Chapter 13: Character and Bit Manipulation

UPPER AND LOWER CASE

Examine the program named uplow.c for an example of a program that does lots of character manipulation. More specifically, it changes the case of alphabetic characters. It illustrates the use of four functions that have to do with case. It should be no problem for you to study this program on your own and understand how it works. The four functions on display in this program are all within the user written function, **mix_up_the_chars()**. Compile and run the program with the file of your choice. The four functions are;

```
isupper(c);  Is the character upper case ?
islower(c);  Is the character lower case ?
toupper(c);  Make the character upper case.
tolower(c);  Make the character lower case.
```

Many more classification and conversion routines are listed in the reference material for your compiler. You should spend time studying these at this time to get an idea of what functions are available.

CLASSIFICATION OF CHARACTERS

Load and display the next program, charclas.c for an example of character counting. We have repeatedly used the backslash n character representing a new line. These are called escape sequences, and some of the more commonly used are defined in the following table;

```
\n  Newline
\t  Tab
\"  Double quote
\\  Backslash
\0  NULL (zero)
```

Consult your compiler documentation for a complete list of escape sequences available with your compiler.

By preceding each of the above characters with the backslash character, the character can be included in a line of text for display, or printing. In the same way that it is perfectly all right to use the letter n in a line of text as a part of someone's name, and as an end-of-line, the other characters can be used as parts of text or for their particular functions.

This example program uses three of the functions that can determine the class of a character, and counts the characters in each class. The number of each class is displayed along with the line itself. The three functions are as follows;

```
isalpha(c); Is the character alphabetic ?
isdigit(c); Is the character a numeral ?
isspace(c); Is the character any of, \n, \t, or blank ?
```

As noted above, many more classification routines are available with your compiler.

This program should be simple for you to find your way through, so no explanation will be given. It was necessary to give an example with these functions used. Compile and run this program with any file you choose.

THE LOGICAL FUNCTIONS

Load and display the program bitops.c. The functions in this group of functions are used to do bitwise operations, meaning that the operations are performed on the bits as though they were individual bits. No carry from bit to bit is performed as would be done with a binary addition. Even though the operations are performed on a single bit basis, an entire byte or integer variable can be operated on in one instruction. The operators and the operations they perform are given in the following table;

```
&  Logical AND, if both bits are 1, the result is 1.
|  Logical OR, if either bit is one, the result is 1.
^  Logical XOR, (exclusive OR), if one and only one
             bit is 1, the result is 1.
~  Logical invert, if bit is 1, the result is 0, and
            if bit is 0, the result is 1.
```

The example program uses several fields that are combined in each of the ways given above. The data is in hexadecimal format. It will be assumed that you already know hexadecimal format if you need to use these operations. If you don't, you will need to study it on your own. Teaching the hexadecimal format of numbers is beyond the scope of this tutorial. Be sure to compile and execute this program and observe the output.

THE SHIFT INSTRUCTIONS

The last two operations to be covered in this chapter are the left shift and the right shift instructions. Load the example program shifter.c for an example using these two instructions. The two operations use the following operators;

```
<<  n Left shift n places.
>>  n Right shift n places.
```

Once again the operations are carried out and displayed using the hexadecimal format. The program should be simple for you to understand on your own, there is no tricky code.

Index Previous Next . . . . . *C Tutorial*

The Webwizard