Chapter 8 - Characters and Strings

Outline

8.1	Introduction
8.2	Fundamentals of Strings and Characters
8.3	Character Handling Library
8.4	String Conversion Functions
8.5	Standard Input/Output Library Functions
8.6	String Manipulation Functions of the String Handling Library
8.7	Comparison Functions of the String Handling Library
8.8	Search Functions of the String Handling Library
8.9	Memory Functions of the String Handling Library
8.10	Other Functions of the String Handling Library

[©] Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.



Objectives

- In this chapter, you will learn:
 - To be able to use the functions of the character handling library (ctype).
 - To be able to use the string and character input/output functions of the standard input/output library (stdio).
 - To be able to use the string conversion functions of the general utilities library (stdlib).
 - To be able to use the string processing functions of the string handling library (string).
 - To appreciate the power of function libraries as a means of achieving software reusability.

[©] Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

8.1 Introduction

- Introduce some standard library functions
 - Easy string and character processing
 - Programs can process characters, strings, lines of text, and blocks of memory
- These techniques used to make
 - Word processors
 - Page layout software
 - Typesetting programs

[©] Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

8.2 Fundamentals of Strings and Characters

Characters

- Building blocks of programs
 - Every program is a sequence of meaningfully grouped characters
- Character constant
 - An int value represented as a character in single quotes
 - 'z' represents the integer value of z

• Strings

- Series of characters treated as a single unit
 - Can include letters, digits and special characters (*, /, \$)
- String literal (string constant) written in double quotes
 - "Hello"
- Strings are arrays of characters
 - String a pointer to first character
- Value of string is the address of first character © Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

8.2 Fundamentals of Strings and Characters

- String definitions
 - Define as a character array or a variable of type char *
 char color[] = "blue";
 char *colorPtr = "blue";
 - Remember that strings represented as character arrays end with '\0'
 - color has 5 elements
- Inputting strings
 - - Copies input into word[]
 - Do not need & (because a string is a pointer)
 - Remember to leave room in the array for '\0'

[©] Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.



8.3 Character Handling Library

- Character handling library
 - Includes functions to perform useful tests and manipulations of character data
 - Each function receives a character (an int) or EOF as an argument
- The following slide contains a table of all the functions in <ctype.h>

[©] Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.



8.3 Character Handling Library

Prototype	Description
<pre>int isdigit(int c);</pre>	Returns true if c is a digit and false otherwise.
<pre>int isalpha(int c);</pre>	Returns true if c is a letter and false otherwise.
<pre>int isalnum(int c);</pre>	Returns true if c is a digit or a letter and false otherwise.
<pre>int isxdigit(int c);</pre>	Returns true if c is a hexadecimal digit character and false otherwise.
<pre>int islower(int c);</pre>	Returns true if c is a lowercase letter and false otherwise.
<pre>int isupper(int c);</pre>	Returns true if c is an uppercase letter; false otherwise.
<pre>int tolower(int c);</pre>	If c is an uppercase letter, tolower returns c as a lowercase letter. Otherwise, tolower returns the argument unchanged.
<pre>int toupper(int c);</pre>	If c is a lowercase letter, toupper returns c as an uppercase letter. Otherwise, toupper returns the argument unchanged.
<pre>int isspace(int c);</pre>	Returns true if c is a white-space character—newline ('\n'), space (' '), form feed ('\f'), carriage return ('\r'), horizontal tab ('\t'), or vertical tab ('\v')—and false otherwise
<pre>int iscntrl(int c);</pre>	Returns true if c is a control character and false otherwise.
<pre>int ispunct(int c);</pre>	Returns true if c is a printing character other than a space, a digit, or a letter and false otherwise.
<pre>int isprint(int c);</pre>	Returns true value if c is a printing character including space (' ') and false otherwise.
<pre>int isgraph(int c);</pre>	Returns true if c is a printing character other than space (' ') and false otherwise.

