

# Lekcija 8 – Karakteri i stringovi

## Pregled

- 8.1 Uvod
- 8.2 Osnovni pojmovi o stringovima i karakterima
- 8.3 Biblioteka za obradu karaktera
- 8.4 Funkcije konverzije stringova
- 8.5 Standardne ulazno-izlazne funkcije
- 8.6 Manipulacija stringovima (biblioteka <string.h>)
- 8.7 Funkcije za poređenje stringova
- 8.8 Funkcije za pretraživanje stringova
- 8.9 Memorijske funkcije
- 8.10 Ostale funkcije u biblioteci <string.h>

## Ciljevi lekcije

- U ovoj lekciji:
  - Naučićete kako da koristite biblioteku `cctype`.
  - Koristićete ulazne i izlazne funkcije sa karakterima i stringovima – biblioteka `stdio`.
  - Upotrebljavaćete funkcije za konverziju stringova – biblioteka `stdlib`.
  - Naučićete sve o procesiranju stringova – biblioteka `string`.
  - Shvatićete značaj biblioteka funkcija kao načina za dostizanje ponovne upotrebljivosti softvera (software reusability).

## 8.1 Uvod

- Uvešćemo neke standardne bibliotečke funkcije
  - Olakšavaju procesiranje stringova i karaktera
  - Programi mogu obrađivati karaktere, stringove, linije teksta i blokove memorije
- Ove tehnike se koriste za izradu
  - Word procesora
  - Softvera za podešavanje izgleda stranica teksta (page layout software)
  - Programa za unošenje teksta (typesetting programs)

## 8.2 Osnovni pojmovi o stringovima i karakterima

- Karakteri
  - Svi programi su kreirani od karaktera
    - Svaki program je niz karaktera grupisanih na određeni način (sa nekim značenjem)
  - Konstante karakteri
    - `int` vrijednost predstavljena kao karakter u jednostrukim navodnicima
    - `'z'` predstavlja cjelobrojnu vrijednost za `z`
- Stringovi
  - Niz karaktera koji se tretira kao cjelina
    - Mogu sadržati slova, cifre i specijalne karaktere (`*`, `/`, `$`)
  - String literali (string konstante) – napisani sa dvostrukim navodnicima
    - `"Hello"`
  - Stringovi su nizovi karaktera
    - String je pokazivač na prvi karakter
    - Vrijednost stringa je adresa prvog karaktera

## 8.2 Osnovni pojmovi o stringovima i karakterima

- Definicija stringa
  - Definišemo ih kao nizove karaktera ili promjenljive tipa `char *`

```
char color[] = "blue";  
char *colorPtr = "blue";
```
  - Zapamtite da stringovi predstavljaju nizove karaktera koji završavaju sa `'\0'`
    - `color` ima 5 elemenata
- Ulazne operacije sa stringovima
  - koristite `scanf`

```
scanf("%s", word);
```

    - Kopira se ulaz u `word[]`
    - Nije potrebno `&` (jer je string već pokazivač)
  - Ne zaboravite prostor u nizu za `'\0'`

## 8.3 Biblioteka za obradu karaktera

- Biblioteka `<ctype.h>`
  - Sadrži funkcije koje testiraju i manipulišu sa karakterima
  - Svaka funkcija ima argument tipa karakter (tj. `int`) ili EOF (end of file indikator) kao argument
- Sledeći slajd sadrži tabelu svih funkcija u `<ctype.h>`

## 8.3 Biblioteka za obradu karaktera

Prototip	Opis
<code>int isdigit( int c );</code>	Vraća <code>true</code> ako je <code>c</code> cifra i <code>false</code> inače.
<code>int isalpha( int c );</code>	Vraća <code>true</code> ako je <code>c</code> slovi i <code>false</code> inače.
<code>int isalnum( int c );</code>	Vraća <code>true</code> ako je <code>c</code> cifra ili slovo i <code>false</code> inače.
<code>int isxdigit( int c );</code>	Vraća <code>true</code> ako je <code>c</code> heksadecimalni karakter i <code>false</code> inače.
<code>int islower( int c );</code>	Vraća <code>true</code> ako je <code>c</code> malo slovo (lowercase) i <code>false</code> inače.
<code>int isupper( int c );</code>	Vraća <code>true</code> ako je <code>c</code> veliko slovo (uppercase); <code>false</code> inače.
<code>int tolower( int c );</code>	Ako je <code>c</code> veliko slovo, <code>tolower</code> vraća ekvivalentno malo slovo. Inače, <code>tolower</code> vraća nepromijenjeni argument.
<code>int toupper( int c );</code>	Ako je <code>c</code> malo slovo, <code>toupper</code> vraća <code>c</code> kao veliko slovo. Inače, <code>toupper</code> vraća nepromijenjeni argument.
<code>int isspace( int c );</code>	Vraća <code>true</code> ako je <code>c</code> bjelina (white-space character) —newline ( <code>'\n'</code> ), space ( <code>' '</code> ), form feed ( <code>'\f'</code> ), carriage return ( <code>'\r'</code> ), horizontal tab ( <code>'\t'</code> ) ili vertical tab ( <code>'\v'</code> )— i <code>false</code> inače
<code>int iscntrl( int c );</code>	Vraća <code>true</code> ako je <code>c</code> kontrolni karakter i <code>false</code> inače.
<code>int ispunct( int c );</code>	Vraća <code>true</code> ako je <code>c</code> karakter koji se može štampati, a nije space, cifra ili slovo i <code>false</code> inače.
<code>int isprint( int c );</code>	Vraća <code>true</code> value ako je <code>c</code> karakter koji se može štampati,uključujući space i <code>false</code> inače.
<code>int isgraph( int c );</code>	Vraća <code>true</code> ako je <code>c</code> karakter koji se može štampati, a nije space ( <code>' '</code> ) i <code>false</code> inače.



