# Chapter 8

# Characters and Strings

© Copyright 2007 by Deitel & Associates, Inc. and Pearson Education Inc.

All Rights Reserved.

# Chapter 8 - Characters and Strings

**Outline**

# 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

# 8.2 Fundamentals of Strings and Characters

- **Characters**
  - A **char** is also an `int` value represented as a character in single quotes
  - `'z'` represents the integer value of letter `z` (ASCII 122)

- **Strings**
  - Series of characters treated as a single unit
    - Can include letters, digits and special characters `(*, /, $)`
  - String literal is written in double quotes such as `"Hello"`
  - Strings are arrays of characters
    - String variable is a pointer to first character
    - Value of string is the address of first character

# 8.2 Fundamentals of Strings and Characters

- String definitions
  - Define as a character array or a variable of type `char *`

```
char str1[ ] = "Hello";
char *str1Ptr = "Hello";
```

  - Remember that strings represented as character arrays end with `'\0'`
    - `variable str1` has `6 elements`

- Inputting strings
  - Use `scanf`

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

    - Copies keyboard input into `word[]`
    - Do not need & (because a string is a pointer)
  - Remember to leave room in the array for `'\0'`

# String Initializing

```
char *isim;
isim = "Mehmet";            // Valid, but not preferred
strcpy(isim, "Mehmet");     // Preferred
```

```
char isim[20];
isim = "Mehmet";            // Compiler error!
strcpy(isim, "Mehmet");     // Correct
```

[Error]
Incompatible types in assignment of 'const char [7]' to 'char [20]'

```
char isim[] =  {'M','e','h','m','e','t',' ','U','s','l','u','\0'};
```

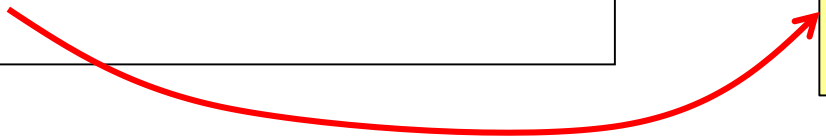
# Example: Copying a String variable into another variable

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

int main() {
 char  isim1[20] = "Mehmet Uslu";
 char  isim2[20];

 strcpy(isim2, isim1);
 printf("%s \n %s \n", isim1, isim2);
}
```

```c
isim2 = isim1; // Compiler error!
```



[Error] Invalid array assignment

# Example: Copying an Array of Strings into another Array

```
#include <stdio.h>
#include <string.h>
#define  N  3    // Number of persons

int main()
{
 int i;
 char  liste1[N][20] = {"Ahmet Gokce",
                        "Fatih Coskun",
                        "Mehmet Uslu"};
 char  liste2[N][20];

 for (i=0; i < N; i++)
 {
     strcpy(liste2[i], liste1[i]);
     printf("%s \n", liste2[i]);
 }

} // end main
```

# Example: ASCII char and int values

```c
#include <stdio.h>

int main()
{
  char w = 'A';  // ASCII 65
  char x = 65;   // ASCII 'A'

  int  y = 'C';  // ASCII 67
  int  z = w+2;  // ASCII 67

  printf("w = %c  %d \n", w, w);
  printf("x = %c  %d \n", x, x);
  printf("y = %c  %d \n", y, y);
  printf("z = %c  %d \n", z, z);

}
```

Program Output

```
w = A  65

x = A  65

y = C  67

z = C  67
```

# Example: Displaying entire ASCII table

```
#include <stdio.h>

int main()
{
  int  i ;

  for ( i = 0 ; i <= 255 ; i++ )
       printf ( "%d    %c\n", i, i ) ;

