Nicholas Arnaud
9/21/2016
NA1163
Intro to Programming

I. Description of problem

String Class Assessment

Problem: You are to create a basic string utility class that will make working with character arrays easier to manage. Your String class should contain the functions listed below. You need to include functions that implement the following features in your String class:

-What this program needed to achieve was to have any string placed into a private class and then modified and/ or changed in six different ways without interfering with any other function and then returned while still having the program's code organized and easy to look through for refinements and clarity.

-A .h and .cpp file submitted that implements the defined String class as per the specifications in Part 1 of the Assessment Description above, following industry-standard coding techniques

-String class code is properly commented to describe the functionality and use of the class

-Submitted code is free from faults and errors

II. Input info

- Command console runs a test of the program first and then allows the user to type his or her own two strings.

III. Output info

-Outputted info displays results and stats of the string that has been declared inside the parameters of "FunString Funstring()"

-Total length of the first declared string

-Showing the length of the first indexed character of the first string

-Displaying whether the first and second string are similar

-Reformatted string into a const char and redisplayed as so

-Seeks a smaller string that could be found with the original first string

-Reformatted string as a C- style string

IV. User Interface

-*Not Applicable

V. System Architecture

Name: FunString()
Description: basic tool to call functions and is the default constructor

Name: FunString()
Argument:(First String, Second String) { m_String = String, m_String2 = sString;}
Description: Will save the strings to private to prevent permanent change

Name: int Length()
Description: Will find the length of the string completely until reaches null

Name: int indexedChar()
Argument: (int j)
Description: Will pick a character in a string

Name: bool Compared()
Argument: (FunString as)
Description: Will compare 2 strings together

Name: char* Append ()
Argument:(FunString as)
Description: Will add the second string after the first string

Name: char* Prepend()
Argument:(FunString as)
Description: Will add the second string before the first string

Name:const char* c_Style()
Description: Will write the string c-styled

Name: char* c_Style()
Arguement:
Description: Will write the string c-styled

Name: int subStrLoc()
Description: Will find a SubString within the first string

Name:int stratString()
Argument: (int k)
Description: Will find the substring within the class after a certain index

Name: void strRepStr
Description:Will replace a substring with another substring

Name:char* c_enterStyle()
Description: Will set string to input C-style string

VI. Implementation Documents

VI. 1 FunString.h

Include source code

```
#pragma once

class FunString
{


public:
	FunString(); //basic tool to call functions
	FunString(char String[]);
	int Length(); //Will find the length of the string completely until reaches null
	char indexedChar(int j);//Will pick a character in a string
	bool Compared(FunString as); //Will compare 2 strings together 0
	char* Append(FunString as); //Will add the second string after the first string
	char* Prepend(FunString as); //Will add the second string before the first string
	const char* c_Style(); // Will write the string c-styled
	void upperCase(); //Will change string characters into uppercase characters
	void lowerCase(); //Will change string characters into lowercase characters
	int subStrLoc(); //Will find a SubString within the first string
	int stratString(int k); //Will find the substring within the class after a certain index
	void strRepStr(); //Will replace a substring with another substring
	char* c_enterStyle(); //Will set string to input C-style string

	char m_String[255]; //string one
	int Strlen; //first string's length
};
```