## Outline



### fig08\_02.c (Part 1 of 2)

```
1  /* Fig. 8.2: fig08_02.c
2      Using functions isdigit, isalpha, isalnum, and isxdigit */
3  #include <stdio.h>
4  #include <ctype.h>
5
6  int main()
7  {
8      printf( "%s\n%s%s\n%s%s\n\n", "According to isdigit: ",
9          isdigit( '8' ) ? "8 is a " : "8 is not a ", "digit",
10         isdigit( '#' ) ? "# is a " : "# is not a ", "digit" );
11
12     printf( "%s\n%s%s\n%s%s\n%s%s\n\n",
13         "According to isalpha:",
14         isalpha( 'A' ) ? "A is a " : "A is not a ", "letter",
15         isalpha( 'b' ) ? "b is a " : "b is not a ", "letter",
16         isalpha( '&' ) ? "& is a " : "& is not a ", "letter",
17         isalpha( '4' ) ? "4 is a " : "4 is not a ", "letter" );
18
19     printf( "%s\n%s%s\n%s%s\n%s%s\n\n",
20         "According to isalnum:",
21         isalnum( 'A' ) ? "A is a " : "A is not a ",
22         "digit or a letter",
23         isalnum( '8' ) ? "8 is a " : "8 is not a ",
24         "digit or a letter",
25         isalnum( '#' ) ? "# is a " : "# is not a ",
26         "digit or a letter" );
27
```





## Outline



fig08\_02.c (Part 2 of 2)

```
28 printf( "%s\n%s%s\n%s%s\n%s%s\n%s%s\n",
29         "According to isxdigit:",
30         isxdigit( 'F' ) ? "F is a " : "F is not a ",
31         "hexadecimal digit",
32         isxdigit( 'J' ) ? "J is a " : "J is not a ",
33         "hexadecimal digit",
34         isxdigit( '7' ) ? "7 is a " : "7 is not a ",
35         "hexadecimal digit",
36         isxdigit( '$' ) ? "$ is a " : "$ is not a ",
37         "hexadecimal digit",
38         isxdigit( 'f' ) ? "f is a " : "f is not a ",
39         "hexadecimal digit" );
40
41 return 0; /* indicates successful termination */
42
43 } /* end main */
```

**Program Output**

According to isdigit:

8 is a digit

# is not a digit

According to isalpha:

A is a letter

b is a letter

& is not a letter

4 is not a letter

According to isalnum:

A is a digit or a letter

8 is a digit or a letter

# is not a digit or a letter

According to isxdigit:

F is a hexadecimal digit

J is not a hexadecimal digit

7 is a hexadecimal digit

\$ is not a hexadecimal digit

f is a hexadecimal digit



## Outline



### fig08\_03.c (Part 1 of 2)

```
1  /* Fig. 8.3: fig08_03.c
2      Using functions islower, isupper, tolower, toupper */
3  #include <stdio.h>
4  #include <ctype.h>
5
6  int main()
7  {
8      printf( "%s\n%s%s\n%s%s\n%s%s\n\n",
9              "According to islower:",
10             islower( 'p' ) ? "p is a " : "p is not a ",
11             "lowercase letter",
12             islower( 'P' ) ? "P is a " : "P is not a ",
13             "lowercase letter",
14             islower( '5' ) ? "5 is a " : "5 is not a ",
15             "lowercase letter",
16             islower( '!' ) ? "! is a " : "! is not a ",
17             "lowercase letter" );
18
19     printf( "%s\n%s%s\n%s%s\n%s%s\n\n",
20             "According to isupper:",
21             isupper( 'D' ) ? "D is an " : "D is not an ",
22             "uppercase letter",
23             isupper( 'd' ) ? "d is an " : "d is not an ",
24             "uppercase letter",
25             isupper( '8' ) ? "8 is an " : "8 is not an ",
26             "uppercase letter",
27             isupper( '$' ) ? "$ is an " : "$ is not an ",
28             "uppercase letter" );
29
```



## Outline



### fig08\_03.c (Part 2 of 2)

```
30     printf( "%s%c\n%s%c\n%s%c\n%s%c\n",
31             "u converted to uppercase is ", toupper( 'u' ),
32             "7 converted to uppercase is ", toupper( '7' ),
33             "$ converted to uppercase is ", toupper( '$' ),
34             "L converted to lowercase is ", tolower( 'L' ) );
35
36     return 0; /* indicates successful termination */
37
38 } /* end main */
```

According to islower:  
p is a lowercase letter  
P is not a lowercase letter  
5 is not a lowercase letter  
! is not a lowercase letter

According to isupper:  
D is an uppercase letter  
d is not an uppercase letter  
8 is not an uppercase letter  
\$ is not an uppercase letter

u converted to uppercase is U  
7 converted to uppercase is 7  
\$ converted to uppercase is \$  
L converted to lowercase is l

