrec

tocka

27

Alternativno rješenje učitavanja - znak po znak:

može se preskočiti

ReceniceZnakPoZnak

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

```
...
```

```
int i; char c;
```

```
i = 0;
```

```
do {
```

```
    c = getche ();
```

```
    if (isprint (c)) tekst[i++] = c;
```

```
    else if (c == '\\b' && i > 0) i--;
```

```
    if (i >= 6 && strcmp (&tekst[i-6],  
        "GOTOVO", 6)== 0) break;
```

```
} while (i < MAXTEXT);
```

```
tekst[i-6] = '\\0';
```

Backspace

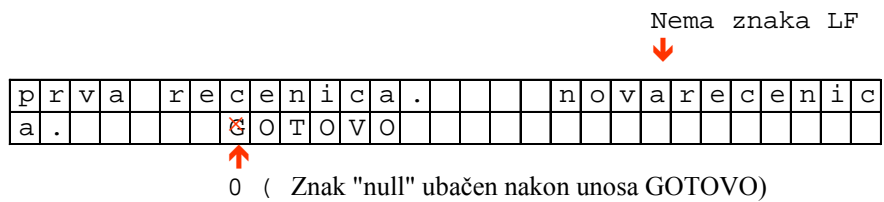
28

Ulazni podaci:

može se preskočiti

```
prva recenica.   nova↵  
recenica.       GOTOVO↵
```

tekst u memoriji računala:



29

Modifikacija da se izbjegne spajanje riječi:

može se preskočiti

```
...  
c = getche ();  
if (c == 0x0d) {  
    c = ' '; printf ("\n");  
}  
...
```

Return

Ispis:

```
prva recenica.  
nova recenica.
```

30

Modifikacija da se izbjegnu suviše praznine:

može se preskočiti

- a) kod ispisa

```
...
for (i=0; recenica[i] == ' '; i++);
printf ("%s\n", &recenica[i]);
...
```

31

Modifikacija da se izbjegnu suviše praznine (ne uzima u obzir \b):

može se preskočiti

- b) kod unosa

```
...
int preskoci; /* zastavica - flag */
i = 0; preskoci = 1;
do {
    c = getche ();
    if(isprint(c) && (!preskoci || c != ' '))
        tekst[i++] = c;
    preskoci = (c == ' ' || c == '.');
    if (i >= 6 && strcmp (&tekst[i-6],
        "GOTOVO", 6) == 0) break;
} while (i < MAXTEXT);
```

32

Učitavanje i ispis podataka

Učitavanje:

`getchar`

`scanf`

`gets`

Ispis:

`putchar`

`printf`

`puts`

33

`getchar`

`<stdio.h>`

- Protip funkcije:
`int getchar(void);`
- Učitava jedan znak.
- Uspješno pročitani znak pretvara u cijeli broj.
- Ako pročita znak koji odgovara kraju datoteke (na DOS-u je to 0x1A ili ^Z, na Unix-u je to 0x04 ili ^D), tada vraća macro vrijednost EOF.
- Ako se pri čitanju dogodi pogreška, vraća EOF.
- macro EOF (end-of-file) je definiran u `<stdio.h>`

34

Primjer: čitati i ispisivati znak po znak

```
IOGetchar
#include <stdio.h>
int main() {
    char c;
    while(1) {
        c = getchar();
        printf("%d ", c);
        if (c == EOF) break;
    }
    return 0;
}
```

Rezultat izvođenja programa:

```
aAB↵
97 65 66 10 /↵
47 10 <Ctrl+Z>↵
-1
```

35

putchar

<stdio.h>

- Prototip funkcije:
`int putchar(int ch);`
- Ispisuje jedan znak.
- Vraća vrijednost uspješno ispisanog znaka ili vraća EOF ukoliko ispis znaka nije uspio.

```
#include <stdio.h>
int main() {
    int i;
    for (i = 'A'; i <= 'Z'; i++)
        putchar(i);
    return 0;
}
→ ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

36

Primjer: Učitati i ispisati niz znakova, pri čemu sva mala slova treba ispisati kao velika. Znakove učitavati dok se ne učitava znak '\n'

```
IOPutchar
#include <stdio.h>
#include <ctype.h>
int main() {
    char slovo[80+1];
    int i, n;

