Lekcija 8 – Karakteri i stringovi

Pregled

Ciljevi lekcije

- U ovoj lekciji:
  - Naučićete kako da koristite biblioteku `ctype.`
  - 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
  - Olakašavaju 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 cjelobrojnju 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);
```

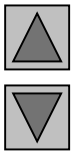
    - Kopira se ulaz u `word[]`
    - Nije potrebno `&` (jer je string već pokazivač)
  - Ne zaboravite prostor u nizu za `'\0'`

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

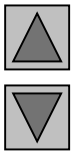
Prototip	Opis
`int isdigit( int c );`	Vraća `true` ako je `c` cifra i `false` inače.
`int isalpha( int c );`	Vraća `true` ako je `c` slovi i `false` inače.
`int isalnum( int c );`	Vraća `true` ako je `c` cifra ili slovo i `false` inače.
`int isxdigit( int c );`	Vraća `true` ako je `c` heksadecimalni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



Outline

fig08_02.c (Part 1 of 2)

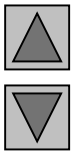
```
1  /* Fig. 8.2: fig08_02.c
2     Using functions isdigit, isalpha, isalnum, and isxdigit */
3  #include <stdio.h>
4  #include <ctype.h>
5
6  int main()
7  {
8     printf( "%s\n%s%s\n%s%s\n\n", "According to isdigit: ",
9         isdigit( '8' ) ? "8 is a " : "8 is not a ", "digit",
10        isdigit( '#' ) ? "# is a " : "# is not a ", "digit" );
11
12    printf( "%s\n%s%s\n%s%s\n%s%s\n%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

```
28    printf( "%s\n%s%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 */
```

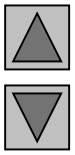
fig08_02.c (Part 2 of 2)



Outline

Program Output

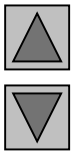
```
According to isdigit:
8 is a digit
# is not a digit

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%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%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
```



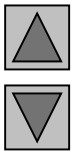
Outline

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 */
```

Program Output

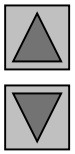
```
According to islower:
p is a lowercase letter
P is not a lowercase letter
5 is not a lowercase letter
! is not a lowercase letter

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
```



Outline

fig08_04.c (Part 1 of 2)

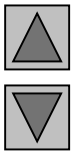
```
1  /* Fig. 8.4: fig08_04.c
2     Using functions isspace, iscntrl, ispunct, isprint, isgraph */
3  #include <stdio.h>
4  #include <ctype.h>
5
6  int main()
7  {
8     printf( "%s\n%s%s%s\n%s%s%s\n%s%s\n\n",
9        "According to isspace:",
10       "Newline", isspace( '\n' ) ? " is a " : " is not 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
```



Outline

fig08_04.c (Part 2 of 2)

```
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 */
```



Outline

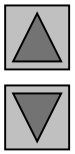
Program Output

```
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

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

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



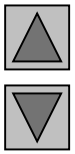
Outline

fig 08_06.c

```c
/* Fig. 8.6: fig08_06.c
   Using atof */
#include <stdio.h>
#include <stdlib.h>

int main()
{
   double d; /* variable to hold converted string */

   d = atof( "99.0" );

   printf( "%s%.3f\n%s%.3f\n",
           "The string \"99.0\" converted to double is ", d,
           "The converted value divided by 2 is ",
           d / 2.0 );

   return 0; /* indicates successful termination */

} /* end main */
```

Program Output

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



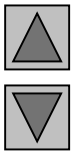
Outline

fig08_07.c

```c
/* Fig. 8.7: fig08_07.c
   Using atoi */
#include <stdio.h>
#include <stdlib.h>

int main()
{
   int i; /* variable to hold converted string */

   i = atoi( "2593" );

   printf( "%s%d\n%s%d\n",
           "The string \"2593\" converted to int is ", i,
           "The converted value minus 593 is ", i - 593 );

   return 0; /* indicates successful termination */

} /* end main */
```