[©] Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

```
Outline

fig08_02.c (Part 1 of 2)
```

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

```
fig08_02.c (Part 2 of
2)
```

9

Outline

printf("%s\n%s%s\n%s%s\n%s%s\n%s%s\n",

isxdigit('F') ? "F is a " : "F is not a ",

isxdigit('J') ? "J is a " : "J is not a ",

isxdigit('7') ? "7 is a " : "7 is not a ",

isxdigit('\$') ? "\$ is a " : "\$ is not a ",

isxdigit('f') ? "f is a " : "f is not a ",

return 0; /* indicates successful termination */

"According to isxdigit:",

"hexadecimal digit",

"hexadecimal digit".

"hexadecimal digit",

"hexadecimal digit",

"hexadecimal digit");

28

29

30

31

32

33

34

35

36

37

38

39 40

41 42

43 } /* end main */

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



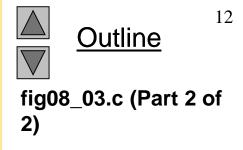
```
1 /* Fig. 8.3: fig08_03.c
     Using functions islower, isupper, tolower, toupper */
3 #include <stdio.h>
4 #include <ctype.h>
5
6 int main()
7 {
      printf( "%s\n%s%s\n%s%s\n%s%s\n\n",
8
             "According to islower:".
9
              islower('p') ? "p is a " : "p is not a ",
10
       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



fig08_03.c (Part 1 of 2)

11

```
printf( "%s%c\n%s%c\n%s%c\n",
30
             "u converted to uppercase is ", toupper('u'),
31
             "7 converted to uppercase is ", toupper('7'),
32
             "$ converted to uppercase is ", toupper('$'),
33
             "L converted to lowercase is ", tolower('L'));
34
35
     return 0; /* indicates successful termination */
36
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 1

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

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

```
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 isprint:
\$ is a printing character

; is a punctuation character

is a punctuation character

According to ispunct:

Alert is not a printing character

Y is not a punctuation 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 String Conversion Functions

- Conversion functions
 - In <stdlib.h> (general utilities library)
- Convert strings of digits to integer and floatingpoint values

Function prototype	Function description
<pre>double atof(const char *nPtr);</pre>	Converts the string nPtr to double.
<pre>int atoi(const char *nPtr);</pre>	Converts the string nPtr to int.
<pre>long atol(const char *nPtr);</pre>	Converts the string nPtr to long int.
<pre>double strtod(const char *nPtr, char **endPtr);</pre>	Converts the string nPtr to double.
<pre>long strtol(const char *nPtr, char **endPtr, int base);</pre>	Converts the string nPtr to long.
<pre>unsigned long strtoul(const char *nPtr, char **endPtr, int base);</pre>	Converts the string nPtr to unsigned long.

[©] Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.



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

1 /* Fig. 8.6: fig08_06.c

Program Output

The string "99.0" converted to double is 99.000

The converted value divided by 2 is 49.500

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



The converted value minus 593 is 2000

fig08_08.c

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

Program Output

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

The string "1000000" converted to long int is 1000000

The converted value divided by 2 is 500000

```
1 /* Fig. 8.9: fig08_09.c
     Using strtod */
3 #include <stdio.h>
4 #include <stdlib.h>
5
6 int main()
7 {
     /* initialize string pointer */
     const char *string = "51.2% are admitted";
10
      double d:
                      /* variable to hold converted sequence */
11
      char *stringPtr; /* create char pointer */
12
13
      d = strtod( string, &stringPtr );
14
15
      printf( "The string \"%s\" is converted to the\n", string );
16
      printf( "double value %.2f and the string \"%s\"\n", d, stringPtr );
17
18
      return 0: /* indicates successful termination */
19
20
21 } /* end main */
The string "51.2% are admitted" is converted to the
double value 51.20 and the string "% are admitted"
```

```
Outline
fig08_09.c
```

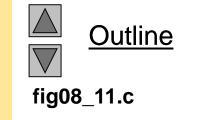
```
1 /* Fig. 8.10: fig08_10.c
      Using strtol */
 #include <stdio.h>
 #include <stdlib.h>
5
6 int main()
7 {
      const char *string = "-1234567abc"; /* initialize string pointer */
8
9
      char *remainderPtr; /* create char pointer */
10
      long x;
                         /* variable to hold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

