

Ugrađene funkcije

Pregled ugrađenih funkcija

Matematičke funkcije <math.h>

```
double fabs(double arg);  
double pow(double arg, double exp);
```

Posebne funkcije <stdlib.h>

```
void exit(int stanje);  
void srand(unsigned int sjeme);  
int rand(void);
```

Standardne funkcije <stdio.h>

```
int getchar(void);  
int putchar(int ch);  
char * gets(char * str);  
int puts(const char * z);  
int printf(const char * format, arg1,..., arg n);  
int scanf(const char * format, arg1,..., arg n);
```

Funkcije znak. niza <string.h>

```
char * strlwr(char * str);  
char * strupr(char * str);  
int strlen(const char * str);  
char * strcat(char * str1, const char * str2);  
char * strcpy(char * str1, const char * str2);  
int strcmp(const char * str1, const char * str2);  
char * strchr(const char * str, int ch);  
char * strstr(const char * str1, const char * str2);
```

Znakovne funkcije <ctype.h>

```
int isdigit(int ch);  
int tolower(int ch);  
int toupper(int ch);
```

Ugrađene matematičke funkcije

<code>#include <math.h></code>	
<code>int abs (int x);</code>	$ x $ za int
<code>long labs (long x);</code>	$ x $ za long int
<code>double fabs (double x);</code>	$ x $ za double
<code>double sin (double x);</code>	$\sin x$
<code>double cos (double x);</code>	$\cos x$
<code>double tan (double x);</code>	$\tan x$
<code>double asin (double x);</code>	$\arcsin x$
<code>double acos (double x);</code>	$\arccos x$
<code>double atan (double x);</code>	$\arctan x$

3

Ugrađene matematičke funkcije

<code>double sinh (double x);</code>	$\operatorname{sh} x$
<code>double cosh (double x);</code>	$\operatorname{ch} x$
<code>double tanh (double x);</code>	$\operatorname{th} x$
<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>	$x \bmod y$
<code>double ceil (double x);</code>	zaokr. na gore
<code>double floor (double x);</code>	zaokr. na dolje

4

Ugrađene posebne funkcije

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

void randomize (void);
ili
void srand (unsigned int seed);
int rand (void);    [0, RAND_MAX] ≡ [0, 32767]
```

Jednoliko preslikavanje cijlobrojnog intervala $[a, b]$ u interval $[c, d]$:

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

5

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 () {
    return (float) rand() / (RAND_MAX+1) * 6 + 1;
}

void main () {
    int i, n, brojac[6] = {0};
    printf ("RAND_MAX = %d\n", RAND_MAX);
    printf ("Upisati broj bacanja kocke >");
    scanf ("%d", &n);

    srand ((unsigned) time(NULL));
    // time() vraća broj sekundi od ponoci, 1. siječnja 1970.

    for (i=0; i<n; i++)
        ++brojac[kocka()-1];

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

■ Rezultati:

```
RAND_MAX = 32767
Upisati broj bacanja kocke >1000
1 176
2 147
3 164
4 172
5 171
6 170

RAND_MAX = 32767
Upisati broj bacanja kocke >1000000
1 166705
2 167469
3 166961
4 166661
5 165966
6 166238
```

6

typedef deklaracija

Kreiranje novih imena za tipove podataka:

```
typedef stari_tip novi_tip;
```

npr.

```
typedef unsigned char bajt;  
typedef unsigned int redni_broj;  
typedef char * tekst;  
typedef unsigned long size_t;  
redni_broj i, j;  
bajt velicina;  
tekst red;  
size_t maxlen;
```

7

Znakovni niz

- Definicija znakovnog niza kao polje znakova:

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

npr.

```
char grad[] = "Zagreb";  
char niz[80+1];
```

- Sve operacije nad nizovima mogu se obavljati pomoću funkcija.

8

Ugrađene funkcije (<string.h>)

▪ Kopiranje niza:

```
char * strcpy(char * str1, const char * str2);
```

Primjer:

```
char grad[] = "Zagreb";  
char niz[80+1];  
strcpy (niz, grad);  
printf ("%s %s\n", grad, niz);
```



Zagreb Zagreb

9

Ugrađene funkcije (<string.h>) - kopiranje:

```
char * strncpy(char * str1, const char * str2, size_t mxl);
```

Primjer:

▪ Rezultat:

```
char grad[] = "Zagreb";  
char niz[80+1];  
strcpy (niz, grad);  
printf ("%s %s\n", grad, niz);           Zagreb Zagreb  
  
strcpy (niz, "Osijek");  
printf ("%s %s\n", grad, niz);           Zagreb Osijek  
  
strncpy (niz, grad, 4);  
printf ("%s %s\n", grad, niz);           Zagreb Zagrek  
  
niz[4]='\0';  
printf ("%s %s\n", grad, niz);           Zagreb Zagr
```

10

Ugrađene funkcije (<string.h>)

- Dodavanje ili konkatencija:

```
char * strcat(char * str1, const char * str2);
```

Primjer:

```
...  
strcat (niz, grad);  
printf ("%s %s", grad, niz);
```



Zagreb Zagrzagreb

11

Ugrađene funkcije (<string.h>)

- Duljina niza:

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

Primjer:

```
printf("%d %d", strlen(grad), strlen(niz));
```



6 10

12

Ugrađene funkcije (<string.h>)

- Pretvorba u velika ili mala slova:

```
char *strlwr(char *s);
```

```
char *strupr(char *s);
```

Primjer:

```
strlwr(niz);  
printf ("%s\n", niz);  
strupr(niz);  
printf ("%s\n", niz);
```

↓

zagrzagreb

ZAGRZAGREB

13

Ugrađene funkcije (<string.h>)

- **Usporedba nizova**

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

```
int strcmpi(const char *s1, const char *s2)
```

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

Primjer:

printf ("%d\n", strcmp(grad, niz));	1	(Zagreb ZAGRZAGREB)
printf ("%d\n", strcmp(niz, grad));	-1	(ZAGRZAGREB Zagreb)
strcpy (niz, "ZAGREB");		
printf ("%d\n", strcmp(grad, niz));	1	(Zagreb ZAGREB)
printf ("%d\n", strcmp(niz, grad));	-1	(ZAGREB Zagreb)
printf ("%d\n", stricmp(grad, niz));	0	(Zagreb ZAGREB)

14

Ugrađene funkcije (<string.h>)

- Usporedba dijela nizova:

```
int strncmp(const char * str1,  
            const char * str2,  
            size_t maxlen);
```

```
int strncmpi(const char * str1,  
             const char * str2,  
             size_t maxlen);
```

```
int strnicmp(const char * str1,  
             const char * str2,  
             size_t maxlen);
```

15

Ugrađene funkcije (<string.h>)

- Traženje znaka u nizu

```
char * strchr(const char * str, int ch);
```

Ako se znak ne pronađe vraća se pokazivač na null

Primjer: (grad="Zagreb")

```
printf ("%s\n", strchr(grad, 'g'));  
printf ("%d\n", strchr(grad, 'g') - grad+1);
```



greb

3

16

Ugrađene funkcije (<string.h>)

- **Traženje podniza**

```
char * strstr(const char * str1,  
              const char * str2);
```

Primjer: (niz="ZAGREB")

```
printf ("%s\n", strstr(niz, "AGR"));  
printf ("%d\n", strstr(niz, "AGR") - niz+1);
```



AGREB

2

17

Funkcije nad znakom (<ctype.h>)

- Pretvorba u veliko slovo

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

- Pretvorba u malo slovo

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

18

Makro nad znakom(<ctype.h>)

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



```
#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 (0x7F ili 0x00-0x1F)
```

```
int isspace(int c);  praznina
```

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

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

19

Izdvajanje rečenica iz unešenog teksta

Zadatak:

- Učitati s tipkovnice stranicu teksta .
- Veličina do 4096 znakova.
- Tekst se sastoji od rečenica odvojenih točkama.
- Unos teksta se završava unosom riječi GOTOVO.
- Ispisati tekst tako da svaka rečenica počne u novom retku.

20

Rješenje - deklaracijski blok

 Recenice

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

21

Rješenje - postupak

```
/* Ucitavanje ulaznih podataka */
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;
}
```

22

Primjer unos teksta

```
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
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