Program Output

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



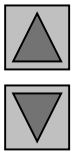
Outline

fig08_08.c

```c
/* Fig. 8.8: fig08_08.c
   Using atol */
#include <stdio.h>
#include <stdlib.h>

int main()
{
   long l; /* variable to hold converted string */

   l = atol( "1000000" );

   printf( "%s%ld\n%s%ld\n",
           "The string \"1000000\" converted to long int is ", l,
           "The converted value divided by 2 is ", l / 2 );

   return 0; /* indicates successful termination */

} /* end main */
```

Program Output

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



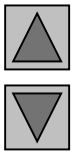
Outline

fig08_09.c

```
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 */
```

Program Output

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



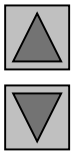
Outline

fig08_10.c

```c
/* Fig. 8.10: fig08_10.c
   Using strtol */
#include <stdio.h>
#include <stdlib.h>

int main()
{
   const char *string = "-1234567abc"; /* initialize string pointer */

   char *remainderPtr; /* create char pointer */
   long x;             /* variable to hold converted sequence */

   x = strtol( string, &remainderPtr, 0 );

   printf( "%s\"%s\"\n%s%ld\n%s\"%s\"\n%s%ld\n",
           "The original string is ", string,
           "The converted value is ", x,
           "The remainder of the original string is ",
           remainderPtr,
           "The converted value plus 567 is ", x + 567 );

   return 0; /* indicates successful termination */

} /* 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 plus 567 is -1234000
```



Outline

fig08_11.c

```c
/* Fig. 8.11: fig08_11.c
   Using strtoul */
#include <stdio.h>
#include <stdlib.h>

int main()
{
   const char *string = "1234567abc"; /* initialize string pointer */
   unsigned long x;    /* variable to hold converted sequence */
    char *remainderPtr; /* create char pointer */

   x = strtoul( string, &remainderPtr, 0 );

   printf( "%s\"%s\"\n%s%lu\n%s\"%s\"\n%s%lu\n",
           "The original string is ", string,
           "The converted value is ", x,
           "The remainder of the original string is ",
           remainderPtr,
           "The converted value minus 567 is ", x - 567 );

   return 0; /* indicates successful termination */

} /* end main */
```

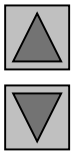
Program Output

```
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

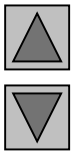
Prototip	Opis
`int getchar( void );`	Učitava sledeći karakter sa standardnog ulaza i vraća cio broj.
`char *gets( char *s );`	Učitava karaktere sa standardnog ulaza u niz `s` do pojave newline ili end-of-file karaktera. Završni null karakter se nadovezuje na niz.
`int putchar( int c );`	Štampa karakter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



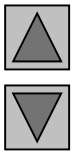
Outline

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 */
```

Program Output

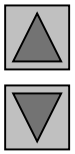
```
Enter a line of text:
Characters and Strings

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
```



Outline

fig8_14.c

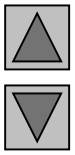
```c
/* Fig. 8.14: fig08_14.c
   Using getchar and puts */
#include <stdio.h>

int main()
{
   char c;              /* variable to hold character input by user */
   char sentence[ 80 ]; /* create char array */
   int i = 0;           /* initialize counter i */

   /* prompt user to enter line of text */
   puts( "Enter a line of text:" );

   /* use getchar to read each character */
   while ( ( c = getchar() ) != '\n') {
      sentence[ i++ ] = c;
   } /* end while */

   sentence[ i ] = '\0';

   /* use puts to display sentence */
   puts( "\nThe line entered was:" );
   puts( sentence );

   return 0; /* indicates successful termination */

} /* end main */
```



Outline

Program Output

```
Enter a line of text:
This is a test.

The line entered was:
This is a test.
```