### Program Output



## Outline



### fig08\_04.c (Part 1 of 2)

```
1  /* Fig. 8.4: fig08_04.c
2     Using functions isspace, iscntrl, ispunct, isprint, isgraph */
3  #include <stdio.h>
4  #include <ctype.h>
5
6  int main()
7  {
8      printf( "%s\n%s%s%s\n%s%s%s\n%s%s\n\n",
9          "According to isspace:",
10         "Newline", isspace( '\n' ) ? " is a " : " is not a ",
11         "whitespace character", "Horizontal tab",
12         isspace( '\t' ) ? " is a " : " is not a ",
13         "whitespace character",
14         isspace( '%' ) ? "% is a " : "% is not a ",
15         "whitespace character" );
16
17     printf( "%s\n%s%s%s\n%s%s\n\n", "According to iscntrl:",
18         "Newline", iscntrl( '\n' ) ? " is a " : " is not a ",
19         "control character", iscntrl( '$' ) ? "$ is a " :
20         "$ is not a ", "control character" );
21
```



```
22 printf( "%s\n%s%s\n%s%s\n%s%s\n\n",
23         "According to ispunct:",
24         ispunct( ';' ) ? "; is a " : "; is not a ",
25         "punctuation character",
26         ispunct( 'Y' ) ? "Y is a " : "Y is not a ",
27         "punctuation character",
28         ispunct( '#' ) ? "# is a " : "# is not a ",
29         "punctuation character" );
30
31 printf( "%s\n%s%s\n%s%s%s\n\n", "According to isprint:",
32         isprint( '$' ) ? "$ is a " : "$ is not a ",
33         "printing character",
34         "Alert", isprint( '\a' ) ? " is a " : " is not a ",
35         "printing character" );
36
37 printf( "%s\n%s%s\n%s%s%s\n", "According to isgraph:",
38         isgraph( 'Q' ) ? "Q is a " : "Q is not a ",
39         "printing character other than a space",
40         "Space", isgraph( ' ' ) ? " is a " : " is not a ",
41         "printing character other than a space" );
42
43 return 0; /* indicates successful termination */
44
45 } /* end main */
```



## Program Output

According to isspace:  
Newline is a whitespace character  
Horizontal tab is a whitespace character  
% is not a whitespace character

According to iscntrl:  
Newline is a control character  
\$ is not a control character

According to ispunct:  
; is a punctuation character  
Y is not a punctuation character  
# is a punctuation character

According to isprint:  
\$ is a printing character  
Alert is not a printing character

According to isgraph:  
Q is a printing character other than a space  
Space is not a printing character other than a space

## 8.4 Funkcije za konverziju stringova

- Funkcije za konverziju
  - U biblioteci `<stdlib.h>` (opšte pomoćne funkcije)
- Konvertuju stringove cifara u cijele i realne (floating-point) vrijednosti

Prototip	Opis funkcije
<code>double atof( const char *nPtr );</code>	Konvertuje string <code>nPtr</code> u <code>double</code> .
<code>int atoi( const char *nPtr );</code>	Konvertuje string <code>nPtr</code> u <code>int</code> .
<code>long atol( const char *nPtr );</code>	Konvertuje string <code>nPtr</code> u <code>long int</code> .
<code>double strtod( const char *nPtr, char **endPtr );</code>	Konvertuje string <code>nPtr</code> u <code>double</code> .
<code>long strtol( const char *nPtr, char **endPtr, int base );</code>	Konvertuje string <code>nPtr</code> u <code>long</code> .
<code>unsigned long strtoul( const char *nPtr, char **endPtr, int base );</code>	Konvertuje string <code>nPtr</code> u <code>unsigned long</code> .



**fig 08\_06.c**

```
1  /* Fig. 8.6: fig08_06.c
2     Using atof */
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  int main()
7  {
8     double d; /* variable to hold converted string */
9
10    d = atof( "99.0" );
11
12    printf( "%s%.3f\n%s%.3f\n",
13           "The string \"99.0\" converted to double is ", d,
14           "The converted value divided by 2 is ",
15           d / 2.0 );
16
17    return 0; /* indicates successful termination */
18
19 } /* end main */
```

The string "99.0" converted to double is 99.000  
The converted value divided by 2 is 49.500

**Program Output**



```
1  /* Fig. 8.7: fig08_07.c
2      Using atoi */
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  int main()
7  {
8      int i; /* variable to hold converted string */
9
10     i = atoi( "2593" );
11
12     printf( "%s%d\n%s%d\n",
13             "The string \"2593\" converted to int is ", i,
14             "The converted value minus 593 is ", i - 593 );
15
16     return 0; /* indicates successful termination */
17
18 } /* end main */
```

```
The string "2593" converted to int is 2593
The converted value minus 593 is 2000
```

**Program Output**



```
1  /* Fig. 8.8: fig08_08.c
2      Using atol */
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  int main()
7  {
8      long l; /* variable to hold converted string */
9
10     l = atol( "1000000" );
11
12     printf( "%s%ld\n%s%ld\n",
13         "The string \"1000000\" converted to long int is ", l,
14         "The converted value divided by 2 is ", l / 2 );
15
16     return 0; /* indicates successful termination */
17
18 } /* end main */
```

```
The string "1000000" converted to long int is 1000000
The converted value divided by 2 is 500000
```

**Program Output**



```
1  /* Fig. 8.9: fig08_09.c
2      Using strtod */
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  int main()
7  {
8      /* initialize string pointer */
9      const char *string = "51.2% are admitted";
10
11     double d;          /* variable to hold converted sequence */
12     char *stringPtr; /* create char pointer */
13
14     d = strtod( string, &stringPtr );
15
16     printf( "The string \"%s\" is converted to the\n", string );
17     printf( "double value %.2f and the string \"%s\"\n", d, stringPtr );
18
19     return 0; /* indicates successful termination */
20
21 } /* end main */
```

