Chapter 8: Characters and Strings

* A string in C is an array of characters ending in the null character ('\0').
* A string is accessed via a pointer to the first character in the string.
* The following declarations are the same,

```
        char color1[] = "blue";
        char *color2 = "blue";
        char color3[] = {'b','l','u','e','\0'};
```

* When declaring a character array to contain a string, the array must be large enough to store the string and its terminating "NULL" character.

* Function "scanf" will read characters until a space, newline, or end-of-file indicator is encountered.
* E.g.

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

* For a character array to be printed as a string, the array must contain a terminating "NULL" character.

Character Handling Library

* The character handling library includes several functions that perform useful tests and manipulations of character data.
* "EOF" normally has the value "-1" and some hardware architectures do not allow negative values to be stored in "char" (0-255) variables.
* Therefore, the character handling functions manipulate characters as integers.
* When using functions from the character handling library, include the "<ctype.h>" header file.

int isdigit(int c)	true if c is a digit
int isalpha(int c)	true if c is a letter
int isalnum(int c)	true if c is a digit or a letter
int isxdigit(int c)	true if c us a hexadecimal digit character
int islower(int c)	true if c is a lowercase letter
int isupper(int c)	true if c is an uppercase letter
int tolower(int c)	return lowercase of c
int toupper(int c)	return uppercase of c
int isspace(int c)	true if c is a white-space character
int iscntrl(int c)	true if c is a control character
int ispunct(int c)	true if c is a printing character other than a space, a digit, or a letter, and 0 otherwise
int isprint(int c)	true if c is a printing character including space
int isgraph(int c)	true if c is a printing character other than space

String Conversion Functions

* The string conversion functions is come from the general utilities library ("stdlib").
* When using functions from the general utilities library, include the "<stdlib.h>" header file.

double atof(const char *nPtr)	converts the string nPtr to

```
                                    double
    int atoi(const char *nPtr)          converts the string nPtr to
                                    int
    long atol(constr 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
```

* If the converted value cannot be represented, the behavior is undefined.
* "**endPtr" is the pointer of the location of the first character after the converted portion of the string.
* E.g.

```
        #include <stdio.h>
        #include <stdlib.h>
        main()
        {
            double d;
            char *string = "51.2% are admitted";
            char *stringPtr;

            d = strtod(string, &stringPtr);
            printf("The string \"%s\" is converted to the\n",
                    string);
            printf("double value %.2f and the string \"%s\"\n",
                    d, stringPtr);
            return 0;
        }
```

* "base" can be specified as 0(any base), or any value between 2 and 36.

Standard Input/Output Library Functions
----
* When using functions from the standard input/output library, include the <stdio.h> header file.

```
    int getchar(void)       input the next character from the
                            standard input and return it as integer
    char *gets(char *s)     input characters from the standard input
                            into the array s until a newline or EOF
                            is encountered. A terminating NULL
                            character is appended to the array.
    int putchar(int c)      Print the character stored in c
    int puts(const char *s) Print 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
                            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
                            from the keyboard
```

```
String Manipulation Functions of the String Handling Library
------------------------------------------------------------
* When using functions from the string handling library, include the
<string.h> header file.
    char *strcpy(char *s1, const char *s2)
                            Copies the string s2 into the array s1.
                            The value of s1 is returned
    char *strncpy(char *s1, const char *s2, size_t n)
                            Copies at most n characters of the string
                            s2 into the array s1. The value of s1 is
                            returned
    char *strcat(char *s1, const char *s2)
                            Appends the string s2 to the 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 the
                            string s2 to the array s1. The first
                            character of s2 overwrites the
                            terminating NULL character of s1. The
                            value of s1 is returned

Comparison Functions of the String Handling Library
---------------------------------------------------
    int strcmp(const char *s1, const char *s2)
                            Compares the string s1 to the string s2.
                            The function returns 0, less than 0 (-1),
                            or greater than 0 (1) if s1 is equal to,
                            less than, or greater than s2,
                            respectively
    int strncmp(const char *s1, const char *s2)
                            Compares up to n characters of the string
                            s1 to the string s2. The function returns
                            0, less than 0 (-1), or greater than 0
                            (1) if s1 is equal to, less than, or
                            greater than s2, respectively
* When the computer compares two strings, it actually compares the
numeric codes of the characters in the strings.

Memory Functions of the String Handling Library
-----------------------------------------------
* The functions treat blocks of memory as character arrays.
* These functions can manipulate any block of data.
    void *memcpy(void *s1, const void *s2, size_t n)
                            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.
    void *memmove(void *s1, const void *s2, size_t n)
                            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 are first copied from the
                            object pointed to by s2 into a temporary
                            array, then from the temporary array into
                            the object pointed to by s1. A pointer to
                            the resulting object is returned.
    int memcmp(const void *s1, const void *s2, size_t n)
```

```
                            Compares the first n characters of the
                            object pointed to by s1 and s2. The
                            function return 0, less than 0 (-1), or
                            greater than 0 (1) if s1 is equal to,
                            less than, or greater than s2
    void *memchr(const void *s, int c, size_t n)
                            Locates the first occurence of c
                            (converted to unsigned char) in the first
                            n charcters of the object pointed to by
                            s. If c is found, a pointer to c in the
                            object is returned. Otherwise NULL is
                            returned.
    void *memset(void *s, int c, size_t n)
                            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.

Other Functions of the String Handling Library
----------------------------------------------
    char *strerror(int errornum)    Maps errornum into a full text
                                    string in a system dependent
                                    manner. A pointer to the string
                                    is returned.
    size_t strlen(const char *s)    Determines the length of string
                                    s. The number of characters
                                    preceding the terminating NULL
                                    character is returned.
* The message generated by "strerror" is system dependent.
```