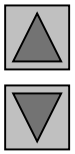
Outline

fig08_15.c

```c
/* Fig. 8.15: fig08_15.c
   Using sprintf */
#include <stdio.h>

int main()
{
   char s[ 80 ]; /* create char array */
   int x;        /* define x */
   double y;     /* define y */

   printf( "Enter an integer and a double:\n" );
   scanf( "%d%lf", &x, &y );

   sprintf( s, "integer:%6d\ndouble:%8.2f", x, y );

   printf( "%s\n%s\n",
           "The formatted output stored in array s is:", s );

   return 0; /* indicates successful termination */

} /* end main */
```

Program Output

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



Outline

fig08_16.c

```c
/* Fig. 8.16: fig08_16.c
   Using sscanf */
#include <stdio.h>

int main()
{
   char s[] = "31298 87.375"; /* initialize array s */
   int x;                     /* define x */
   double y;                  /* define y */

   sscanf( s, "%d%lf", &x, &y );

   printf( "%s\n%s%6d\n%s%8.3f\n",
           "The values stored in character array s are:",
           "integer:", x, "double:", y );

   return 0; /* indicates successful termination */

} /* end main */
```

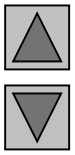
Program Output

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

8.6 Manipulacija stringovima

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

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



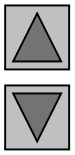
Outline

fig08_18.c

```c
/* Fig. 8.18: fig08_18.c
   Using strcpy and strncpy */
#include <stdio.h>
#include <string.h>

int main()
{
   char x[] = "Happy Birthday to You"; /* initialize char array x */
   char y[ 25 ];                       /* create char array y */
    char z[ 15 ];                       /* create char array z */

   /* copy contents of x into y */
   printf( "%s%s\n%s%s\n",
           "The string in array x is: ", x,
           "The string in array y is: ", strcpy( y, x ) );

   /* copy first 14 characters of x into z. Does not copy null
      character */
   strncpy( z, x, 14 );

   z[ 14 ] = '\0'; /* append '\0' to z's contents */
   printf( "The string in array z is: %s\n", z );

   return 0; /* indicates successful termination */

```

Program Output

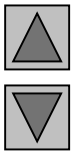
```
The string in array x is: Happy Birthday to You
The string in array y is: Happy Birthday to You
The string in array z is: Happy Birthday
```



Outline

fig08_19.c

```c
/* Fig. 8.19: fig08_19.c
   Using strcat and strncat */
#include <stdio.h>
#include <string.h>

int main()
{
   char s1[ 20 ] = "Happy "; /* initialize char array s1 */
   char s2[] = "New Year ";  /* initialize char array s2 */
    char s3[ 40 ] = "";       /* initialize char array s3 */

   printf( "s1 = %s\ns2 = %s\n", s1, s2 );

   /* concatenate s2 to s1 */
   printf( "strcat( s1, s2 ) = %s\n", strcat( s1, s2 ) );

   /* concatenate first 6 characters of s1 to s3. Place '\0'
      after last character */
   printf( "strncat( s3, s1, 6 ) = %s\n", strncat( s3, s1, 6 ) );

   /* concatenate s1 to s3 */
   printf( "strcat( s3, s1 ) = %s\n", strcat( s3, s1 ) );

   return 0; /* indicates successful termination */

} /* end main */
```



Outline

Program Output

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

8.7 Funkcije za poređenje stringova

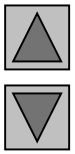
- Upoređivanje stringova
  - Kompjuter upoređuje numeričke ASCII kodove karaktera u stringu
  - Pronaći u literaturi listu ASCII kodova

```
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 );
```

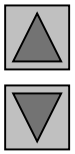
  - Upoređuje najviše `n` karaktera stringa `s1` sa stringom `s2`
  - Vraća iste vrijednosti kao i `strcmp`