The string "51.2% are admitted" is converted to the double value 51.20 and the string "% are admitted"

Program Output



## fig08\_10.c

```
1  /* Fig. 8.10: fig08_10.c
2      Using strtol */
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  int main()
7  {
8      const char *string = "-1234567abc"; /* initialize string pointer */
9
10     char *remainderPtr; /* create char pointer */
11     long x;              /* variable to hold converted sequence */
12
13     x = strtol( string, &remainderPtr, 0 );
14
15     printf( "%s\\\"%s\\\"\\n%s%ld\\n%s\\\"%s\\\"\\n%s%ld\\n",
16           "The original string is ", string,
17           "The converted value is ", x,
18           "The remainder of the original string is ",
19           remainderPtr,
20           "The converted value plus 567 is ", x + 567 );
21
22     return 0; /* indicates successful termination */
23
24 } /* end main */
```

```
The original string is "-1234567abc"
The converted value is -1234567
The remainder of the original string is "abc"
The converted value plus 567 is -1234000
```

## Program Output



## Outline



### fig08\_11.c

```
1  /* Fig. 8.11: fig08_11.c
2     Using strtoul */
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  int main()
7  {
8     const char *string = "1234567abc"; /* initialize string pointer */
9     unsigned long x;    /* variable to hold converted sequence */
10    char *remainderPtr; /* create char pointer */
11
12    x = strtoul( string, &remainderPtr, 0 );
13
14    printf( "%s\\\"%s\\\"\\n%s%lu\\n%s\\\"%s\\\"\\n%s%lu\\n",
15           "The original string is ", string,
16           "The converted value is ", x,
17           "The remainder of the original string is ",
18           remainderPtr,
19           "The converted value minus 567 is ", x - 567 );
20
21    return 0; /* indicates successful termination */
22
23 } /* end main */
```

### Program Output

```
The original string is "1234567abc"
The converted value is 1234567
The remainder of the original string is "abc"
The converted value minus 567 is 1234000
```

## 8.5 Standardne ulazno-izlazne funkcije

- Funkcije u `<stdio.h>`
- Za manipulisanje karakterima i stringovima

Prototip	Opis
<code>int getchar( void );</code>	Učitava sledeći karakter sa standardnog ulaza i vraća cio broj.
<code>char *gets( char *s );</code>	Učitava karaktere sa standardnog ulaza u niz <code>s</code> do pojave newline ili end-of-file karaktera. Završni null karakter se nadovezuje na niz.
<code>int putchar( int c );</code>	Štampa karakter koji se čuva u <code>c</code> .
<code>int puts( const char *s );</code>	Štampa string <code>s</code> a zatim i newline karakter.
<code>int sprintf( char *s, const char *format, ... );</code>	Isto kao <code>printf</code> , ali se štampa u string <code>s</code> a ne na standardni izlaz (ekran).
<code>int sscanf( char *s, const char *format, ... );</code>	Isto kao <code>scanf</code> , ali se učitava iz stringa <code>s</code> a ne sa standardnog ulaza (tastature).



## Outline



### fig08\_13.c (Part 1 of 2)

```
1  /* Fig. 8.13: fig08_13.c
2     Using gets and putchar */
3  #include <stdio.h>
4
5  int main()
6  {
7     char sentence[ 80 ]; /* create char array */
8
9     void reverse( const char * const sPtr ); /* prototype */
10
11    printf( "Enter a line of text:\n" );
12
13    /* use gets to read line of text */
14    gets( sentence );
15
16    printf( "\nThe line printed backwards is:\n" );
17    reverse( sentence );
18
19    return 0; /* indicates successful termination */
20
21 } /* end main */
22
```





## Outline



fig08\_13.c (Part 1 of 2)

```
23 /* recursively outputs characters in string in reverse order */
24 void reverse( const char * const sPtr )
25 {
26     /* if end of the string */
27     if ( sPtr[ 0 ] == '\0' ) {
28         return;
29     } /* end if */
30     else { /* if not end of the string */
31         reverse( &sPtr[ 1 ] );
32
33         putchar( sPtr[ 0 ] ); /* use putchar to display character */
34     } /* end else */
35
36 } /* end function reverse */
```

Enter a line of text:  
Characters and Strings

The line printed backwards is:  
sgnirtS dna sretcarahC

Enter a line of text:  
able was I ere I saw elba

The line printed backwards is:  
able was I ere I saw elba

Program Output



```
1  /* Fig. 8.14: fig08_14.c
2     Using getchar and puts */
3  #include <stdio.h>
4
5  int main()
6  {
7      char c;           /* variable to hold character input by user */
8      char sentence[ 80 ]; /* create char array */
9      int i = 0;         /* initialize counter i */
10
11     /* prompt user to enter line of text */
12     puts( "Enter a line of text:" );
13
14     /* use getchar to read each character */
15     while ( ( c = getchar() ) != '\n' ) {
16         sentence[ i++ ] = c;
17     } /* end while */
18
19     sentence[ i ] = '\0';
20
21     /* use puts to display sentence */
22     puts( "\nThe line entered was:" );
23     puts( sentence );
24
25     return 0; /* indicates successful termination */
26
27 } /* end main */
```

Enter a line of text:  
This is a test.

The line entered was:  
This is a test.