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

- Functions in <stdio.h>
- Used to manipulate character and string data

Function prototype	Function description
<pre>int getchar(void);</pre>	Inputs the next character from the standard input and returns it as an integer.
<pre>char *gets(char *s);</pre>	Inputs characters from the standard input into the array s until a newline or end-of-file character is encountered. A terminating null character is appended to the array.
<pre>int putchar(int c);</pre>	Prints the character stored in c.
<pre>int puts(const char *s);</pre>	Prints the string s followed by a newline character.
<pre>int sprintf(char *s, const char *format,);</pre>	Equivalent to printf, except the output is stored in the array s instead of printing it on the screen.
<pre>int sscanf(char *s, const char *format,);</pre>	Equivalent to scanf, except the input is read from the array s instead of reading it from the keyboard.



Outline

fig08_13.c (Part 1 of 2)

```
Using gets and putchar */
3 #include <stdio.h>
5 int main()
     char sentence[ 80 ]; /* create char array */
     void reverse( const char * const sPtr ); /* prototype */
      printf( "Enter a line of text:\n" );
      /* use gets to read line of text */
      gets( sentence );
      printf( "\nThe line printed backwards is:\n" );
      reverse( sentence );
      return 0; /* indicates successful termination */
21 } /* end main */
```

1 /* Fig. 8.13: fig08_13.c

4

6

7 8

9 10

11 12

13

14 15

16

17 18

19 20

22

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

36 } /* end function reverse */

The line printed backwards is:

The line printed backwards is:

Enter a line of text:

Characters and Strings

sqnirtS dna sretcarahC

able was I ere I saw elba

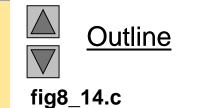
able was I ere I saw elba

Enter a line of text:

```
Outline
fig08_13.c (Part 1 of 2)
```

Program Output

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



© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

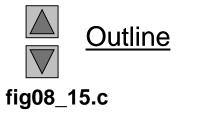
27 } /* end main */

Enter a line of text: This is a test.

The line entered was: This is a test.



```
1 /* Fig. 8.15: fig08_15.c
      Using sprintf */
3 #include <stdio.h>
4
 int main()
6
     char s[ 80 ]; /* create char array */
7
      int x;
                /* define x */
8
      double y; /* define y */
10
      printf( "Enter an integer and a double:\n" );
11
      scanf( "%d%1f", &x, &y );
12
13
      sprintf( s, "integer:%6d\ndouble:%8.2f", x, y );
14
15
      printf( "%s\n%s\n",
16
              "The formatted output stored in array s is:", s );
17
18
      return 0; /* indicates successful termination */
19
20
21 } /* end main */
Enter an integer and a double:
298 87.375
The formatted output stored in array s is:
```



integer:

double:

298

87.38

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

```
Outline
fig08_16.c
```

double: 87.375

8.6 String Manipulation Functions of the String Handling Library

- String handling library has functions to
 - Manipulate string data
 - Search strings
 - Tokenize strings
 - Determine string length

,	Function description
*SI, const char *S2)	Copies string s2 into array s1. The value of s1 is returned.
<pre>"s1, const char *s2, size_t n)</pre>	Copies at most n characters of string s2 into array s1. The value of s1 is returned.
*SI, Const Char *S2)	Appends string s2 to array s1. The first character of s2 overwrites the terminating null character of s1. The value of s1 is returned.
<pre>char *strncat(char *s1, const char *s2, size_t n)</pre>	Appends at most n characters of string s2 to array s1. The first character of s2 overwrites the terminating null character of s1. The value of s1 is returned.