    /* učitavanje retka teksta */
    for (i = 0; (slovo[i] = getchar()) != '\n'; i++);

    /* broj procitanih znakova */
    n = i;

    /* ispis retka, ali velikim slovima */
    for(i = 0; i < n; ++i)
        putchar(toupper(slovo[i]));

    return 0;
}
```

37

Izvođenje programa

Ulazni niz podataka:

Mali auto

Što će se kod učitavanja desiti sa znakom '\n'

slovo →

...	M	a	l	i		a	u	t	o	\n	?	?	?	?	?	?	?	?	...
-----	---	---	---	---	--	---	---	---	---	----	---	---	---	---	---	---	---	---	-----

U polje je upisan i znak '\n', ali brojač *i* nakon završetka petlje ima vrijednost 9, tako da se znak '\n' neće ispisati u drugoj petlji

MALI AUTO

38

Primjer: Učitati i ispisati niz znakova, pri čemu sva mala slova treba ispisati kao velika. Znakove učitavati dok se ne učita EOF

```
for(i = 0; (slovo[i] = getchar()) != EOF; i++);  
n = i;  
for(i = 0; i < n; ++i)  
    putchar(toupper(slovo[i]));
```

```
Mali↵  
auto↵  
vozi↵  
<Ctrl+Z>↵
```

vrijednost n će biti 15

slovo →

...	M	a	l	i	\n	a	u	t	o	\n	v	o	z	i	\n	EOF	?	?	...
-----	---	---	---	---	----	---	---	---	---	----	---	---	---	---	----	-----	---	---	-----

```
MALI  
AUTO  
VOZI
```

Prva tri znaka '\n' učitana su u niz i uzrokuju prijelaz u novi red kod ispisa. EOF se neće ispisati jer je n == 15

39

gets, puts

<stdio.h>

- Učitava znakove u niz *s* dok ne učita '\n' ili znak koji odgovara kraju datoteke (^Z ili ^D). Tada učitani znak '\n' ili učitani oznaku kraja datoteke zamijeni sa znakom '\0'. Vraća *s* ukoliko je učitavanje uspješno. Ukoliko pri čitanju nastupi pogreška ili se kao **prvi** znak pročita oznaka kraja datoteke, vraća NULL

```
char *gets(char *s);
```

- Ispisuje niz znakova *s* i '\n'. Vraća nenegativni cijeli broj u slučaju kada ispis uspije, inače vraća EOF

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

40

Primjer: uzastopno učitavati i ispisivati retke teksta (redak sigurno nije dulji od 80 znakova) dok se ne upiše redak u kojem se pojavljuje tekst DOSTA

```
#include <stdio.h>
#include <string.h>
int main() {
    char redak[80+1];
    while (strstr(gets(redak), "DOSTA") == NULL)
        puts(redak);
    return 0;
}
```

Izvođenje programa:

```
Now is the time↵
Now is the time
for all good men↵
for all good men
sada je DOSTA pisanja↵
```

41

scanf

<stdio.h>

- Prototip funkcije:
`int scanf(const char *format, arg_1, arg_2, ..., arg_n);`
- čita iz "standardne ulazne jedinice" (tipkovnica) u skladu sa zadanim formatom, te obavlja konverziju pročitanih znakova u podatke. Konverzija se obavlja na temelju konverzijskih specifikacija koje su dio formata, a rezultati se pohranjuju na lokacije na koje pokazuju argumenti `arg_1, ..., arg_n`
- argumenti `arg_1, ..., arg_n` su pokazivači i moraju odgovarati konverzijskim specifikacijama po broju, redosljedju i tipu
- funkcija vraća broj uspješno pročitanih podataka ili EOF ukoliko se pri čitanju dogodi pogreška

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

42

Konverzijske specifikacije za `scanf`

`%[širina] [modifikator] tip`

Tip

d	cijeli broj s predznakom (dekadski)
u	cijeli broj bez predznaka (dekadski)
o	cijeli broj bez predznaka (oktalni)
x	cijeli broj bez predznaka (heksadekadski)
e, f, g	broj s pomičnim zarezom, sa ili bez eksponenta (u <code>scanf</code> nema razlike među tipovima e, f, g)
c	jedan znak
s	niz znakova

43

Konverzijske specifikacije za `scanf`

Modifikator (zadaje se opcionalno)

h	uz cjelobrojni tip (d, o, x, u): konverzija u <code>short int</code>
l	uz cjelobrojni tip (d, o, x, u): konverzija u <code>long int</code> uz realni tip (e, f, g): konverzija u <code>double</code>