VI.2 Source.cpp

```
#include <iostream>
#include "FunString.h"


int main()
{
	// fs.initialize(); setup for input strings
	//	fs.Length(); //Is part 1 of Assessment
	//	fs.indexedChar(); //Is part 2 of Assessment
	//	fs.Compared(); //Is part 3 of Assessment
	//	fs.Append(); //Is part 4 of Assessment
	//	fs.Prepend(); //Is part 5 of Assessment
	//	fs.c_Style(); //Is part 6 of Assessment
	//	fs.lowerCase(); //Is part 7 of Assessment
	//	fs.upperCase(); //Is part 8 of Assessment
	//	fs.subStrLoc(); //Is part 9 of Assessment
	//	fs.stratString(); //Is part 10 of Assessment
	//	fs.strRepStr(); //Is part 11 of Assessment
	//	fs.c_enterStyle(); //Is part 12 of Assessment


	FunString fs = FunString("TestStatement1"); // tested first string
	FunString as = FunString("TestStatement2"); // tested second string


	for (int i = 0; i < 2; ) //for loop will run the preset strings first and on second run will use
the users' inputted strings
	{
		system("cls"); //starts a fresh blank screen
		std::cout << "The first string is: " << fs.m_String << "\n" << "The second string is:
" << as.m_String << "\n \n"; // reads given strings

		std::cout << "Number of characters in the string is: " << fs.Length() << " \n \n"; //
displays the number of characters in the string "m_String"

		std::cout << "Indexed character in the string is: " << fs.indexedChar(3) << "\n \n";
//Displays a character in String at a given index
```

```
                (fs.Compared(as) == 1) ? std::cout << "The two strings are very much alike \n \n"
: std::cout << "The two strings are not similar at all \n \n"; // outputs the result of whether or not
the strings are similar

                std::cout << "The second string added added to the first string is: " <<
fs.Append(as) << "\n \n"; //runs the function "Append" to add a second string after the first string
and outputs the result

                std::cout << "The second string placed before the first string is: " <<
fs.Prepend(as) << "\n \n"; // runs the function "Prepend" to add a second string before the first
string and outputs the result

                std::cout << "The String in c-style: " << fs.c_Style() << "\n \n"; //runs the function
"c_Style" to make the string in a c-style string

                fs.lowerCase();         //runs the lowercase function
                std::cout << "The String in all lowercase characters is: " << fs.m_String << "\n \n";
// displays all characters in the string lowercase

                fs.upperCase(); // runs the uppercase function
                std::cout << "The String in all uppercase characters is: " << fs.m_String << "\n
\n"; // displays all characters in the string uppercase

                std::cout << "With '0' being false and '1' being found, the substring is: " <<
fs.subStrLoc() << "\n \n"; // tells whether or not the substring in function "subStrLoc" at any point
in the string

                std::cout << "With '0' being false and '1' being found, the substring is:  \n" <<
fs.stratString(4) << "\n \n"; // tells whether or not the substring in function "stratString" starting
from a certain point in the string

                i++; // increments the for loop to '2' for the if statement to work


                if (i != 2) // if loop will make sure the program only asks for user input once while
the loop only runs twice
                {
                        std::cout << "Please enter first string: \n";
                        fs.c_enterStyle(); //runs function that has the user input his very own
string

                        std::cout << "Now please enter second string: \n"; // asks user
                        as.c_enterStyle(); //runs function that has the user input yet another string
                }
        }
        system("pause");
        return 1;
}
```