Outline

fig08_21.c

```c
/* Fig. 8.21: fig08_21.c
   Using strcmp and strncmp */
#include <stdio.h>
#include <string.h>

int main()
{
   const char *s1 = "Happy New Year"; /* initialize char pointer */
   const char *s2 = "Happy New Year"; /* initialize char pointer */
   const char *s3 = "Happy Holidays"; /* initialize char pointer */

   printf("%s%s\n%s%s\n%s%s\n\n%s%2d\n%s%2d\n%s%2d\n\n",
          "s1 = ", s1, "s2 = ", s2, "s3 = ", s3,
          "strcmp(s1, s2) = ", strcmp( s1, s2 ),
          "strcmp(s1, s3) = ", strcmp( s1, s3 ),
          "strcmp(s3, s1) = ", strcmp( s3, s1 ) );

   printf("%s%2d\n%s%2d\n%s%2d\n",
          "strncmp(s1, s3, 6) = ", strncmp( s1, s3, 6 ),
          "strncmp(s1, s3, 7) = ", strncmp( s1, s3, 7 ),
          "strncmp(s3, s1, 7) = ", strncmp( s3, s1, 7 ) );

   return 0; /* indicates successful termination */

} /* end main */
```



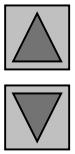
Outline

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

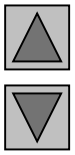
Prototip	Opis
`char *strchr( const char *s, int c );`	Locira prvo pojavljivanje karaktera `c` u stringu `s`. Ako je `c` nađen, vraća se pokazivač na `c`. Inače, vraća se `NULL`.
`size_t strcspn( const char *s1, const char *s2 );`	Izračunava i vraća dužinu početnog dijela (segmenta) stringa `s1` koji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



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
```



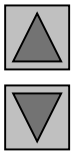
Outline

```
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 */
```

fig08_23.c (Part 2 of 2)

Program Output

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



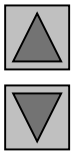
Outline

fig08_24.c

```c
/* Fig. 8.24: fig08_24.c
   Using strcspn */
#include <stdio.h>
#include <string.h>

int main()
{
   /* initialize two char pointers */
   const char *string1 = "The value is 3.14159";
    const char *string2 = "1234567890";

   printf( "%s%s\n%s%s\n\n%s\n%s%u",
           "string1 = ", string1, "string2 = ", string2,
           "The length of the initial segment of string1",
           "containing no characters from string2 = ",
           strcspn( string1, string2 ) );

   return 0; /* indicates successful termination */

} /* end main */
```

Program Output

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

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



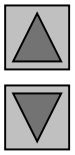
Outline

fig08_25.c

```c
/* Fig. 8.25: fig08_25.c
   Using strpbrk */
#include <stdio.h>
#include <string.h>

int main()
{
   const char *string1 = "This is a test"; /* initialize char pointer */
   const char *string2 = "beware";         /* initialize char pointer */

   printf( "%s\"%s\"\n'%c'%s\n\"%s\"\n",
           "Of the characters in ", string2,
           *strpbrk( string1, string2 ),
           " is the first character to appear in ", string1 );

   return 0; /* indicates successful termination */

} /* end main */
```

Program Output

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



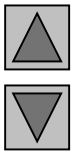
Outline

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"
```



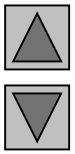
Outline

fig08_27.c

```c
/* Fig. 8.27: fig08_27.c
   Using strspn */
#include <stdio.h>
#include <string.h>

int main()
{
   /* initialize two char pointers */
   const char *string1 = "The value is 3.14159";
    const char *string2 = "aehi lsTuv";

    printf( "%s%s\n%s%s\n\n%s\n%s%u\n",
            "string1 = ", string1, "string2 = ", string2,
            "The length of the initial segment of string1",
            "containing only characters from string2 = ",
            strspn( string1, string2 ) );

    return 0; /* indicates successful termination */

} /* end main */
```

Program Output

```
string1 = The value is 3.14159
string2 = aehi lsTuv

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



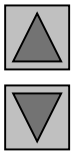
Outline

fig08_28.c