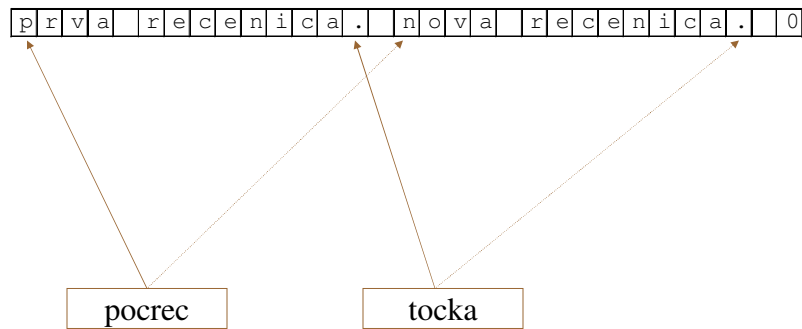
23

Rješenje - analiza niza i ispis rezultata

```
/* 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);
}
```

24

Prikaz promjene pokazivača



25

Izdvajanje rečenica iz unešenog teksta

Zadatak:

- Učitati s tipkovnice stranicu teksta
- Veličina do 4096 znakova.
- Tekst se sastoji od rečenica odvojenih točkama.
- Unos teksta se završava unosom riječi GOTOVO.
- Ispisati tekst tako da svaka rečenica počne u novom retku.
- Vidjeli smo rješenje gdje se učitava riječ po riječ
- Sada ćemo vidjeti modifikacije
 - učitavanje znak po znak
 - izbjegavanje spajanja riječi
 - izbjegavanje suvišnih praznina

26

Alternativno rješenje učitavanja - znak po znak:

ReceniceZnakPoZnak - (rmpz1a.c)

```
#include <ctype.h>
...
int i=0; char c;
do {
    c = getche ();
    if (isprint(c)) tekst[i++] = c;
    else if (c=='\b' && i>0) i--;
    if (i>=6 && strncmp(&tekst[i-6], "GOTOVO", 6) ==0) {
        tekst[i-6] = '\0';
        break;
    }
} while (i < MAXTEXT);
```

'\b' - backspace

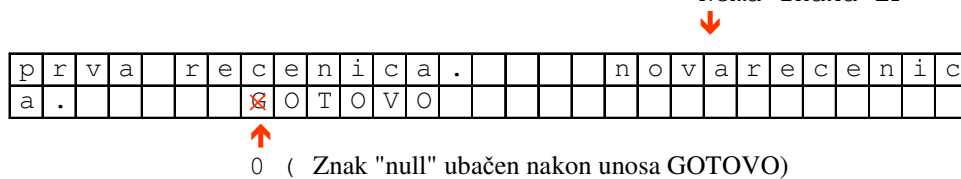
27

Ulazni podaci (rmpz1a.c):

prva recenica. nova,
recenica. GOTOVO,

tekst u memoriji računala:

Nema znaka LF



28

Modifikacija da se izbjegne spajanje riječi (rmpz1b.c):

```
/* Ucitavanje
do {
    c = getche ();
    if (c == 0x0d) {
        c = ' ';
        printf ("\n");
    }
    if (isprint(c) tekst[i++] = c;
    else if (c == '\b' && i > 0) i--;
    if (i>=6 && strcmp(&tekst[i-6], "GOTOVO", 6) ==0) {
        tekst[i-6] = '\0';
        break;
    }
} while (i < MAXTEXT);
```

CR - Return

LF – nova linija

Ispis (rmpz1b.c):
prva recenica.
nova recenica.

Modifikacija da se izbjegne i više praznina (rmpz1b.c):

```
do {
    c = getche ();
    if (c == 0x0d) {
        c = ' ';
        printf ("\n");
    }
    if (isprint(c) && (!preskoci || c!=' ')) {
        tekst[i++] = c;
    }
    else if (c == '\b' && i > 0) i--;
    preskoci = (c == ' ' || c == '.');
    if (i>=6 && strcmp(&tekst[i-6], "GOTOVO", 6) == 0) {
        tekst[i-6] = '\0';
        break;
    }
} while (i < MAXTEXT);
```

/* Ucitavanje

Ispis (rmpz1c.c):
prva recenica.
nova recenica.

