Exercises – STL Strings

The standard template library (STL) provides a comprehensive selection of containers for use in C++ projects. We will go into depth on a number of the most important containers later but for now, to get you used to using the STL, we are just going to focus on strings. The string class provides a substantial increase in functionality over the old char* arrays which you have previously used. For example it allows us to create strings, copy strings, concatenate strings using the + operator, compare strings using == and other useful things. Generally most people use it in preference to Char arrays.

To use the STL strings in your project you need to #include<string>

All the STL containers are inside the std namespace so it's easiest to use that name space:

```
#include <string>
using namespace std;
```

Creating a string is easy:

```
string myString1
```

The following code demonstrates some of the features of the string class:

```
string myString1 = "Hello World";  //demonstrate creating a string
string myString2 = myString1; // demonstrate copying a string

if (myString1 == myString2) // demonstrate equality test
{
	cout << "strings are the same" << endl;
}

myString1 = "Goodbye!"; //changes contents of string 1

cout <<"myString2 = " << myString2<<endl; //demonstrates that strings are now different

cout <<"myString1="<<myString1<<endl;//we didn't just copy the pointer as with the old char[]

if (myString1 != myString2) //demonstrate inequality test
{
cout << "strings are not the same" << endl;
}

myString2 += "... " + myString1; //demonstrate using += for concatenation
cout << "myString2 = " << myString2 << endl;
```

As can be seen the class is quite sophisticated and quite intuitive compared to char[]. In particular the =, ==, !=, and += operators all work as one would expect, because they have been overloaded in the class, compared to char[] where they are work on the pointers to the array.

1. Start a new project and test the code above. Make sure you understand it. Add more strings and experiment with the functionality that the class provides. Refer to the MSDN documentation for a list of the functions and how they work.

2. You will write a simple text adventure.

   The main purpose of this exercise is to allow you to familiarize yourself with the string handling functions which are part of the string class. You should find that it is much easier to write code which manipulates strings with this class than it was with the old char* format.

   You should already have your completed text adventure from your assignment, so for this exercise you can either create a new game as described below or update your assignment to use the std::string class.

   The game is turn based with each turn consisting of the player typing an input and the game state updating accordingly. The game should have the minimum of these inputs:

   North, South, East, West, Take<Item Name>, Drop <Item Name>, Examine Room, Examine <Item Name>, Use <Item Name>, Inventory (displays what the player is carrying)

   1. Create a suitable structure for rooms. This should have a name for the room and a description. Each room has four exits which may connect to four other rooms. Store the rooms in an array (you can use a dynamic array if you know how to do it or just static for the time being).
   2. Implement movement: You need to parse the player input to detect movement commands (North, South, East, West).
   3. Whenever a command is successfully parsed display a suitable confirmation messages, in this case a description of the new room.
   4. Next add a list of objects and display them when players enter a room that contains them
   5. Implement the take and drop commands
   6. Implement the examine commands
   7. Implement a simple puzzle to unlock a door using a key.
   8. Feel free to add as much functionality as you have time for!