```c
/* Fig. 8.28: fig08_28.c
   Using strstr */
#include <stdio.h>
#include <string.h>

int main()
{
   const char *string1 = "abcdefabcdef"; /* initialize char pointer */
   const char *string2 = "def";          /* initialize char pointer */

   printf( "%s%s\n%s%s\n\n%s\n%s%s\n",
           "string1 = ", string1, "string2 = ", string2,
           "The remainder of string1 beginning with the",
           "first occurrence of string2 is: ",
           strstr( string1, string2 ) );

   return 0; /* indicates successful termination */

} /* end main */
```

Program Output

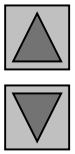
```
string1 = abcdefabcdef
string2 = def

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



Outline

fig08_29.c

```c
/* Fig. 8.29: fig08_29.c
   Using strtok */
#include <stdio.h>
#include <string.h>

int main()
{
   /* initialize array string */
   char string[] = "This is a sentence with 7 tokens";
    char *tokenPtr; /* create char pointer */

   printf( "%s\n%s\n\n%s\n",
           "The string to be tokenized is:", string,
           "The tokens are:" );

   tokenPtr = strtok( string, " " ); /* begin tokenizing sentence */

   /* continue tokenizing sentence until tokenPtr becomes NULL */
   while ( tokenPtr != NULL ) {
      printf( "%s\n", tokenPtr );
      tokenPtr = strtok( NULL, " " ); /* get next token */
   } /* end while */

   return 0; /* indicates successful termination */

} /* end main */
```



Outline

Program Output

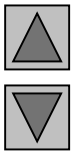
```
The string to be tokenized is:
This is a sentence with 7 tokens

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 manipulisati sa bilo kakvim blokom podatka
- Tip argumenta je `void *`
  - Svaki pokazivač može biti dodijeljen promjenljivoj tipa `void *` i obrnuto
  - `void *` ne može bit dereferenciran
    - Svaka funkcija ima i argument koji određuje broj bajtova (karaktera) koje treba obraditi

8.9 Memorijske funkcije

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



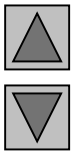
Outline

fig08_31.c

```c
/* Fig. 8.31: fig08_31.c
   Using memcpy */
#include <stdio.h>
#include <string.h>

int main()
{
   char s1[ 17 ];                    /* create char array s1 */
   char s2[]  = "Copy this string"; /* initialize char array s2 */

   memcpy( s1, s2, 17 );
   printf( "%s\n%s\"%s\"\n",
           "After s2 is copied into s1 with memcpy,",
           "s1 contains ", s1 );

   return 0; /* indicates successful termination */

} /* end main */
```

Program Output

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



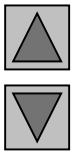
Outline

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 */
```

Program Output

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



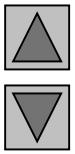
Outline

fig08_33.c

```
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[] = "ABCDEFG"; /* 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 */
```

Program Output

```
s1 = ABCDEFG
s2 = ABCDXYZ

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



Outline

Fig8_34.c

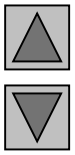
```
1  /* Fig. 8.34: fig08_34.c
2     Using memchr */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8     const char *s = "This is a string"; /* initialize char pointer */
9
10    printf( "%s\'%c\'%s\"%s\"\n",
11            "The remainder of s after character ", 'r',
12          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

Program Output

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

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`



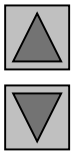
Outline

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 */
```

Program Output

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



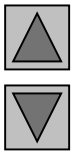
Outline

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
```



Outline

fig08_38.c

```c
/* Fig. 8.38: fig08_38.c
   Using strlen */
#include <stdio.h>
#include <string.h>

int main()
{
   /* initialize 3 char pointers */
   const char *string1 = "abcdefghijklmnopqrstuvwxyz";
    const char *string2 = "four";
    const char *string3 = "Boston";

   printf("%s\"%s\"%s%lu\n%s\"%s\"%s%lu\n%s\"%s\"%s%lu\n",
      "The length of ", string1, " is ",
      ( unsigned long ) strlen( string1 ),
      "The length of ", string2, " is ",
      ( unsigned long ) strlen( string2 ),
      "The length of ", string3, " is ",
      ( unsigned long ) strlen( string3 ) );

   return 0; /* indicates successful termination */

} /* end main */
```

Program Output

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