} // end main
```

# 8.3 Character Handling Library

- Character handling library
  - Includes functions to perform useful tests and manipulations of character data
  - Each function receives **one character (an `int`)** or `EOF` as an argument

- The following slide contains a table of all the character handling functions in `<ctype.h>`

# 8.3 Character Handling Library

| Prototype | Description |
|---|---|
| `int isdigit( int c );` | Returns `true` if `c` is a digit and `false` otherwise. |
| `int isalpha( int c );` | Returns `true` if `c` is a letter and `false` otherwise. |
| `int isalnum( int c );` | Returns `true` if `c` is a digit or a letter and `false` otherwise. |
| `int isxdigit( int c );` | Returns `true` if `c` is a hexadecimal digit character and `false` otherwise. |
| `int islower( int c );` | Returns `true` if `c` is a lowercase letter and `false` otherwise. |
| `int isupper( int c );` | Returns `true` if `c` is an uppercase letter; `false` otherwise. |
| `int tolower( int c );` | If `c` is an uppercase letter, `tolower` returns `c` as a lowercase letter. Otherwise, `tolower` returns the argument unchanged. |
| `int toupper( int c );` | If `c` is a lowercase letter, `toupper` returns `c` as an uppercase letter. Otherwise, `toupper` returns the argument unchanged. |
| `int isspace( int c );` | 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 |
| `int iscntrl( int c );` | Returns `true` if `c` is a control character and `false` otherwise. |
| `int ispunct( int c );` | Returns `true` if `c` is a printing character other than a space, a digit, or a letter and `false` otherwise. |
| `int isprint( int c );` | Returns `true` value if `c` is a printing character including space (`' '`) and `false` otherwise. |
| `int isgraph( int c );` | Returns `true` if `c` is a printing character other than space (`' '`) and `false` otherwise. |

- False means returned int is 0
- True means returned int is > 0

# Example: isdigit() function

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

int main()
{
  int sonuc1, sonuc2;
  sonuc1 = isdigit('7');
  sonuc2 = isdigit('a');

  printf("sonuc1= %d \n", sonuc1);
  printf("sonuc2= %d \n", sonuc2);

}
```

Program Output

```
sonuc1= 4
sonuc2= 0
```

Nonzero means true, zero means false.

# Example : <ctype.h> functions

Part 1 of 2

```c
/* Fig. 8.2: fig08_02.c
   Using functions isdigit, isalpha, isalnum, and isxdigit */
#include <stdio.h>
#include <ctype.h>

int main()
{
   printf( "%s\n%s%s\n%s%s\n\n", "According to isdigit: ",
       isdigit( '8' ) ? "8 is a " : "8 is not a ", "digit",
       isdigit( '#' ) ? "# is a " : "# is not a ", "digit" );

   printf( "%s\n%s%s\n%s%s\n%s%s\n%s%s\n\n",
       "According to isalpha:",
       isalpha( 'A' ) ? "A is a " : "A is not a ", "letter",
       isalpha( 'b' ) ? "b is a " : "b is not a ", "letter",
       isalpha( '&' ) ? "& is a " : "& is not a ", "letter",
       isalpha( '4' ) ? "4 is a " : "4 is not a ", "letter" );

   printf( "%s\n%s%s\n%s%s\n%s%s\n\n",
       "According to isalnum:",
       isalnum( 'A' ) ? "A is a " : "A is not a ",
       "digit or a letter",
       isalnum( '8' ) ? "8 is a " : "8 is not a ",
       "digit or a letter",
       isalnum( '#' ) ? "# is a " : "# is not a ",
       "digit or a letter" );
```

Part 2 of 2

```
 printf( "%s\n%s%s\n%s%s\n%s%s\n%s%s\n%s%s\n",
       "According to isxdigit:",
       isxdigit( 'F' ) ? "F is a " : "F is not a ",
       "hexadecimal digit",
       isxdigit( 'J' ) ? "J is a " : "J is not a ",
       "hexadecimal digit",
       isxdigit( '7' ) ? "7 is a " : "7 is not a ",
       "hexadecimal digit",
       isxdigit( '$' ) ? "$ is a " : "$ is not a ",
       "hexadecimal digit",
       isxdigit( 'f' ) ? "f is a " : "f is not a ",
       "hexadecimal digit" );

} // 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
```

• In Deitel chapter3, we have already seen the ternary conditional operator:

```
char x = '8';
printf("%c  %s \n", x,
  isdigit(x)  ?  "is a digit"  :  "is not a digit");
```

• The following is the equivalent if statement:

```
char x = '8';
if (isdigit(x))
    printf("%c  is a digit \n", x);
else
    printf("%c  is not a digit \n", x);
```

# 8.4 String Conversion Functions

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

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

# Example : atof() (ascii to float)

```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 );
} // end main
```

Program Output

```
The string "99.0" converted to double is 99.000

The converted value divided by 2 is 49.500
```

# Example : atoi() (ascii to integer)

```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 );
} // end main
```

Program Output

```
The string "2593" converted to int is 2593

The converted value minus 593 is 2000
```

**IMPORTANT :**

- String conversion functions such as atoi() require a null terminated **string** argument.
- They do not work for a single char.

