Lesson 9: Strings
In C++ strings are really arrays, but there are some different functions that are used for strings, like adding to strings, finding the length of strings, and also of checking to see if strings match.

The definition of a string would be anything that contains more than one character strung together. For example, "This" is a string. However, single characters will not be strings, though they can be used as strings.

Strings are arrays of chars. Static strings are words surrounded by double quotation marks.

"This is a static string"

To declare a string of 50 letters, you would want to say:

char string[50];

This would declare a string with a length of 50 characters. Do not forget that arrays begin at zero, not 1 for the index number. In addition, a string ends with a null character, literally a '/0' character. However, just remember that there will be an extra character on the end on a string. It is like a period at the end of a sentence, it is not counted as a letter, but it still takes up a space. Technically, in a fifty char array you could only hold 49 letters and one null character at the end to terminate the string.

TAKE NOTE: char *arry;
Can also be used as a string. If you have read the tutorial on pointers, you can do something such as:

arry = new char[256];

which allows you to access arry just as if it were an array. Keep in mind that to use delete you must put [] between delete and arry to tell it to free all 256 bytes of memory allocated.

For example,
delete [] arry.

Strings are useful for holding all types of long input. If you want the user to input his or her name, you must use a string.

Using cin>> to input a string works, but it will terminate the string after it reads the first space. The best way to handle this situation is to use the function cin.getline. Technically cin is a class, and you are calling one of its member functions. The most important thing is to understand how to use the function however.

The prototype for that function is:
cin.getline(char *buffer, int length, char terminal_char);

The char *buffer is a pointer to the first element of the character array, so that it can actually be used to access the array. The int length is simply how long the string to be input can be at its maximum (how big the array is). The char terminal_char means that the string will terminate if the user inputs whatever that character is. Keep in mind that it will discard whatever the terminal character is.

It is possible to make a function call of cin.getline(arry, '\n'); without the length, or vice versa, cin.getline(arry, 50); without the terminal character. Note that \n is the way of actually telling the compiler you mean a new line, i.e. someone hitting the enter key.

For a example:

```
#include <iostream.h>
int main()
{
  char string[256];   //A nice long string
  cout<<"Please enter a long string: ";
  cin.getline(string, 256, '\n'); //The user input goes into string
  cout<<"Your long string was:"<<endl<<string;
  return 0;
}
```

Remember that you are actually passing the address of the array when you pass string because arrays do not require a reference operator (&) to be used to pass their address. Other than that, you could make \n any character you want (make sure to enclose it with single quotes to inform the compiler of its character status) to have the getline terminate on that character.

String.h is a header file that contains many functions for manipulating strings. One of these is the string comparison function.

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

strcmp will accept two strings. It will return an integer. This integer will either be:
Negative if s1 is less than s2.
Zero if s1 and s2 are equal.
Positive if s1 is greater than s2.

Strcmp is case sensitive. Strcmp also passes the address of the character array to the function to allow it to be accessed.

int strcmpi(const char *s1, const char *s2);

strcmp will accept two strings. It will return an integer. This integer will either be:
Negative if s1 is less than s2.
Zero if the s1 and s2 are equal.
Positive if s1 is greater than s2.

Strcmpi is not case sensitive, if the words are capitalized it does not matter.Not ANSI C++

char *strcat(char *desc, char *src);

strcat is short for string concatenate, which means to add to the end, or append. It adds the second string to the first string. It returns a pointer to the concatenated string.

char *strupr(char *s);

strupr converts a string to uppercase. It also returns a string, which will all be in uppercase. The input string, if it is an array and not a static string, will also all be uppercase. Not ANSI C++

char *strlwr(char *s);

strlwr converts a string to lowercase. It also returns a string, which will all be in uppercase. The input string, if it is an array, will also all be uppercase.

size_t strlen(const char *s);

strlen will return the length of a string, minus the termating character(/0). The size_t is nothing to worry about. Just treat it as an integer, which it is.

Here is a small program using many of the previously described functions:

```
#include <iostream.h>   //For cout
#include <string.h>     //For many of the string functions
int main()
{
  char name[50];            //Declare variables
  char lastname[50];        //This could have been declared on the last line...
  cout<<"Please enter your name: ";   //Tell the user what to do
  cin.getline(name, 50, '\n');       //Use gets to input strings with spaces or
//just to get strings after the user presses enter
  if(!strcmpi("Alexander", name))  //The ! means not, strcmpi returns 0 for
  {                               //equal strings
    cout<<"That's my name too."<<endl; //Tell the user if its my name
  }
  else                            //else is used to keep it from always
  {                                                    //outputting this line
     cout<<"That's not my name.";
  }
   cout<<"What is your name in uppercase..."<<endl;
  strupr(name);                   //strupr converts the string to uppercase
  cout<<name<<endl;
  cout<<"And, your name in lowercase..."<<endl;
  strlwr(name);                   //strlwr converts the string to lowercase
  cout<<name<<endl;
  cout<<"Your name is "<<strlen(name)<<" letters long"<<endl;  //strlen returns
//the length of the string
  cout<<"Enter your last name:";
  cin.getline(lastname, 50, '\n'); //lastname is also a string
  strcat(name, " ");                              //We want to space the two names apart
  strcat(name, lastname);      //Now we put them together, we a space in
//the middle
  cout<<"Your full name is "<<name; //Outputting it all...
  return 0;
}
```