

Lab 3.1 String Processing in C/C++ using arrays of characters

In the original C language, strings were processed by using arrays of characters. For example, we might declare a character array like this:

```
char lastName[25];
```

which is an array of characters which can be used to hold a persons last name (for last names of up to 25 characters in length.

Since string processing is such a common activity, certain conventions evolved and were added to the C language in order to use arrays of characters to hold and process strings. For example:

1. There is a special initialization form that may be used with character arrays to give a string an initial value:

```
char lastName[] = "Smith";
```

2. By convention, all c-style character arrays that hold strings will have a (hidden) null character ('\0') as the last character in the array. So for example, the lastName character array actually contains the characters:

```
Index:    0    1    2    3    4    5
Characters: 'S', 'm', 'i', 't', 'h', '\0'
```

3. A library of functions was created to process c-style character arrays. It may be accessed by:

```
#include <cstring>
...
char lastName[] = "Smith";
if (strcmp(lastName, "Jones") == 0)
    cout << "The name is Smith";
int l = strlen(lastName);
```

4. Additionally, the C++ stream libraries were designed to allow special input (and output) to c-style character arrays, for example:

```
cin >> lastName;
cout << "The name you entered was"
    << lastName << endl;
```

Objectives

In this lab, you will practice using standard character arrays for doing string processing in C. You will become familiar with the cstring standard library

functions `strlen()` and `strcmp()`, and you will implement your own version of the `strlen` function.

Instructions

1. Create a new program called `Lab3-1.cpp`. As a simple first step, declare an array of characters that can hold up to a maximum of 50 characters. Then ask the user to input a string, and store it in your character array. Use the standard stream I/O objects `cin` and `cout` to input the string from the user and display the string that the user entered.
2. Declare a second array of characters, and have the user input another string. Then, use the `strcmp()` function to determine if the two strings the user input are the same or are different. The `strcmp()` function takes 2 character arrays as parameters, and it returns 0 if the strings are equal, and it returns a non-zero value if the strings are not equal. If the strings are equal, display a message on the terminal declaring that “The 2 strings you entered were the same string!”. If they are not equal, display the message “The 2 strings you entered were different. Test your program a couple of times with equal and non-equal strings.
3. Use the `strlen()` function to find out the number of characters in the 2 strings entered by the user. Display the string and the number of characters in the string for each input to the terminal.
4. Write your own version of the `strlen()` function, called `myStrlen()`. `myStrlen()` should take a single character array as a parameter, and return a single integer as its result. You need to write a loop in order to count the number of characters in the character array that is passed into your `myStrlen()` function. How will you do this? Remember, the last character of your character array strings will be the null character, `'\0'`. You need to loop through the character array until you find the null character. Once you find the index of the null character, you can figure out the length of the string that should be returned. For example, in the character array holding the string “Smith”, the null character was at index 5. The string “Smith” has 5 characters, so the index where the null terminator occurs is actually the same as the length of the passed in string.

Lab 3.2 String Processing in C/C++ using the string data type

The C++ language added a new mechanism (somewhat confusingly) for processing strings, the string data type. The string data type is actually an example of an object. We will be studying the creation of objects and classes later on in this class. To use the string data type, you need to include the string library (different from the cstring library for character arrays):

```
#include <string>
```

This allows you to create variables using the newly defined string data type:

```
string lastName;
```

Notice that lastName here is of type 'string', it is not a character array as we used in the example of part 3.1. You can use cin and cout stream output to input and output string data types, just like we can for character arrays:

```
cin >> lastName;  
cout << "The namer you entered was: " << lastName << endl;
```

The string data type also supports all of the functions of the cstring library, like strlen() and strcmp(), but using different syntax. For example, to get the length of a string data type:

```
cout << "The length of the string is: " << lastName.length();
```

One nice things about strings is that they overload and support many of the arithmetic operators to manipulate string expressions. For example:

```
string firstName = "Agent";  
string lastName = "Smith";
```

```
string fullName;
```

```
fullName = firstName + " " + lastName; // + can be used to do string  
concatenation
```

```
if (firstName == lastName) // == can be used to compare strings  
    cout << "The first name is the same as the last name";
```

Instructions

1. Create a new program called Lab3-2.cpp. Do the steps 1-3 from part 3.1 of the lab, but using the string data type and appropriate syntax (e.g. compare if strings are equal using ==, find their size using the .length() syntax, etc.
2. For extra practice, read and try using some of the other string functions, such as the substr() and find() methods.

Lab 3 Finished

You have now completed Lab 3. If your program compiles and runs correctly and you have successfully uploaded your source file to the eCollege online submission site, then you are done. At this point, you might want to reread the sections in chapter 8 on the string data type, and in section 9 on using character arrays for string processing if any of the concepts are still fuzzy to you.