Outline

**Program Output**



## Outline

fig08\_15.c

```
1  /* Fig. 8.15: fig08_15.c
2     Using sprintf */
3  #include <stdio.h>
4
5  int main()
6  {
7      char s[ 80 ]; /* create char array */
8      int x;        /* define x */
9      double y;     /* define y */
10
11     printf( "Enter an integer and a double:\n" );
12     scanf( "%d%lf", &x, &y );
13
14     sprintf( s, "integer:%6d\ndouble:%8.2f", x, y );
15
16     printf( "%s\n%s\n",
17           "The formatted output stored in array s is:", s );
18
19     return 0; /* indicates successful termination */
20
21 } /* end main */
```

```
Enter an integer and a double:
298 87.375
The formatted output stored in array s is:
integer:   298
double:   87.38
```

Program Output



```
1  /* Fig. 8.16: fig08_16.c
2     Using sscanf */
3  #include <stdio.h>
4
5  int main()
6  {
7      char s[] = "31298 87.375"; /* initialize array s */
8      int x;                    /* define x */
9      double y;                 /* define y */
10
11     sscanf( s, "%d%lf", &x, &y );
12
13     printf( "%s\n%s%6d\n%s%8.3f\n",
14            "The values stored in character array s are:",
15            "integer:", x, "double:", y );
16
17     return 0; /* indicates successful termination */
18
19 } /* end main */
```

```
The values stored in character array s are:
integer: 31298
double:  87.375
```

**Program Output**

## 8.6 Manipulacija stringovima

- Funkcije biblioteke `<string.h>` imaju ulogu da
  - Manipulišu stringovima
  - Pretražuju stringove
  - Tokenizuju stringove
  - Odrede dužinu stringa

Prototip	Opis
<code>char *strcpy( char *s1, const char *s2 )</code>	Kopira string <code>s2</code> u niz <code>s1</code> . Vraća se vrijednost <code>s1</code> .
<code>char *strncpy( char *s1, const char *s2, size_t n )</code>	Kopira najviše <code>n</code> karaktera string <code>s2</code> u niz <code>s1</code> . Vraća se vrijednost <code>s1</code> .
<code>char *strcat( char *s1, const char *s2 )</code>	Nadovezuje string <code>s2</code> na niz <code>s1</code> . Prvi karakter <code>s2</code> eliminiše završni null karakter za <code>s1</code> . Vraća se vrijednost <code>s1</code> .
<code>char *strncat( char *s1, const char *s2, size_t n )</code>	Nadovezuje najviše <code>n</code> karaktera stringa <code>s2</code> na niz <code>s1</code> . Prvi karakter <code>s2</code> eliminiše završni null karakter za <code>s1</code> . Vraća se vrijednost <code>s1</code> .



## Outline

fig08\_18.c

```
1  /* Fig. 8.18: fig08_18.c
2     Using strcpy and strncpy */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8      char x[] = "Happy Birthday to You"; /* initialize char array x */
9      char y[ 25 ];                      /* create char array y */
10     char z[ 15 ];                      /* create char array z */
11
12     /* copy contents of x into y */
13     printf( "%s%s\n%s%s\n",
14             "The string in array x is: ", x,
15             "The string in array y is: ", strcpy( y, x ) );
16
17     /* copy first 14 characters of x into z. Does not copy null
18        character */
19     strncpy( z, x, 14 );
20
21     z[ 14 ] = '\0'; /* append '\0' to z's contents */
22     printf( "The string in array z is: %s\n", z );
23
24     return 0; /* indicates successful termination */
25
```

```
The string in array x is: Happy Birthday to You
The string in array y is: Happy Birthday to You
The string in array z is: Happy Birthday
```

Program Output



```
1  /* Fig. 8.19: fig08_19.c
2     Using strcat and strncat */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8      char s1[ 20 ] = "Happy "; /* initialize char array s1 */
9      char s2[] = "New Year "; /* initialize char array s2 */
10     char s3[ 40 ] = "";      /* initialize char array s3 */
11
12     printf( "s1 = %s\ns2 = %s\n", s1, s2 );
13
14     /* concatenate s2 to s1 */
15     printf( "strcat( s1, s2 ) = %s\n", strcat( s1, s2 ) );
16
17     /* concatenate first 6 characters of s1 to s3. Place '\0'
18        after last character */
19     printf( "strncat( s3, s1, 6 ) = %s\n", strncat( s3, s1, 6 ) );
20
21     /* concatenate s1 to s3 */
22     printf( "strcat( s3, s1 ) = %s\n", strcat( s3, s1 ) );
23
24     return 0; /* indicates successful termination */
25
26 } /* end main */
```



```
s1 = Happy  
s2 = New Year  
strcat( s1, s2 ) = Happy New Year  
strncat( s3, s1, 6 ) = Happy  
strcat( s3, s1 ) = Happy Happy New Year
```



Outline

**Program Output**

## 8.7 Funkcije za poređenje stringova

- Upoređivanje stringova
  - Kompjuter upoređuje numeričke ASCII kodove karaktera u stringu
  - Pronaći u literaturi listu ASCII kodova

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

- Upoređuje string s1 sa stringom s2
- Vraća negativan broj ako je  $s1 < s2$ , nulu ako je  $s1 == s2$  ili pozitivan broj ako je  $s1 > s2$

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

- Upoređuje najviše n karaktera stringa s1 sa stringom s2
- Vraća iste vrijednosti kao i `strcmp`