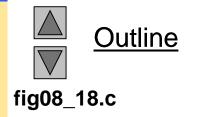


```
1 /* Fig. 8.18: fig08_18.c
      Using strcpy and strncpy */
3 #include <stdio.h>
4 #include <string.h>
6 int main()
7 {
      char x[] = "Happy Birthday to You"; /* initialize char array x */
      char y[ 25 ];
                                         /* create char array y */
      char z[ 15 ];
                                          /* create char array z */
10
11
      /* copy contents of x into y */
12
      printf( "%s%s\n%s%s\n",
13
              "The string in array x is: ", x,
14
              "The string in array y is: ", strcpy( y, x ) );
15
16
      /* copy first 14 characters of x into z. Does not copy null
17
         character */
18
      strncpy( z, x, 14 );
19
20
      z[14] = '\0'; /* append '\0' to z's contents */
21
      printf( "The string in array z is: %s\n", z );
22
23
      return 0; /* indicates successful termination */
24
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
      Using strcat and strncat */
3 #include <stdio.h>
4 #include <string.h>
5
6 int main()
7 {
      char s1[ 20 ] = "Happy "; /* initialize char array s1 */
8
      char s2[] = "New Year "; /* initialize char array s2 */
9
      char s3[ 40 ] = ""; /* initialize char array s3 */
10
11
      printf( "s1 = %s\ns2 = %s\n", s1, s2 );
12
13
      /* concatenate s2 to s1 */
14
      printf( "strcat( s1, s2 ) = %s\n", strcat( s1, s2 );
15
16
      /* concatenate first 6 characters of s1 to s3. Place '\0'
17
         after last character */
18
      printf( "strncat( s3, s1, 6 ) = %s\n", strncat( s3, s1, 6 ) );
19
20
      /* concatenate s1 to s3 */
21
      printf( "strcat( s3, s1 ) = %s\n", strcat( s3, s1 );
22
23
      return 0; /* indicates successful termination */
24
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
```



8.7 Comparison Functions of the String Handling Library

- Comparing strings
 - Computer compares numeric ASCII codes of characters in string
 - Appendix D has a list of character codes

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

- Compares string s1 to s2
- Returns a negative number if s1 < s2, zero if s1 == s2 or a positive number if s1 > s2

- Compares up to n characters of string s1 to s2
- Returns values as above

[©] Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.



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

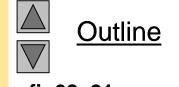


fig08_21.c

25 } /* end main */

```
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 Search Functions of the String Handling Library

Function prototype	Function description
<pre>char *strchr(const char *s, int c);</pre>	Locates the first occurrence of character c in string s. If c is found, a pointer to c in s is returned. Otherwise, a NULL pointer is returned.
<pre>size_t strcspn(const char *s1, const char *s2);</pre>	Determines and returns the length of the initial segment of string s1 consisting of characters not contained in string s2.
<pre>size_t strspn(const char *s1, const char *s2);</pre>	Determines and returns the length of the initial segment of string s1 consisting only of characters contained in string s2.
<pre>char *strpbrk(const char *s1, const char *s2);</pre>	Locates the first occurrence in string s1 of any character in string s2. If a character from string s2 is found, a pointer to the character in string s1 is returned. Otherwise, a NULL pointer is returned.
<pre>char *strrchr(const char *s, int c);</pre>	Locates the last occurrence of c in string s. If c is found, a pointer to c in string s is returned. Otherwise, a NULL pointer is returned.
<pre>char *strstr(const char *s1, const char *s2);</pre>	Locates the first occurrence in string s1 of string s2. If the string is found, a pointer to the string in s1 is returned. Otherwise, a NULL pointer is returned.
<pre>char *strtok(char *s1, const char *s2); © Copyright 1992–2004 by Deitel & Association</pre>	A sequence of calls to strtok breaks string s1 into "tokens"—logical pieces such as words in a line of text—separated by characters contained in string s2. The first call contains s1 as the first argument, and subsequent calls to continue tokenizing the same string contain NULL as the first argument. A pointer to the current token is returned by each call. If there are no more tokens when the function is called, NULL is returned.

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



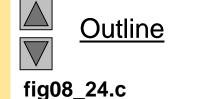
fig08_23.c (Part 1 of 2)

21

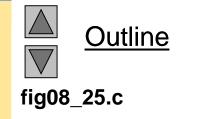
```
/* if character2 was found in string */
      if ( strchr( string, character2 ) != NULL ) {
23
         printf( "\'%c\' was found in \"%s\".\n",
24
                 character2, string );
25
      } /* end if */
26
      else { /* if character2 was not found */
27
         printf( "\'%c\' was not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

