CSCI 240 Lecture Notes - Part 6

More on the char data type

We have seen that C++ has a built-in data type to represent **single** characters. It is one of the fundamental data types of the language.

Here's a surprise - internally, a character is represented by a numeric code in the range of 0 to 255. (32 to 127 are standard; some of 0 to 31 and all of 128 to 255 depend on the computer you use).

Since each *char* is associated with an integer value, in C++ chars can be treated as integers. This is unusual since usually C++ does not permit us to mix data types. We will see some examples of this below.

To review:

The data type name is **char**. To declare a char variable:

```
char ch;
```

A char literal is written with **single** quotes around it: 'a'

cout knows how to display char data:

```
char ch = 'a';
```

```
cout << ch;
```

Some special cases:

A backslash char is written: `'\\'` - single quote, backslash, backslash, single quote

A single quote char is written as: `'\''` - single quote, backslash, single quote, single quote

A double quote char is written as: `'\"'` - single quote, backslash, double quote, single quote

The common "escape sequence" characters (newline, tab, etc.) are also written with single quotes, but are still typed with the leading \: `'\n'`

```
ch = '\n';         //stores newline char in ch
```

Other chars with no keyboard representation are written in base 8 using three digits: `'\007'`. This is not covered further in this course.

The ASCII Character Set

The ***American Standard Code for Information Interchange (ASCII)*** is a standard mapping of integer values to characters, used by most computer systems (exception - IBM mainframes).

Some useful ones to know (i.e. memorize these for this course)

```
blank     = 32
digit '0' = 48
digit '9' = 57
'A'       = 65
'Z'       = 90
'a'       = 97
'z'       = 122
```

Note: '1' is 49, '2' is 50, etc. so if you know '0' you know all the digits. 'B' is 66, 'C' is 67, 'b' is 98, 'c' is 99, etc. so you can figure out any letter, too.

Since chars are really small ints (internally) we can do some useful tricks with them by treating them as integers.

```
char ch = 'A';
int i;

//print Ascii value of ch (65). Note the typecast

cout << (int) ch;


// print ASCII values of A..Z

for (i = 'A'; i <= 'Z'; i++)
  {
  cout << "\nThe ASCII value of " << (char) i << " is " << i;
  }
```

will print:

```
The ASCII value of A is 65
The ASCII value of B is 66
The ASCII value of C is 67
...
```

Character Functions

There is a large set of standard C++ functions that deal exclusively with characters. They can be found in the *<cctype>*, which is automatically included in most IDEs, but if it's not, just add a *#include <cctype>* at the top of the source code file.

One subset of the standard C++ character functions test whether a character *IS* of a certain type: capital, lower case, digit, punctuation, whitespace, etc.

For example, if you want to test if a character entered by the user *is a digit*, you can call

```
if ( isdigit(ch) )        //ch is a char variable
  {
  cout << ch << " is a digit";
  }
else
  {
  cout << ch << " is NOT a digit";
  }
```

Some of the other "is" functions:

Function	Purpose
isdigit	This function tests if the passed in character is a digit. It returns *true* if the character is a digit and *false* otherwise.
isalpha	This function tests if the passed in character is alphabetic. It returns *true* if the character is a alphabetic and *false* otherwise.
isupper	This function tests if the passed in character is an uppercase alphabetic character. It returns *true* if the character is an uppercase character and *false* otherwise.
islower	This function tests if the passed in character is a lowercase alphabetic character. It returns *true* if the character is a lowercase character and *false* otherwise.
isalnum	This function tests if the passed in character is an alphanumeric character. It returns *true* if the character is alphanumeric and *false* otherwise.
isspace	This function tests if the passed in character is a whitespace character (space, tab, newline, etc...). It returns *true* if the character is a whitespace character and *false* otherwise.
ispunct	This function tests if the passed in character is a punctuation character. It returns *true* if the character is punctuation and *false* otherwise.

Another subset of the character functions changes a character. Two of those functions change an alphabetic character to the opposite case. They are:

- `toupper`
- `tolower`

The *toupper* function takes a character as its argument and returns a character. If the passed-in character is a lowercase alphabetic character, then the uppercase version will be returned. If the passed-in character is not a lowercase alphabetic character, it is returned unchanged.

The *tolower* function takes a character as its argument and returns a character. If the passed-in character is an uppercase alphabetic character, then the lowercase version will be returned. If the passed-in character is not an uppercase alphabetic character, it is returned unchanged.

Suppose you have a single *char* (not a string) from the user. If it is 'q' or 'Q', that signals quit. Before we had to code something like:

```
while (ch != 'q' && ch != 'Q')
  {
  // do something
  }
```

Now we could do this:

```
while (toupper(ch) != 'Q')
  {
  // do something
  }
```

In other words, ch is converted to uppercase and the result of that is compared to 'Q'.

Notice, too, that we can compare chars with !=, = =, <=, etc. just like numbers (since they are small ints).