## Outline

fig08\_21.c

```
1  /* Fig. 8.21: fig08_21.c
2     Using strcmp and strncmp */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8     const char *s1 = "Happy New Year"; /* initialize char pointer */
9     const char *s2 = "Happy New Year"; /* initialize char pointer */
10    const char *s3 = "Happy Holidays"; /* initialize char pointer */
11
12    printf("%s%s\n%s%s\n%s%s\n\n%s%2d\n%s%2d\n%s%2d\n\n",
13          "s1 = ", s1, "s2 = ", s2, "s3 = ", s3,
14          "strcmp(s1, s2) = ", strcmp( s1, s2 ),
15          "strcmp(s1, s3) = ", strcmp( s1, s3 ),
16          "strcmp(s3, s1) = ", strcmp( s3, s1 ) );
17
18    printf("%s%2d\n%s%2d\n%s%2d\n",
19          "strncmp(s1, s3, 6) = ", strncmp( s1, s3, 6 ),
20          "strncmp(s1, s3, 7) = ", strncmp( s1, s3, 7 ),
21          "strncmp(s3, s1, 7) = ", strncmp( s3, s1, 7 ) );
22
23    return 0; /* indicates successful termination */
24
25 } /* end main */
```



## Outline



### Program Output

```
s1 = Happy New Year  
s2 = Happy New Year  
s3 = Happy Holidays
```

```
strcmp(s1, s2) = 0  
strcmp(s1, s3) = 1  
strcmp(s3, s1) = -1
```

```
strncmp(s1, s3, 6) = 0  
strncmp(s1, s3, 7) = 1  
strncmp(s3, s1, 7) = -1
```

## 8.8 Funkcije za pretraživanje stringova

Prototip	Opis
<code>char *strchr( const char *s, int c );</code>	Locira prvo pojavljivanje karaktera <code>c</code> u stringu <code>s</code> . Ako je <code>c</code> nađen, vraća se pokazivač na <code>c</code> . Inače, vraća se <code>NULL</code> .
<code>size_t strcspn( const char *s1, const char *s2 );</code>	Izračunava i vraća dužinu početnog dijela (segmenta) stringa <code>s1</code> koji se sastoji od karaktera koji ne pripadaju stringu <code>s2</code> .
<code>size_t strspn( const char *s1, const char *s2 );</code>	Izračunava i vraća dužinu početnog dijela (segmenta) stringa <code>s1</code> koji se sastoji od karaktera koji pripadaju stringu <code>s2</code> .
<code>char *strpbrk( const char *s1, const char *s2 );</code>	Locira prvo pojavljivanje u stringu <code>s1</code> bilo kog karaktera iz stringa <code>s2</code> . Ako je karakter iz <code>s2</code> nađen, vraća se pokazivač na karakter u stringu <code>s1</code> . Inače, vraća se <code>NULL</code> .
<code>char *strrchr( const char *s, int c );</code>	Locira posljednje pojavljivanje karaktera <code>c</code> u stringu <code>s</code> . Ako je <code>c</code> nađen, vraća se pokazivač na <code>c</code> u string <code>s</code> . Inače, vraća se <code>NULL</code> .
<code>char *strstr( const char *s1, const char *s2 );</code>	Locira prvo pojavljivanje stringa <code>s1</code> u stringu <code>s2</code> . Ako je string pronađen, vraća se pokazivač na string u <code>s1</code> . Inače, vraća se <code>NULL</code> .
<code>char *strtok( char *s1, const char *s2 );</code>	Niz poziva funkciji <code>strtok</code> razbija string <code>s1</code> u “tokene”—logčke djeliće kaošto u riječi u liniji teksta koje su međusobno razdvojene karakterima koji pripadaju stringu <code>s2</code> . Prvi poziv sadrži <code>s1</code> kao prvi argument, a sledeći pozivi nastavljaju sa tokenizacijom istog stringa, ali imaju <code>NULL</code> kao prvi argument. Vraća se pokazivač na tekući token u svakom pozivu. Ako više nema tokena, vraća se <code>NULL</code> .



## Outline



### fig08\_23.c (Part 1 of 2)

```
1  /* Fig. 8.23: fig08_23.c
2     Using strchr */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8     const char *string = "This is a test"; /* initialize char pointer */
9     char character1 = 'a';                 /* initialize character1 */
10    char character2 = 'z';                 /* initialize character2 */
11
12    /* if character1 was found in string */
13    if ( strchr( string, character1 ) != NULL ) {
14        printf( "'%c' was found in \"%s\".\n",
15               character1, string );
16    } /* end if */
17    else { /* if character1 was not found */
18        printf( "'%c' was not found in \"%s\".\n",
19               character1, string );
20    } /* end else */
21
```



## Outline



fig08\_23.c (Part 2 of 2)

```
22  /* if character2 was found in string */
23  if ( strchr( string, character2 ) != NULL ) {
24      printf( "\\'%c\\' was found in \"%s\".\n",
25              character2, string );
26  } /* end if */
27  else { /* if character2 was not found */
28      printf( "\\'%c\\' was not found in \"%s\".\n",
29              character2, string );
30  } /* end else */
31
32  return 0; /* indicates successful termination */
33
34 } /* end main */
```

```
'a' was found in "This is a test".
'z' was not found in "This is a test".
```

Program Output



## Outline



fig08\_24.c

```
1  /* Fig. 8.24: fig08_24.c
2      Using strcspn */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8      /* initialize two char pointers */
9      const char *string1 = "The value is 3.14159";
10     const char *string2 = "1234567890";
11
12     printf( "%s%s\n%s%s\n\n%s\n%s%u",
13             "string1 = ", string1, "string2 = ", string2,
14             "The length of the initial segment of string1",
15             "containing no characters from string2 = ",
16             strcspn( string1, string2 ) );
17
18     return 0; /* indicates successful termination */
19
20 } /* end main */
```

```
string1 = The value is 3.14159
string2 = 1234567890
```

```
The length of the initial segment of string1
containing no characters from string2 = 13
```