```
short i, j; int k; long m;
float x; double y, z, w;
scanf("%hd %hx %d %ld %f %le %lg %lf",
      &i, &j, &k, &m, &x, &y, &z, &w);
printf("%d %d %d %d %f %f %f %f",
      i, j, k, m, x, y, z, w);
```

```
1 42 3 4 5.100000 6.100000 7.100000 8.100000
```

44

Konverzijske specifikacije za `scanf`

Širina (zadaje se opcionalno)

- najveći broj znakova koje je dopušteno pročitati uz dotičnu konverzijsku specifikaciju

Primjer:

```
Ana912 3 4567↵
```

```
char ime[20];
int i, j;
scanf("%s %d %d", ime, &i, &j);
ime → "Ana912"    i → 3    j → 4567
scanf("%8s %3d %2d", ime, &i, &j);
ime → "Ana912"    i → 3    j → 45
scanf("%4s %3d %2d", ime, &i, &j);
ime → "Ana9"      i → 12   j → 3
```

45

Format za `scanf`

- u svom jednostavnijem obliku, format se sastoji od konverzijskih specifikacija i bjelina (*white-space characters: blank, tab, newline*).
- jedna ili više bjelina u formatu znači: preskoči 0 ili više bjelina na ulazu, dok ne dođeš do znaka koji nije bjelina
- konverzijska specifikacija znači:
 - u slučaju tipova `d`, `u`, `o`, `x`, `e`, `f`, `g`, `s`, prvo preskoči sve bjeline na ulazu, a zatim pročitaj grupu znakova koji se mogu pretvoriti u odgovarajući podatak. Ukoliko se niti prvi pročitani znak ne može pretvoriti u traženi podatak, funkcija prestaje s čitanjem ulaza i vraća broj podataka koje je do tada uspješno pročitala i obavila konverziju
 - u slučaju tipa `c` pročitaj znak s ulaza (to može biti i bjelina)

46

Primjer:

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

```
38 -15 5.51
+151
```

Preskače bjeline s ulaza (zbog %d). Čita znakove 38 (čita dok god ulaz odgovara specifikaciji %d). 38 pretvara u int kojeg upisuje na adresu &i. Preskače bjeline (zbog bjeline u formatu). -15 pretvara u int, upisuje na &j. Preskače bjeline. Čita znakove 5.51 i pretvara ih u float, upisuje na &x. Preskače bjeline. Čita znakove +151, pretvara u float i upisuje na &y. Ostatak ulaza ostaje nepročitan (npr. sljedeći getchar () bi pročitao znak \n).

```
38 -15 5.510000 151.000000
```

47

Primjer:

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

```
38
-15.012 24+25 7.8
```

Preskače bjeline s ulaza. Čita znakove 38, pretvara u int kojeg upisuje na adresu &i. Preskače bjeline. -15 pretvara u int, upisuje na &j (točka nije pročitana jer ne može biti dio cijelog broja). Čita točku i znakove 012, pretvara u float, upisuje na &x. Preskače bjeline. Čita znakove 24 (+ nije pročitano), pretvara u float i upisuje na &y. Čita znakove +25, pretvara u float i upisuje na &z. Preskače bjeline. Čita znak 7, pretvara u int, zapisuje na adresu &w. Ostatak ulaza ostaje nepročitan (npr. sljedeći getchar () bi pročitao znak '.').

