LAB MANUAL 09

STRINGS

Lab Objectives:

At the end of this lab students will know about

- What is the purpose of Strings?
- How to use Strings in C++ programs

Strings

By now, you have learned how to declare a char variable to hold characters. ; You might also have learned that a string (like a person's first or last name) in C++ can simply be represented by a null terminated sequence of characters stored in a char array. These days you can use the string type for strings. In the bad old days, before circa 1994, you had to use character arrays, because there was no native string type in C++ most implementations. There is still no string type in C. A string, represented as a character array, is called a "C string". Here's a quick little review of declaring and initializing C strings.

To declare a character array:

char s1[8];

To assign a value to an array when you declare it:

char s1[8] = "one";

Actually, if you are assigning a value at the same time as you are declaring the char array, you don't need to specify an array size.

char s1[] = "one";

Assigning a value either way results in a character array of finite length, established at compile time.

Note: You cannot assign the value of a C string after the declaration as you do with other simple variable assignments.

i.e. s1 = "one"; <- WRONG

To assign a value after the declaration, you need to use a C String function such as strcpy. We'll look at that next.

C++ supports a wide range of C string manipulation functions. Following are some of them:

1.1 strcpy

Use this function if you would like to copy one string to another string.

```
#include <string.h>
#include  <iostream>
using namespace std;
int main()
{
char Original[30] = "I am the original!";
 char Copy[30] = "I am the copy!";
cout << "Before strcpy(): " << endl;
cout << "Original = " << Original << endl;
cout << "Copy = " << Copy << endl;
strcpy(Copy, Original);
cout << "After strcpy(): " << endl;
cout << "Original = " << Original << endl;
cout << "Copy = " << Copy << endl;
return 0;
}
```

Here is the running result of the above program:

Before strcpy():

```
Original = I am the original!
Copy = I am the copy!
After strcpy():
Original = I am the original!
Copy = I am the original!
```

1.2 strlen:

Use this function if you would like to know the length of a string.

```
#include <string.h>
#include <iostream>
using namespace std;
int main()
{
char String1[30]= "I am the string!";
cout << "Here is string:" << String << endl;
cout << "Its length is: " << strlen(String) << " characters\n";
return 0;
}
```

Here is the running result of the above program:

```
Here is string:I am the string!
Its length is: 16 characters
```

String Class:

C String is one of the two ways that you can manipulate strings in C++. The other way is through the string class. This allows strings to be represented as objects in C++. The string class gives you strings of unbounded length because the string space is allocated at run-time. Compare this to C Strings where the finite string length of char array variables is defined at compile time.

Moreover, assignment, comparison, and concatenation of strings using the string class is arguably easier than using C Strings. However, you'll still see C String code around in older programs so it's best to know both. Why, you may ask, do we need the string class when we already have C string? Here is why: null terminated strings (C strings) cannot be manipulated by any of the standard C++ operators. Nor can they take part in normal C++ expressions. For example, consider this fragment:

```
char s1[80], s2[80], s3[80];
s1 = "one"; // error
s2 = "two"; // error
```

```
s3 = "three"; //error
```

As the comments show, in C++, it not possible to use the assignment operator to give a character array a new value. To do this, you need to use the strcpy function that was discussed above:

```
strcpy(s1, "one");

strcpy(s2, "two");

strcpy(s3, "three");
```

There are also other reasons for using strings. For example, the null-terminated C String does not check if the array is out of bound and that contributes to many string problems encountered by C++ programmers, experienced and inexperienced alike. Following is an example showing you how to use the string class to manipulate strings. The example gives you a feeling of how string objects work, which means you do not need to remember how it really works.

```
// Purpose: Demonstrates the use of the string class

#include <iostream>

#include <string>

using namespace std;

int main()

{

string str1("This is a test");

string str2("ABCDEFG");

cout << "Initial strings: \n";

cout << "str1: " << str1 << endl;

cout << "str2: " << str2 << "\n\n";

// demonstrate length()

cout << "Length of str1 is: " << str1.length() << endl;

return 0;

}
```

Here is how it runs:

Initial strings:

str1: This is a test

str2: ABCDEFG

Length of str1 is: 14

Lab Tasks

Question # 1

Write a C++ code to count the number of a's in a string

Question # 2

Write a program in C++ to read a sentence and replace lowercase characters by uppercase and vice-versa.

Question # 3

Write a C++ program to remove characters in String except Alphabets.

Question # 4

Write a C++ program to concatenate one string after the other without using any library function.

Question # 5

Write a C++ program which stores names of five cities and print the names of only those cities which start from K.