Modifikacija da se izbjegne više praznina u ispisu (rzpz1b1.c):

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

```
Ispis (rzpz1c.c):
prva recenica.
nova recenica.
```

Ugrađene funkcije za učitavanje i ispis podataka

Standardne IO funkcije <stdio.h>

Učitavanje:

```
int getchar(void);
char * gets(char * str);
int scanf(const char * format, arg1,..., arg n);
```

Ispis:

```
int putchar(int ch);
int puts(const char * z);
int printf(const char * format, arg1,..., arg n);
```


Funkcija `getchar`

- Funkcija `int getchar(void);`
iz `<stdio.h>` koristi se za unos znak po znak.
Uspješno pročitani znak pretvara u cijeli pozitivni broj
Nakon nailaska na kraj datoteke (^Z za DOS, ^D za Unix) ili na pogrešku vraća EOF (tj. -1).
- Funkcija se poziva na sljedeći način:

```
znakovna_varijabla = getchar();
```

Znakovi su *baferirani* sve dok se ne unese **Return**

Ovo je bolje riješeno u nestandardnoj funkciji `getche`

Objekti vraćaju `int` - to može biti problem kod implementacije kada je `char` u stvari `unsigned`!

33

Primjer čitanja znak po znak s `getchar()`:

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

Rezultat izvođenja programa:

```
aA
97='a' | 65='A' | 10='
'|
32=' '| 47='/' | 10='
'|
-1=' '|
```

34

Funkcija putchar

- Funkcija `int putchar(int ch);` iz `<stdio.h>` koristi se za ispis znak po znak
 - vrati vrijednost ispisanog znaka ili **EOF** za slučaj greške
 - pretvara `int` argument u `unsigned char` prije ispisa
- `putchar` funkcija se poziva na sljedeći način:
`putchar(znakovna_varijabla);`

Primjer:

```
char c;  
.  
.  
.  
putchar(c);
```

35

Modifikacija: Kriterij za zaustavljanje petlje je EOF

 iopcr.c

```
#include <stdio.h>  
  
void main() {  
    char slovo[80];  
    int i, n; /* učitavanje linije teksta */  
    for(i=0; (slovo[i] = getchar()) != '\n';  
        i++);  
    n = i; /* broj procitanih znakova */  
    /* ispis linije ali velikim slovima */  
    for(i=0; i<=n; ++i)  
        putchar(toupper(slovo[i]));  
    printf("#");  
}
```

Mali auto.
↓
MALI AUTO
#

U niz se unosi i znak '\n' (Return)!

Primjer: Učitati niz slova i ispisati ih kao velika

iopcr.c

```
#include <stdio.h>
void main() {
    char slovo[80];
    int i, n;
    for(i=0; (slovo[i] = getchar()) != EOF; i++);
    n = i;
    for(i=0; i<n; ++i)
        putchar(toupper(slovo[i]));
}
```

Uneseni znak '\n' (Return)
koristi kod ispisa!

```
mali auto↵
dobro↵
vozi.↵
^Z↵
↓
MALI AUTO
DOBRO
VOZI.
```

Funkcija scanf

```
#include <stdio.h>
...
int scanf(const char *format, arg1, ..., argn);
```

scanf vraća broj uspješno obrađenih ulaznih vrijednosti koje povezuje s navedenim argumentima, ili 0 ukoliko unesena vrijednost ne odgovara ili EOF za ulaznu grešku prije pridruživanja

Argumenti moraju odgovarati po broju, redoslijedu i tipu formatskim specifikacijama.

Argumenti su pokazivači:

- za polje se navodi njegovo ime (pokazivač na nulti član), a
- za obične varijable se navodi adresa (npr. **&x**).

Primjer korištenja funkcije `scanf`

```
#include <stdio.h>
void main() {
    char naziv[20];
    int sifra;
    float cijena;
    scanf("%s %d %f", naziv, &sifra, &cijena);
```

. . .

Ulazni podaci bi se mogli zadati, na primjer, na slijedeće načine:

1.	2.	3.	4.
Indeks 32145 10.5	Indeks 32145 10.5	Indeks 32145 10.5	Indeks 32145 10.5

Za naredbu:

```
scanf("%s%d%f", naziv, &sifra, &cijena);
```

Samo 2. unos bi bio ispravan!

39

Primjer: Učitavanje niza uz kontrolu postojanja znaka u skupu znakova.

```
char slovo[80];
scanf("%[ ABCDEFGHIJKLMNOPQRSTUVWXYZ]", slovo);
printf("%s\n", slovo);
```

Ulazni niz podataka:

MALi Auto dobro vozi.↓

Ispis:

MALi A

Znak 'i' nije u definiranom skupu i izazvat će prekid.

