CSC 2430 – Data Structures 1

Lab #4 – "String Manipulation Package & 'Last, First' Name Formatting"

Arrays and ASCIIZ-String Manipulations

Header and Implementation files. Multiple-source-file applications

Due: Wednesday, February 3, 2016

For this lab, you must create a Visual C++ project that contains 3 separate source-code files:

stringPkg.h	a .h "header" file that contains function prototype declarations
stringPkg.cpp	a .cpp source file that contains implementations of the functions specified in stringPkg.h.
main.cpp	a .cpp source file that contains a main( ) program to solve the LastCommaFirst application problem specified below.

This lab involves the implementation of a small collection of functions that manipulate standard arrays of characters that represent character strings using the ASCIIZ format (that is, string-values with a null character (e.g., 0, or '\0') appended to the end as a sentinel). The function prototypes for this string library are specified in file "stringPkg.h", shown below on the last page of this assignment handout.

To begin, create a new, empty Visual C++ project for a Console Application. Then create a new C++ Header file in the project (menu Project/Add New Item), named "stringPkg.h". Download the "stringPkg.h" header file from the assignment on Blackboard, copy this file into your project directory, and use the menu Project / Add-Existing-file to incorporate it into your project workspace.

Now, create another new C++ Source file in the project, named "stringPkg.cpp". In this source file, you must provide an implementation of each of the functions that are specified in "stringPkg.h". Therefore, the code layout for "stringPkg.cpp" will look similar to:

```
// stringPkg.cpp
// Implementation of the stringPkg library for CSC 2430
// Written by: …… your name ……
// …… your standard source code heading comments ……

#include  "stringPkg.h"

int stringLength(const char s[])
{
    . . . . . your implementation code . . . . .
}

void stringCopy(char dest[], const int destBuffSize, const char src[])
{
    . . . . . your implementation code . . . . .
}

. . . etc/etc/etc . . . for all the functions
```

Notice the

#include "stringPkg.h"

statement, which imports all the function prototype declarations. This statement must be specified by any source code file in the application that wants to work with or utilize any of the library functions in the package. The quote " … " marks indicate that the included file "stringPkg.h" is a file that is local to this project. < > brackets (such as <iostream>) indicate that the file is part of standard C++.

In "stringPkg.cpp", design and code an implementation of each of the functions that are declared in "stringPkg.h". The purpose of this lab is to give you experience in working with ASCIIZ strings and manipulating their values as arrays of characters. Therefore

For this lab, you are <u>not</u> allowed to use the built-in 'string' datatype or <u>any</u> of the built-in "str…( )" functions from the <cstring> library, such as strlen, strcpy, strcat, strcmp, etc.

As an example of an implementation of a function, stringLength( ) could be implemented as:

```
int stringLength(const char s[])
{
    int len = 0;

    while(s[len] != 0)  ++len;   // count chars until encounter null char
    // or while(s[len]) ++len;

    return(len);
}
```

As another example, stringCopy( ) could be implemented in various ways:

```
void stringCopy(char dest[],const int destBuffSize, const char src[])
{
    int len = stringLength(src);

    if(len >= destBuffSize)
        len = destBuffSize-1; // truncate if necessary, leave room for null

    for(int i = 0; i < len; ++i)     // count through the string-value
    {
        dest[i] = src[i];     // copy each character
    }
    dest[len] = 0;            // append the ASCIIZ null at end
}
```

Notice that it is OK in your implementation to utilize other functions from your stringPkg, as needed.

An alternative implementation that checks the length constraint as the loop proceeds might be:

```
void stringCopy(char dest[], const int destBuffSize, const char src[])
{
    int i = 0;

    // look for null sentinel as long as within dest buffer size
    while( (src[i] != 0) && (i < destBuffSize-1) )
    {
        dest[i] = src[i];     // copy each character
        ++i;                  // move to next position
    }
    dest[i] = '\0';           // Another form for ASCIIZ null
}
```

Notice that none of the functions in your stringPkg library perform any input or output operations. For this lab, all I/O must be done in the main Application program (see below).