VI.3 FunString.cpp

```
#include <iostream>
#include "FunString.h"
#include <iostream>
#include "FunString.h"

FunString::FunString()
{
	// the default function constructor
};

FunString::FunString(char string[])
{

	int i;
	for (i = 0; string[i] != '\0'; i++) // for loop reads all the characters in the given string
arguments
	{
		m_String[i] = string[i]; // sets all the characters into the string class
	}
	m_String[i] = '\0'; // adds the null character to the end of the string to prevent errors
}

int FunString::Length()
{
	int i = 0;
	for (; m_String[i] != 0; i++);// loop runs until "i" reaches the null character of the string
m_String

	Strlen = i; //saves the length of the length of first string
	return Strlen; //returns number of characters in string
}

char FunString::indexedChar(int j) // finds a character at a certain index that was inputted in
main
```

```
{
        char indChar = m_String[j]; // sets a char variable as the indexed char inside the string
        return indChar; // returns the character in the string
}

bool FunString::Compared(FunString as) //compares two strings
{

        if (m_String == as.m_String)  // if statement used to find a difference between 2 inputted
strings
        {
                return true; // bool function becomes true and breaks from function
        }
        else  // else segment runs if comparison between the 2 strings are different in any way
        {
                return false;     // bool function becomes false and function ends
        }
}

char* FunString::Append(FunString as) //adds the second string to the first
{
        as.Length(); // runs the function Length to find the the length of the first string also
allowing Strlen to be used without error
        int m_Length = Strlen; // stores the first string's length as an integer
        int i; // creates a variable to be used in for loop
        for (i = 0; i < as.Strlen; i++) // for loop that repeats until the total length of the second
string is counted
        {
                m_String[m_Length + i] = as.m_String[i]; // adds more space for characters and
adds another character into the string
        }
        m_String[m_Length + i] = '\0'; // adds the final null character to the end of the string to
prevent errors
        return m_String; // returns the new string
}

char* FunString::Prepend(FunString as)
{
        as.Length(); // runs the function Length to find the total length of the second string also
allowing Strlen to be used without error
        int m_Length = as.Strlen;       // stores the second strings length as an integer
        int i; // creates a variable to be used in for loop
        for (i = 0; i < Strlen; i++) // for loop that repeats until the total length of the first string is
counted
        {
                as.m_String[m_Length + i] = m_String[i]; // adds more space for characters and
adds another character into the string
        }
        as.m_String[m_Length + i] = '\0'; // adds the final null character to the end of the string to
prevent errors
        return as.m_String; // returns the new string
}

const char* FunString::c_Style()
{
        const char* constString = m_String;   // creates a const character pointer towards the
original string
        return constString; // returns the now c-style string
}

void FunString::lowerCase()
{

        for (int j = 0; m_String[j]; j++) //loops through all the characters in the string
        {
                if (m_String[j] >= 65 && m_String[j] <= 90) // checks for any uppercase characters
                {
                        char c = m_String[j]; // sets new character and defines it as the uppercase
character needed to be changed
                        c += 32;          // adds 32 to change the value of character to its lowercase
equivalent in the ASCII table
                        m_String[j] = c; // sets the new lower case character into where the
uppercase character was in the string
                }
                else if ((int)m_String[j] >= 97 && (int)m_String[j] <= 122) // if the character is
already lowercase it goes here
                {
                        // the character is left alone and loops over looking at the next character
                        m_String[j];
                }
        }
}

void FunString::upperCase()
{
	for (int j = 0; m_String[j]; j++) // loops through all the characters in the string
	{
		if (m_String[j] >= 97 && m_String[j] <= 122) // checks for any lowercase
characters
		{

			m_String[j] = (int)m_String[j] - 32; // subtracts 32 to change the value of
character to its uppercase equivalent in the ASCII table

		}
		else if (m_String[j] >= 65 && m_String[j] <= 90) //if the character is already
uppercase it goes here
		{
			// the character is left alone and loops over looking at the next character
			m_String[j];
		}
	}
}

int FunString::subStrLoc()
{
	int i, j, temp;	//used to get index given string and arrays
	char substr[20] = { "state" }; //sets the substring
	std::cout << "Looking for the substring : " << substr << "\n"; //shows what the substring is
to the user
	for (i = 0; m_String[i] != '\0'; i++) //for loop to search string for the substring
	{
		j = 0; //used to index the substring
		if (m_String[i] == substr[j]) //if indeed the string contains the substring
		{
			temp = i + 1; //saves at what index the substring is located
			while (m_String[i] == substr[j]) //continues as long as both strings don't
equal to the null character
			{
				i++;
				j++;
			}

			if (substr[j] == '\0') //if the substring is found in the string
			{
				return temp;
			}
}
                    else //if the substring is not found
                    {
                              i = temp;
                              temp = 0; //causes break loop
                    }
          }
     }

     if (temp == 0) //breaks the loop
          return temp;
}

int FunString::stratString(int k)
{
     int w, nope; //used to index trough given string and arrays
     char subs[20] = { "ent" }; // creates the substring
     for (; m_String[k] != '\0'; k++) //for loop to search the string to find the substring
     {
          w = 0; // defined to index substring
          if (m_String[k] == subs[w]) // runs if the string has the substring
          {
               nope = w + 1; //saves the index value where the substring was found
               while (m_String[k] == subs[w])//continues as long as both strings don't
equal to the null character
               {
                    k++;
                    w++;
               }
               if (subs[w] == '\0') //once the substring reaches the null character
               {
                    return nope; // breaks loop and substring was found before the
end of main string
               }
               else
               {
                    w = nope; // saves the end location of the substring
                    nope = 0; // the function returns false and substring is not found
               }
          }
     }
     return nope;
}
```

```
void FunString::strRepStr()
{
	//No understanding yet
	// is a bonus
}

char* FunString::c_enterStyle()
{
	char string[255]; // sets string in c-style
	std::cin >> string; //user changes the preset string
	std::cout << "\n \n"; // adding some space between lines
	int i;
	for (i = 0; string[i] != '\0'; i++) // for loop reads all the characters in the given string
arguments
	{
		m_String[i] = string[i]; // sets all the characters into the string class
	}
	m_String[i] = '\0'; // adds the null character to the end of the string to prevent errors

	return m_String;
}
```

VII. Read Me

VII.1 Controls

-The user only types in the first and second string

VII.2 How to obtain Application

-This program application can be located within my github page at "https://github.com/NicholasArnaud/String-Class-Assessment". Once reaching the page, you can just simply open the program by downloading the ".exe" file.

VII.3 How to use Application

-Application is used to output various information about one or two strings used inside the parameters of a function.

-To use and run the program. Make sure that there are 2 strings inside the first function in main file within parenthesis. Then simply press the “f5” key and the program should run.