Gornji unos može se napisati i kao:

```
scanf("%[ A-Z]", slovo);
```

40

Primjer: Učitavanje znakovnog niza sve dok se ne pojavi znak '\n'

```
char slovo[80];  
...  
scanf("%[^\\n]", slovo);
```

Čita dok ne bude unešen znak \n

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

Čita dok ne bude unešen neki od navedenih znakova: , ! ? - .

41

Izgled formatske specifikacije kod funkcije `scanf`

`%[širina][modifikator]tip`

	tip
[širina]	d cijeli broj s predznakom, int
n broj mjesta predviđenih za ulazni podatak	o oktalni broj, unsigned int
	x heksadecimalni broj
	u cijeli broj bez predznaka
[modifikator]	e, f, g broj s pomičnim zarezom, može i E ili G
* čita i ignorira input	c jedan znak ili n znakova (čita i praznine, bez automatskog '\\0' mora biti dovoljno mjesta)
h cjelobrojni argument je short	s znakovni niz (bez razmaka, '\\0' se automatski dodaje na kraj)
l cjelobrojni argument je long ili realni argument je double	[...] kontrola unosa u znakova
	%[0-9] unos brojeva
	%[01] unos praznine 0 i 1
	%[^! ? -] ignoriranje unosa znakova ! ? -

42

Odvajanje formatskih specifikacija kod `scanf`

Pojedinačne formatske specifikacije se mogu pisati neposredno jedna za drugom ili se mogu razdvojiti **prazninom**, **Tab**-om ili znakom `\n`.

Obično se koristi praznina.

`c` - format prihvaća bilo koji unešeni znak.

Primjer za učitavanje znakova, unos: `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`

43

Funkcija `printf`

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

```
...
```

```
int printf(const char *format, arg1, ..., argn);
```

printf kao rezultat daje broj bajtova ispisanih na standardnoj izlaznoj jedinici (`stdout`).

Argumenti mogu biti varijable, imena polja ili izrazi.

44

Primjer korištenja funkcije `printf`

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

void main() {
    float i = 6.0, j = 3.0;
    printf("%f%f    %f %f", i, j, i+.008, sqrt(i+j));
}
```

Izgled ispisa:

```
6.0000003.000000    6.008000 3.000000
```

45

Primjer korištenja funkcije `printf`

```
#include <stdio.h>
int main() {
    double x = 5000.0, y = 0.0025;
    printf("%.2f %.4f %.2f %.2f\n", x, y, x*y, x/y);
    printf("%.2e %.2e %.2e %.2e\n", x, y, x*y, x/y);
    printf("%.2g %.2g %.2g %.2g\n", x, y, x*y, x/y);
    printf("%g %g %g %g\n", x, y, x*y, x/y);
}
```

Ispis:

```
.24f 5000.00 0.0025 12.50 2000000.00
.2e  5.00e+003 2.50e-003 1.25e+001 2.00e+006
.2g  5e+003 0.0025 13 2e+006
g    5000 0.0025 12.5 2e+006
```

46

Izgled formatske specifikacije kod funkcije `printf`

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

<u>[znak]</u>		<u>[širina]</u>	
ništa	desno pozicioniranje (" <code>%d</code> ")	<i>n</i>	najmanje <i>n</i> mjesta
□	ispisuje '-' predznak, a umjesto '+' predznaka ispisuje se praznina (" <code>% f</code> ")	<u>tip</u>	
-	lijevo pozicioniranje (" <code>%-e</code> ")	c	znak
+	rezultat uvijek počinje s '+' ili '-' (" <code> %+d</code> ")	d	cijeli broj s predznakom
0	dodavanje nula s lijeve strane (" <code>%0f</code> ")	u	cijeli broj bez predznaka
#	konverzija na alternativan način za <code>int</code> i <code>realne</code>	o	oktalni broj bez predznaka bez vodećih nula
		e	broj s pomičnim zarezom prikazan u eksponencijalnom obliku
		f	broj s pomičnim zarezom
		g	broj s pomičnim zarezom (e ili f oblika ovisno o vrijednosti)
		s	znakovni niz
		x	heksadecimalni broj bez oznake 0x ispred rezultata
<u>[.preciznost]</u>			
ništa	preciznost po definiciji		
.0	preciznost po definiciji za <code>d</code> , <code>o</code> , <code>u</code> i <code>x</code> ; bez decimalne točke za <code>e</code> , <code>f</code>		
.n	najviše <i>n</i> znakova		

47

Primjeri korištenja funkcije `printf`