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

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



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



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

```
Outline
fig08_26.c
```

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

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



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

```
Outline
fig08_28.c
```

first occurrence of string2 is: defabcdef

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



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 Memory Functions of the Stringhandling Library

- Memory Functions
 - In <stdlib.h>
 - Manipulate, compare, and search blocks of memory
 - Can manipulate any block of data
- Pointer parameters are void *
 - Any pointer can be assigned to void *, and vice versa
 - void * cannot be dereferenced
 - Each function receives a size argument specifying the number of bytes (characters) to process

8.9 Memory Functions of the Stringhandling Library

Function prototype	Function description
<pre>void *memcpy(void *s1, const void *s2, size_t n);</pre>	Copies n characters from the object pointed to by s2 into the object pointed to by s1. A pointer to the resulting object is returned.
<pre>void *memmove(void *s1, const void *s2, size_t n);</pre>	Copies n characters from the object pointed to by s2 into the object pointed to by s1. The copy is performed as if the characters were first copied from the object pointed to by s2 into a temporary array and then from the temporary array into the object pointed to by s1. A pointer to the resulting object is returned.
<pre>int memcmp(const void *s1, const void *s2, size_t n);</pre>	Compares the first n characters of the objects pointed to by s1 and s2. The function returns 0, less than 0 or greater than 0 if s1 is equal to, less than or greater than s2.
<pre>void *memchr(const void *s, int c, size_t n);</pre>	Locates the first occurrence of c (converted to unsigned char) in the first n characters of the object pointed to by s. If c is found, a pointer to c in the object is returned. Otherwise, NULL is returned.
<pre>void *memset(void *s, int c, size_t n);</pre>	Copies c (converted to unsigned char) into the first n characters of the object pointed to by s. A pointer to the result is returned.

[©] Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.



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

```
Outline
fig08_31.c
```

After s2 is copied into s1 with memcpy,

s1 contains "Copy this string"

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

```
Outline
fig08_32.c
```

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

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

```
Outline
fig08_33.c
```

memcmp(s1, s2, 7) = -1 memcmp(s2, s1, 7) = 1

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

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

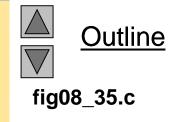
```
Outline
Fig8_34.c
```

8.10 Other Functions of the String Handling Library

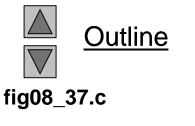
- char *strerror(int errornum);
 - Creates a system-dependent error message based on errornum
 - Returns a pointer to the string
- size_t strlen(const char *s);
 - Returns the number of characters (before NULL) in string s



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



```
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 */
No such file or directory
```



```
1 /* Fig. 8.38: fig08_38.c
      Using strlen */
 #include <stdio.h>
 #include <string.h>
5
6 int main()
7 {
      /* initialize 3 char pointers */
8
      const char *string1 = "abcdefghijk1mnopqrstuvwxyz";
      const char *string2 = "four";
10
      const char *string3 = "Boston";
11
12
      printf("%s\"%s\"%s\"u\n%s\"%s\"%s\"u\n",
13
         "The length of ", string1, " is ",
14
         ( unsigned long ) strlen( string1 ),
15
         "The length of ", string2, " is ",
16
         ( unsigned long ) strlen( string2 ),
17
         "The length of ", string3, " is ",
18
         ( unsigned long ) strlen( string3 ) );
19
20
      return 0; /* indicates successful termination */
21
22
23 } /* end main */
The length of "abcdefghijklmnopgrstuvwxyz" is 26
The length of "four" is 4
The length of "Boston" is 6
```