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

int main()
{
  printf("%d \n", atoi("5") ); // Correct
  printf("%d \n", atoi('5') );  // Wrong!

}
```

# 8.5 Standard Input/Output Library Functions

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

| Function prototype | Function description |
|---|---|
| `int getchar( void );` | Inputs the next character from the standard input and returns it as an integer. |
| `char *gets( char *s );` | 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. |
| `int putchar( int c );` | Prints the character stored in `c`. |
| `int puts( const char *s );` | Prints the string `s` followed by a newline character. |
| `int sprintf( char *s, const char *format, ... );` | Equivalent to `printf`, except the output is stored in the array `s` instead of printing it on the screen. |
| `int sscanf( char *s, const char *format, ... );` | Equivalent to `scanf`, except the input is read from the array `s` instead of reading it from the keyboard. |

# Example : Reversing a string with recursive method

Part 1 of 2

```c
/* Fig. 8.13: fig08_13.c
   Using gets and putchar */
#include <stdio.h>
#include <stdlib.h>

void reverse( const char * const sPtr ); // prototype

int main()
{
   char sentence[ 80 ]; // create char array

   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 );
} // end main
```

Part 2 of 2

```
// recursively outputs characters in string in reverse order
void reverse( const char * const sPtr )
{
   // if end of the string
   if ( sPtr[ 0 ] == '\0' ) { // base case
      return;
   } // end if
   else { // if not end of the string
      reverse( &sPtr[ 1 ] ); // recursion step

      putchar( sPtr[ 0 ] ); // use putchar to display character
   } // end else

} // end function reverse
```

Program Output

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

The line printed backwards is:
sgnirtS dna sretcarahC
```

# Example : getchar()
# (Get one character at a time)

```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'; // terminate string

   // use puts to display sentence
   puts( "\nThe line entered was:" );
   puts( sentence );
} // end main
```

Program Output

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

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

# Example : sprintf() (string print formatted)

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

int main()
{
   char s[ 80 ]; // create char array
   int x;        // x value to be input
   double y;     // y value to be input

   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 );
} // 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
```

# Example : sscanf()
# (string scan formatted)

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

int main()
{
   char s[] = "31298 87.375"; // initialize array s
   int x;    // x value to be input
   double y; // y value to be input

   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 );
} // end main
```

Program Output

```
The values stored in character array s are:

integer: 31298
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 prototype | Function description |
|---|---|
| `char *strcpy( char *s1, const char *s2 )` | Copies string `s2` into array `s1`. The value of `s1` is returned. |
| `char *strncpy( char *s1, const char *s2, size_t n )` | Copies at most `n` characters of string `s2` into array `s1`. The value of `s1` is returned. |
| `char *strcat( char *s1, 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. |
| `char *strncat( char *s1, const char *s2, size_t n )` | 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. |

# Example: strlen() (string length)

- `size_t  strlen( const char *s );`
  - Returns the number of characters (before `NULL`) in string `s`
  - `size_t` is a defined type of integer

```
/* 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 ", strlen( string1 ),
      "The length of ", string2, " is ", strlen( string2 ),
      "The length of ", string3, " is ", strlen( string3 ) );
} // end main
```

Program Output

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

# Example : String Copying

```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'; /* terminate string in z */
   printf( "The string in array z is: %s\n", z );
} // end main
```

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
```

# Example : String Concatenating

```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 to empty

   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 ) );
} // end main
```

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 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`

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

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

# Example : String Comparing

```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 ) );
} // end main
```

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 Search Functions of the String Handling Library

| Function prototype | Function description |
|---|---|
| `char *strchr( const char *s, int c );` | 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. |
| `size_t strcspn( const char *s1, const char *s2 );` | Determines and returns the length of the initial segment of string `s1` consisting of characters not contained in string `s2`. |
| `size_t strspn( const char *s1, const char *s2 );` | Determines and returns the length of the initial segment of string `s1` consisting only of characters contained in string `s2`. |
| `char *strpbrk( const char *s1, const char *s2 );` | 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. |
| `char *strrchr( const char *s, int c );` | 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. |
| `char *strstr( const char *s1, const char *s2 );` | 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. |
| `char *strtok( char *s1, const char *s2 );` | 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. |