```
#include <stdio.h>
void main() {
    char tekst[]="studomat";
    printf("%5s %10s %10.5s %.5s",
           tekst,tekst,tekst,tekst);
}
```

Ispis:

```
studomat      studomat      studo studo
```

48

Primjeri korištenja funkcije `printf`

```
#include <stdio.h>
void main() {
    int i = 123;    float x = 12.0, y = -3.3;
    printf(":%6d %7.0f %10.1e:\n", i, x, y);
    printf(":%-6d %-7.0f %-10.1e:\n", i, x, y);
    printf(":%+6d %+7.0f %+10.1e:\n", i, x, y);
    printf(":%-+6d %-+7.0f %-+10.1e:\n", i, x, y);
    printf(":%7.0f %#7.0f %7g %#7g:\n", x, x, y, y);
}
```

Ispis:

-alternativna konverzija
-kombiniranje znakova

```
:   123      12  -3.3e+000:
:123      12      -3.3e+000 :
:  +123     +12  -3.3e+000:
: +123    +12      -3.3e+000 :
:      12      12.   -3.3 -3.30000:
```

49

Primjeri korištenja funkcije `printf`

```
#include <stdio.h>
void main() {
    int i=1234, j=01777, k=0xa08c;
    printf(":%8u %8o %8x:\n", i, j, k);
    printf(":%-8u %-8o %-8x:\n", i, j, k);
    printf(":%#8u %#8o %#8X:\n", i, j, k);
    printf(":%08u %08o %08X:\n", i, j, k);
}
```

Ispis:

-ništa alternativno
-nule ispred

```
:   1234      1777      a08c:
:1234      1777      a08c   :
:   1234      01777      0XA08C:
:00001234 00001777 0000A08C:
```

50

Funkcije `gets` i `puts`

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

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

Ove funkcije nude jednostavnu zamjenu za `scanf` i `printf` kada se radi o znakovnim nizovima.

Unos pomoću `gets` funkcije završava s *Enter*.


Funkcija `gets` stavlja `'\0'` na kraju unesenoga, uzima praznine i ne kontrolira koliko je karaktera uneseno (što je potencijalno opasno)!

51

Primjer korištenja funkcija `gets` i `puts`

```
#include <stdio.h>  
void main() {  
    char red[80];  
    gets(red);  
    puts(red);  
}
```

Još jedan primjer:

 ElementarnaEnkripcija

52

Zadatak: Pročitati vrijednosti za **mr** ≤ 50 i **ns** ≤ 10 .
 Pročitati vrijednosti članova 2D 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
ispiši naslov
ponavljaj za sve retke od 1 do mr
| izračunaj sumu retka
| ispiši redak i sumu
| ispiši crt
izračunaj sume stupaca i ukupnu sumu
ispisi sume stupaca i ukupnu sumu
kraj
```

Rješenje u C-u - SumeRedakaIStupaca.c

```
#include <stdio.h>
#define MAXR 50
#define MAXS 10
void main() {
    int mr, ns, i, j;
    float a[MAXR][MAXS], sums[MAXS], sumr, ukupno;
    do {
        printf("Upisi broj redaka i stupaca: ");
        scanf ("%d %d",&mr, &ns);
    } while (mr<=0 || mr>MAXR || ns<=0 || ns>MAXS);
    printf("Upisi polje po retcima\n");
    for (i=0; i<mr; i++)
        for (j=0; j<ns; j++) scanf("%f", &a[i][j]);
```

Rješenje u C-u, nastavak

```
/* ispis naslova */
printf("\n\nPolje A:\n\n    !");
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")
/* ispis elemenata redaka, sume svakog retka i crte*/
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");
}
```

56

Rješenje u C-u, nastavak

```
/* racunanje sume stupaca i ukupne sume */
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];
}

/* ispis sume stupaca i ukupne sume */
printf("SumS!");
for (j=0; j<ns; j++) printf(" %5.1f", sums[j]);
printf(" %5.1f\n\n", ukupno);
}
```

57

Rezultat izvođenja

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	

58