Finally, create another new C++ Source file in the project for the actual Application, named something like "main.cpp" or "LastNameFirst.cpp" or "lab4.cpp". In an application source file, you can write the code to test out your implementation of the stringPkg library, or to implement some other application program that utilizes the stringPkg library. Normally, at the start of this application file (after your standard source-file code heading comments), you would be sure to include the header file for the stringPkg library along with any other necessary includes. Usually it is better to include the standard headers first, before including header(s) for your own libraries, for example:

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

#include  "stringPkg.h"
```

To summarize, this lab's Visual Studio C++ project 'Lab4' should have one Header .h file (e.g., stringPkg.h) and two Source .cpp files (e.g., stringPkg.cpp and LastNameFirst.cpp). When you instruct Visual C++ to "build the application", it will compile all the specified source files and then link them all together into a single application (.exe) executable program (usually located in the Debug directory for the project). Even though the source code is specified in separate files, the project will build a single .exe executable program using the project name (e.g., named 'Lab4.exe').

The LastNameFirst Application

Write an application program that obtains a cstring value from the user (e.g., using cin.getline( )) representing a name in the form

First Last

converts it and stores it in another cstring variable in the form

Last, First

and finally prints out a message demonstrating that the conversion operation worked.
In addition, the program must print out another message indicating which of the two cstring forms is alphabetically less than the other.

For example, suppose the user enters a person's name as a cstring value on a single input line:

Mike Tindall

The output should be

Thank you Tindall, Mike for entering Mike Tindall.
The First Name is less than the Last Name.

The original string entered by the user separates the names using one or more blank (e.g., whitespace) characters between the first name and the last name. Your program must convert this original cstring into a separate cstring variable that has the names in the reverse order and has a single comma followed by a single blank character separating the last and the first names. Your program must use your **stringPkg** functions to manipulate the cstring-values, including copying and concatenating values, retrieving or searching for specific characters (like the blanks, etc.), comparing values, etc.
Again, your program must not use the built-in 'string' datatype or any of the <cstring> library functions. It must use cstrings and functions implemented in your stringPkg.

To Hand In:

1. Printed source code program listings for all (three) files in your project.
2. Printed sample execution(s) demonstrating correct operation of your program.
   Run this program three times (at least), once using your instructor's name "Mike Tindall" as the input string, once using "Fred Flintstone", and once using your own name.

```
// stringPkg.h
// A library of ASCIIZ string functions.
// All functions in this library manipulate character arrays
// whose values are sequences of characters (string-values)
// utilizing the ASCIIZ format.

// ------------------------------------------------------------
// Return length of a string, length >= 0.
// Length is a count of the characters in the string-value,
//    not including the null at the end.
// An empty string ("") has length 0.
int stringLength(const char s[]);

// ------------------------------------------------------------
// Copy string src to string dest.
void stringCopy(char dest[], const int destBuffSize, const char src[]);

// ------------------------------------------------------------
// Concatenate string t to end of string dest
void stringConcatenate(char dest[], const int destBuffSize, const char t[]);

// ------------------------------------------------------------
// Retrieve character from string s[position].
// Position must be between 0 and (stringLength-1).
// Return 0 if position is out of range.
char stringGetchar(const char s[], const int position);

// ------------------------------------------------------------
// Find ch in string s and return position: 0 - (stringLength-1)
// Return -1 if ch not found in s.
int  stringFindchar(const char s[], const char ch);

// ------------------------------------------------------------
// Set resultString[] to a string value that is a copy of
//    a specified substring of original string s.
// If specified start position is not located within the string s,
//    then set resultString to the empty string ("").
// If specified len < 0, or if (start + len) > the length of s
//    then set resultString to the longest possible substring.
void stringSubstring(
       char  resultString[],      // new string buffer
       const int resultBuffSize,  //  result array buffer size
       const char s[],            // the original string
       const int start,           // starting position of substring within s
       const int len = -1);       // length of substring within s
                                  //   len<0: longest possible substring
// ------------------------------------------------------------
// Alphabetically compare string s to string t, based on ASCII charset.
// Return an integer value < 0 if s <  t
// Return an integer value 0   if s == t
// Return an integer value > 0 if s >  t
int  stringCompare(const char s[], const char t[]);
// ------------------------------------------------------------
```