Program Output





## Outline



fig08\_25.c

```
1  /* Fig. 8.25: fig08_25.c
2     Using strpbrk */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8      const char *string1 = "This is a test"; /* initialize char pointer */
9      const char *string2 = "beware";         /* initialize char pointer */
10
11      printf( "%s\\\"%s\\\"\\n'%c'\\\"%s\\n\\\"%s\\\"\\n",
12              "Of the characters in ", string2,
13              *strpbrk( string1, string2 ),
14              " is the first character to appear in ", string1 );
15
16      return 0; /* indicates successful termination */
17
18 } /* end main */
```

```
Of the characters in "beware"
'a' is the first character to appear in
"This is a test"
```

Program Output



## Outline



fig08\_26.c

```
1  /* Fig. 8.26: fig08_26.c
2      Using strrchr */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8      /* initialize char pointer */
9      const char *string1 = "A zoo has many animals "
10                          "including zebras";
11      int c = 'z'; /* initialize c */
12
13      printf( "%s\n%s'%c'%s\"%s\"\\n",
14              "The remainder of string1 beginning with the",
15              "last occurrence of character ", c,
16              " is: ", strrchr( string1, c ) );
17
18      return 0; /* indicates successful termination */
19
20 } /* end main */
```

Program Output

The remainder of string1 beginning with the  
last occurrence of character 'z' is: "zebras"



```
1  /* Fig. 8.27: fig08_27.c
2     Using strspn */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8     /* initialize two char pointers */
9     const char *string1 = "The value is 3.14159";
10    const char *string2 = "ae hi lsTuv";
11
12    printf( "%s%s\n%s%s\n\n%s\n%s%u\n",
13            "string1 = ", string1, "string2 = ", string2,
14            "The length of the initial segment of string1",
15            "containing only characters from string2 = ",
16            strspn( string1, string2 ) );
17
18    return 0; /* indicates successful termination */
19
20 } /* end main */
```

```
string1 = The value is 3.14159
string2 = ae hi lsTuv
```

```
The length of the initial segment of string1
containing only characters from string2 = 13
```

## Program Output



## Outline



fig08\_28.c

```
1  /* Fig. 8.28: fig08_28.c
2      Using strstr */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8      const char *string1 = "abcdefabcdef"; /* initialize char pointer */
9      const char *string2 = "def";          /* initialize char pointer */
10
11     printf( "%s%s\n%s%s\n\n%s\n%s%s\n",
12             "string1 = ", string1, "string2 = ", string2,
13             "The remainder of string1 beginning with the",
14             "first occurrence of string2 is: ",
15             strstr( string1, string2 ) );
16
17     return 0; /* indicates successful termination */
18
19 } /* end main */
```

```
string1 = abcdefabcdef
string2 = def
```

```
The remainder of string1 beginning with the
first occurrence of string2 is: defabcdef
```

**Program Output**



```
1  /* Fig. 8.29: fig08_29.c
2      Using strtok */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8      /* initialize array string */
9      char string[] = "This is a sentence with 7 tokens";
10     char *tokenPtr; /* create char pointer */
11
12     printf( "%s\n%s\n\n%s\n",
13             "The string to be tokenized is:", string,
14             "The tokens are:" );
15
16     tokenPtr = strtok( string, " " ); /* begin tokenizing sentence */
17
18     /* continue tokenizing sentence until tokenPtr becomes NULL */
19     while ( tokenPtr != NULL ) {
20         printf( "%s\n", tokenPtr );
21         tokenPtr = strtok( NULL, " " ); /* get next token */
22     } /* end while */
23
24     return 0; /* indicates successful termination */
25
26 } /* end main */
```

[Outline](#)**Program Output**

The string to be tokenized is:  
This is a sentence with 7 tokens

The tokens are:

This  
is  
a  
sentence  
with  
7  
tokens

## 8.9 Memorijske funkcije

- Memorijske funkcije
  - U biblioteci `<stdlib.h>`
  - Manipulišu, upoređuju i pretražuju blokove memorije
  - Mogu manipulirati sa bilo kakvim blokom podatka
- Tip argumenta je `void *`
  - Svaki pokazivač može biti dodijeljen promjenljivoj tipa `void *` i obrnuto
  - `void *` ne može biti dereferenciran
    - Svaka funkcija ima i argument koji određuje broj bajtova (karaktera) koje treba obraditi

## 8.9 Memorijske funkcije

Prototip	Opis
<code>void *memcpy( void *s1, const void *s2, size_t n );</code>	Kopira <i>n</i> karaktera iz objekta na koji pokazuje <i>s2</i> u objekat na koji pokazuje <i>s1</i> . Vraća pokazivač na rezultujući objekat.
<code>void *memmove( void *s1, const void *s2, size_t n );</code>	Kopira <i>n</i> karaktera iz objekta na koji pokazuje <i>s2</i> u objekat na koji pokazuje <i>s1</i> . Kopiranje se izvodi tako što se karakteri kopiraju u pomoćni niz a onda iz pomoćnog niza u objekat na koji pokazuje <i>s1</i> . Vraća pokazivač na rezultujući objekat.
<code>int memcmp( const void *s1, const void *s2, size_t n );</code>	Upoređuje prvih <i>n</i> karaktera objekta na koji pokazuje <i>s1</i> sa objektom na koji pokazuje <i>s2</i> . Vraća 0, manje od nule ili veće od nule ako je <i>s1</i> jednak, manji ili veći od <i>s2</i> .
<code>void *memchr( const void *s, int c, size_t n );</code>	Locira prvo pojavljivanje <i>c</i> (konvertovanog u <code>unsigned char</code> ) u prvih <i>n</i> karaktera objekta na koji pokazuje <i>s</i> . Ako je <i>c</i> pronađeno, vraća se pokazivač na <i>c</i> . Inače, vraća se <code>NULL</code> .
<code>void *memset( void *s, int c, size_t n );</code>	Kopira <i>c</i> (konvertovano u <code>unsigned char</code> ) u prvih <i>n</i> karaktera objekta na koji pokazuje <i>s</i> . Vraća pokazivač na rezultat.