```
38 -15 0.012000 24.000000 25.000000 0.000000
```

48

Primjer:

```
int i, j, rez;  
rez = scanf("%d%d", &i, &j);  
printf("%d %d %d", i, j, rez);
```

12 a8↵

Preskače bjeline s ulaza. Čita znakove 12, pretvara u `int` kojeg upisuje na adresu `&i`. Preskače bjeline. Znak `a` se ne može pretvoriti u `int`, prekida učitavanje, vraća 1. Ostatak ulaza ostaje nepročitan (npr. sljedeći `getchar()` bi pročitao znak `'a'`).

12 -858993460 1

"smeće", jer vrijednost varijable `j` nije učitana

49

Napomena uz konverzijsku specifikaciju `%c`

`%c` bjelinu prihvaća jednako kao bilo koji drugi znak

A B C↵

```
char x, y, z;  
scanf("%c%c%c", &x, &y, &z);    ⇨ x=A, y= , z=B  
scanf("%c %c %c", &x, &y, &z); ⇨ x=A, y=B, z=C  
scanf(" %c %c %c", &x, &y, &z); ⇨ x=A, y=B, z=C
```

A B C↵

```
scanf("%c %c %c", &x, &y, &z); ⇨ x= , y=A, z=B  
scanf(" %c %c %c", &x, &y, &z); ⇨ x=A, y=B, z=C
```

ABC↵

```
scanf(" %c %c %c", &x, &y, &z); ⇨ x=A, y=B, z=C  
scanf("%c%c%c", &x, &y, &z);    ⇨ x=A, y=B, z=C
```

50

Napomena uz konverzijsku specifikaciju %s

%s prestaje učitavati znakove kad naiđe na prvu bjelinu

Ana Marija ↵

```
char ime1[80+1], ime2[80+1];
scanf("%s", ime1);           ⇒ ime1="Ana"
scanf("%2s", ime1);         ⇒ ime1="An"
scanf("%10s", ime1);        ⇒ ime1="Ana"
scanf("%s%s", ime1, ime2);  ⇒ ime1="Ana" ime2="Marija"
```

51

Učitavanje niza znakova koji sadrži bjeline

Kako učitati niz koji sadrži bjeline?

Ana Marija ↵

```
char ime[80+1];
```

```
/* ne preskače bjeline na početku, učitava sve znakove dok ne
dođe do znaka \n */
```

```
scanf("%[^\n]", ime);      ⇒ ime=" Ana Marija "
```

```
/* ne preskače bjeline na početku, učitava sve znakove dok ne
dođe do znaka \n ili učitava 10 znakova*/
```

```
scanf("%10[^\n]", ime);   ⇒ ime=" Ana Mari"
```

52

printf

<stdio.h>

- Prototip funkcije:
`int printf(const char *format, arg_1, arg_2, ..., arg_n);`
- Funkcija obavlja ispis na "standardnu izlaznu jedinicu" (zaslon) u skladu sa zadanim formatom.
- Vrijednosti argumenata `arg_1, ..., arg_n`, formatiraju se u skladu s konverzijskim specifikacijama koje su dio formata.
- Ostali znakovi koji se nalaze u `format` ispisuju se nepromijenjeni
- Funkcija vraća broj uspješno ispisanih bajtova ili EOF ukoliko se pri pisanju dogodi pogreška

Opisane su tek najvažnije mogućnosti funkcije `printf`.

Detaljniji opis funkcije `printf` može se pronaći u gotovo svakom C priručniku.

53

Izgled konverzijskih specifikacija kod funkcije printf

`%[znak][širina][.preciznost]tip`

[znak]

ništa	desno pozicioniranje
praznina	tiska - predznak, a umjesto + predznaka je praznina
-	lijevo pozicioniranje
+	rezultat uvijek počinje s + ili -
#	konverzija na alternativan način ne utječe na <code>c s d i u</code> ispisuje vodeće 0 ispisuje vodeće <code>0x</code> ili <code>0X</code> ispisuje dec. točku i kad nema decimala za <code>e E F</code> ispisuje prateće 0 za <code>g G</code>

54

Konverzijske specifikacije za `printf`

Tip

d	cijeli broj s predznakom (dekadski)
u	cijeli broj bez predznaka (dekadski)
o	cijeli broj bez predznaka (oktalni)
x, X	cijeli broj bez predznaka (heksadekadski), a-f ili A-F
c	jedan znak
s	niz znakova
p	pokazivač
f	brojevi s pomičnim zarezom (float, double) bez eksponenta
e, E	s eksponentom (<i>scientific notation</i>), ispisuje e ili E
g, G	po potrebi sa ili bez eksponenta, ispisuje e ili E

55

Konverzijske specifikacije za `printf`

Širina (zadaje se opcionalno)

- određuje najmanju širinu polja
- 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 (zadaje se opcionalno)

- za tipove `e`, `E`, `f`: određuje broj znamenki iza dec. točke
- za ostale tipove, `d`, `o`, `u`, `c`, `x`, `g`, `G`, `s`: ima drugačije značenje, ovdje se preciznost za te tipove neće razmatrati
- ako se preciznost ne zada, koristi se preciznost po definiciji (npr. za `e`, `E`, `f`, to je šest znamenki iza decimalne točke)

56

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|
|321.000000000000|0.00000012340|7654320128.00000000000|
```