# Example : strchr()
# (Searching char in string)

Part 1 of 2

```c
/* Fig. 8.23: fig08_23.c
   Using strchr */
#include <stdio.h>
#include <string.h>

int main()
{
   const char *string = "This is a test"; // initialize char pointer
   char character1 = 'a'; // initialize character1
   char character2 = 'z'; // initialize character2

   // if character1 was found in string
   if ( strchr( string, character1 ) != NULL ) {
      printf( "\'%c\' was found in \"%s\".\n",
         character1, string );
   } // end if
   else { // if character1 was not found
      printf( "\'%c\' was not found in \"%s\".\n",
         character1, string );
   } // end else
```

Part 2 of 2

```
   // if character2 was found in string
   if ( strchr( string, character2 ) != NULL ) {
      printf( "\'%c\' was found in \"%s\".\n",
         character2, string );
   } // end if
   else { // if character2 was not found
      printf( "\'%c\' was not found in \"%s\".\n",
         character2, string );
   } // end else

} // end main
```

Program Output

```
'a' was found in "This is a test".

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

# Example : strstr()
# (Searching string2 in string1)

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

int main()
{
   const char *string1 = "abcdefabcdef"; // string to search
   const char *string2 = "def"; // string to search for

   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 ) );

} // end main
```

Program Output

```
string1 = abcdefabcdef
string2 = def

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

# Example : strtok() (string tokenizing)

- Tokenizing means parsing a sentence into its words.

```
/* 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
} // end main
```

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
```

# Example: Tokenizing by using a set of seperators

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

int main()
{
   char string[] = "aaa bbb,ccc;ddd?eee-fff.ggg! hhh";
   char *tokenPtr;
   char *ayraclar = " ,;.!?-"; // Set of word seperators

   printf( "String : %s \n", string);

   tokenPtr = strtok( string, ayraclar);
   while ( tokenPtr != NULL ) {
      printf( "%s\n", tokenPtr );
      tokenPtr = strtok( NULL, ayraclar);
   }

} // end main
```

Program Output

```
String : aaa bbb,ccc;ddd?eee-uuu.nnn! yyy

aaa
bbb
ccc
ddd
eee
uuu
nnn
yyy
```

# Example : strerror() (string of error)

- `char * strerror( int errornum );`
  - Creates a system-dependent error message based on `errornum`
  - Returns a pointer to the string

```
/* Fig. 8.37: fig08_37.c
   Using strerror */
#include <stdio.h>
#include <string.h>

int main()
{
   printf( "%s\n", strerror( 2 ) );

}
```

Program Output

```
No such file or directory
```

# Example: Displaying all system errors

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

int main()
{
   int i;

   for (i=0; i < 50; i++)
       printf( "Error Number = %d  %s \n", i, strerror( i ) );

} // end main
```

# Program Output

```
Error Number = 0  No error
Error Number = 1  Operation not permitted
Error Number = 2  No such file or directory
Error Number = 3  No such process
Error Number = 4  Interrupted function call
Error Number = 5  Input/output error
Error Number = 6  No such device or address
Error Number = 7  Arg list too long
Error Number = 8  Exec format error
Error Number = 9  Bad file descriptor
Error Number = 10  No child processes
Error Number = 11  Resource temporarily unavailable
Error Number = 12  Not enough space
Error Number = 13  Permission denied
Error Number = 14  Bad address
Error Number = 15  Unknown error
Error Number = 16  Resource device
Error Number = 17  File exists
Error Number = 18  Improper link
Error Number = 19  No such device
Error Number = 20  Not a directory
Error Number = 21  Is a directory
Error Number = 22  Invalid argument
Error Number = 23  Too many open files in system
Error Number = 24  Too many open files
```

```
Error Number = 25  Inappropriate I/O control operation
Error Number = 26  Unknown error
Error Number = 27  File too large
Error Number = 28  No space left on device
Error Number = 29  Invalid seek
Error Number = 30  Read-only file system
Error Number = 31  Too many links
Error Number = 32  Broken pipe
Error Number = 33  Domain error
Error Number = 34  Result too large
Error Number = 35  Unknown error
Error Number = 36  Resource deadlock avoided
Error Number = 37  Unknown error
Error Number = 38  Filename too long
Error Number = 39  No locks available
Error Number = 40  Function not implemented
Error Number = 41  Directory not empty
Error Number = 42  Illegal byte sequence
Error Number = 43  Unknown error
Error Number = 44  Unknown error
Error Number = 45  Unknown error
Error Number = 46  Unknown error
Error Number = 47  Unknown error
Error Number = 48  Unknown error
Error Number = 49  Unknown error
```