## Outline



fig08\_31.c

```
1  /* Fig. 8.31: fig08_31.c
2      Using memcpy */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8      char s1[ 17 ];           /* create char array s1 */
9      char s2[] = "Copy this string"; /* initialize char array s2 */
10
11     memcpy( s1, s2, 17 );
12     printf( "%s\n%s\n%s\n",
13            "After s2 is copied into s1 with memcpy,",
14            "s1 contains ", s1 );
15
16     return 0; /* indicates successful termination */
17
18 } /* end main */
```

Program Output

```
After s2 is copied into s1 with memcpy,
s1 contains "Copy this string"
```



## Outline



fig08\_32.c

```
1  /* Fig. 8.32: fig08_32.c
2      Using memmove */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8      char x[] = "Home Sweet Home"; /* initialize char array x */
9
10     printf( "%s%s\n", "The string in array x before memmove is: ", x );
11     printf( "%s%s\n", "The string in array x after memmove is: ",
12         memmove( x, &x[ 5 ], 10 ) );
13
14     return 0; /* indicates successful termination */
15
16 } /* end main */
```

```
The string in array x before memmove is: Home Sweet Home
The string in array x after memmove is: Sweet Home Home
```

**Program Output**



```
1  /* Fig. 8.33: fig08_33.c
2     Using memcmp */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8      char s1[] = "ABCDEFGH"; /* initialize char array s1 */
9      char s2[] = "ABCDXYZ"; /* initialize char array s2 */
10
11     printf( "%s%s\n%s%s\n\n%s%2d\n%s%2d\n%s%2d\n",
12             "s1 = ", s1, "s2 = ", s2,
13             "memcmp( s1, s2, 4 ) = ", memcmp( s1, s2, 4 ),
14             "memcmp( s1, s2, 7 ) = ", memcmp( s1, s2, 7 ),
15             "memcmp( s2, s1, 7 ) = ", memcmp( s2, s1, 7 ) );
16
17     return 0; /* indicate successful termination */
18
19 } /* end main */
```

```
s1 = ABCDEFGH
s2 = ABCDXYZ
```

```
memcmp( s1, s2, 4 ) = 0
memcmp( s1, s2, 7 ) = -1
memcmp( s2, s1, 7 ) = 1
```

Program Output

**Fig8\_34.c**

```
1  /* Fig. 8.34: fig08_34.c
2      Using memchr */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8      const char *s = "This is a string"; /* initialize char pointer */
9
10     printf( "%s\ '%c\ '%s\ \"%s\ \"\n",
11             "The remainder of s after character ", 'r',
12             " is found is ", memchr( s, 'r', 16 ) );
13
14     return 0; /* indicates successful termination */
15
16 } /* end main */
```

The remainder of s after character 'r' is found is "ring"

**Program Output**

## 8.10 Ostale funkcije u biblioteci

- `char *strerror( int errornum );`
  - Kreira sistemski-zavisne greške bazirane na `errornum`
  - Vraća pokazivač na string
- `size_t strlen( const char *s );`
  - Vraća broj karaktera (bez NULL) u stringu `s`



## Outline



fig08\_35.c

```
1  /* Fig. 8.35: fig08_35.c
2     Using memset */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8     char string1[ 15 ] = "BBBBBBBBBBBBBB"; /* initialize string1 */
9
10    printf( "string1 = %s\n", string1 );
11    printf( "string1 after memset = %s\n", memset( string1, 'b', 7 ) );
12
13    return 0; /* indicates successful termination */
14
15 } /* end main */
```

```
string1 = BBBBBBBBBBBBBB
string1 after memset = bbbbbbbBBBBBBB
```

**Program Output**



## Outline



fig08\_37.c

```
1  /* Fig. 8.37: fig08_37.c
2      Using strerror */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8      printf( "%s\n", strerror( 2 ) );
9
10     return 0; /* indicates successful termination */
11
12 } /* end main */
```

Program Output

No such file or directory



```
1  /* Fig. 8.38: fig08_38.c
2      Using strlen */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8      /* initialize 3 char pointers */
9      const char *string1 = "abcdefghijklmnopqrstuvwxyz";
10     const char *string2 = "four";
11     const char *string3 = "Boston";
12
13     printf("%s\\\"%s\\\"%s%lu\\n%s\\\"%s\\\"%s%lu\\n%s\\\"%s\\\"%s%lu\\n",
14         "The length of ", string1, " is ",
15         ( unsigned long ) strlen( string1 ),
16         "The length of ", string2, " is ",
17         ( unsigned long ) strlen( string2 ),
18         "The length of ", string3, " is ",
19         ( unsigned long ) strlen( string3 ) );
20
21     return 0; /* indicates successful termination */
22
23 } /* end main */
```

```
The length of "abcdefghijklmnopqrstuvwxyz" is 26
The length of "four" is 4
The length of "Boston" is 6
```

## Program Output