57

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|
```

58

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|
```

59

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|
```

60

Zadatak: Pročitati vrijednosti za $mr \leq 50$ i $ns \leq 10$.
 Pročitati vrijednosti članova dvodimenzionalnog realnog polja
 od mr redaka i ns stupaca. Ispisati pročitano polje, sume
 redaka i sume stupaca te ukupnu sumu u obliku:

Polje A:

!	1	2	3	4	...	Sumr
1	! xxx.x	xxx.x	xxx.x	xxx.x	...	xxx.x
...	!		...			
!						
Sums!	xxx.x	xxx.x	xxx.x	xxx.x	...	xxx.x

Rješenje u pseudokodu

```

čitaj mr i ns dok ne budu ispravni
učitaj realno polje od mr redaka i ns
stupaca
ispisi naslov
ponavljaj za sve retke od 1 do mr
  | izračunaj sumu retka
  | ispiši redak i sumu
  | ispiši crtu
izračunaj sume stupaca i ukupnu sumu
ispisi sume stupaca i ukupnu sumu
kraj
  
```

Rješenje u C-u

📁 SumeRedakaIStupaca

```
#include <stdio.h>
#define MAXR 50
#define MAXS 10
int main() {
    int mr, ns, i, j;
    float a[MAXR][MAXS], sums[MAXS], sumr, ukupno;
    /* čitanje mr i ns dok ne budu ispravni */
    do {
        printf("Upisite vrijednosti za broj redaka i
            stupaca: ");
        scanf ("%d %d",&mr, &ns);
    } while (mr <= 0 || mr > MAXR ||
        ns <= 0 || ns > MAXS);
```

63

Rješenje u C-u, nastavak

```
/* čitanje polja od mr redaka i ns stupaca */
printf("Upisite polje po retcima\n");
for (i=0; i < mr; i++) {
    for (j=0; j < ns; j++) {
        scanf("%f", &a[i][j]);
    }
}
/* ispis naslova */
printf("          Polje A:\n\n");
printf("          !");
for (j=1; j <= ns; j++) printf("%6d",j);
printf("  Sumr\n");
for (j=1; j <= 6*(ns+1)+5; j++) printf("%c",'=');
printf("\n");
```

64

Rješenje u C-u, nastavak

```
/* ponavljaj za sve retke od 1 do mr */
for (i=0; i<mr; i++) {
    sumr = 0;
    for (j=0; j<ns; j++) {
        sumr += a[i][j];
    }
    printf("%3d !",i+1);
    for (j=0; j<ns; j++) printf(" %5.1f", a[i][j]);
    printf(" %5.1f\n", sumr);
    for (j=1; j <= 6 * (ns+1) + 5; j++) printf("-");
    printf("\n");
}
```

65

Rješenje u C-u, nastavak

```
/* izračunaj sume stupaca i ukupnu sumu */
ukupno = 0;
for (j=0; j<ns; j++) {
    sums[j] = 0;
    for (i=0; i<mr; i++) {
        sums[j] += a[i][j];
    }
    ukupno += sums[j];
}
/* ispisi sume stupaca i ukupnu sumu */
printf("Sums!");
for (j=0; j<ns; j++) printf(" %5.1f", sums[j]);
printf(" %5.1f\n",ukupno);

return 0;
}
```

66

Izvođenje programa:

Upisite vrijednosti za broj redaka i stupaca: 2 4↵

Upisite polje po retcima

1 2 3 4↵

0.1 0.2 -0.1 -0.2↵

Polje A:

	!	1	2	3	4	Sumr
1	!	1.0	2.0	3.0	4.0	10.0
2	!	0.1	0.2	-0.1	-0.2	0.0
Sums	!	1.1	2.2	2.9	3.8	10.0

67