Donray Williams

9/21/16

Create a String Class – Assessment 1 – Year 1

# 1.0 Requirements Documentation

# 1.1 Description of problem

A. Name: Create a String Class

B. Problem Statement: Create a string class that will make working with character arrays easier to manage.

C. Problem Specification: The class that is created should include functions that complete the following)

1. Query the string’s length, returning an integer.
2. Access a character at a certain index within the string class.
3. Compare if the string is the same as another string class.
4. Append one string to another.
5. Prepend one string to another.
6. Return the string as a basic constant C-style string (const char\*)
7. Convert the string to a duplicate containing all lowercase letters.
8. Convert the string to a duplicate containing all uppercase letters.
9. Find a sub-string within the string class.
10. Find a sub-string within the string class, starting from a certain index within the string.
11. Replace a sub-string found within the string with a different sub-string.
12. Set the string to an input C-style string.

1.2 Input Information

A. The user must enter two strings that are to be used within the functions.

1.3 Output Information

A. The console must display each result of the functions based on what the user entered.

1.4 User Interface

A. Not applicable.

# 2.0 System Architecture

2.1 Member Functions in the class.

* Prototype: getLength (No arguments)

Description: Returns the instance of m\_length that was set in MyString.

Precondition: None

Postcondition:

Visibility: Public

* Prototype: index (int input)

Description: Takes in the char m\_Data with paramater input then returns it.

Precondition: Must have an index

Postcondition:

Visibility: Public

* Prototype: compare (MyString str)

Description: Comapares each character that is inside of each string to check if they are equal.

Precondition: Must have two strings

Postcondition:

Visibility: Public

* Prototype: append (MyString str)

Description: Appends MyString to paramaters that are passed in Main.cpp.

Precondition: Must have two strings

Postcondition:

Visibility: Public

* Prototype: prepend (MyString str)

Description: Prepends "Front -> " which is passed in my Main.cpp on to the previous appended word.

Precondition: Must have two strings

Postcondition:

Visibility: Public

* Prototype: ToUpper (No arguments)

Description: Uppercases every character in the prepended word above.

Precondition: None

Postcondition:

Visibility: Public

* Prototype: ToLower (No arguments)

Description: Lowercases every character in the prepended word above.

Precondition: None

Postcondition:

Visibility: Public

* Prototype: findSubString (char \* sub)

Description: Using a bool, search for a certain character inside of the string if found return true, if not return false.

Precondition: Must be passed a char \*

Postcondition:

Visibility: Public

* Prototype: findSubStringIndex (int input ,char \* sub)

Description: Using a bool, goes to a pre set index and searches for a character if found return true, if not return false.

Precondition: Must be passed an input and char \*

Postcondition:

Visibility: Public

* Prototype: constCStyle (No arguments)

Description: Sets m\_Data to now equal a const char \* then return that variable that was set to the const char \*.

Precondition: None

Postcondition:

Visibility: Public

* Prototype: Replace (char \* mine, char \* sub)

Description: Searches the string for a preset character which is set in parameters in the Main.cpp then swaps in with another character that is also in the same parameter.

Precondition: Must be passed two char \*

Postcondition:

Visibility: Public

Member Variables:

**char** m\_Data [255] – A character array that stores the string.

**int** m\_length – Integer that stores the length of the string.

CPP File:

#include <iostream>

#include "MyString.h"

MyString::MyString(char \*F) //Constructor definition sets the valuue of the string

{

int i;

for (i = 0; F[i] != '\0'; ++i) //Loops array and sets values, stops when it reaches null ( End of word )

{

m\_Data[i] = F[i]; //F populates the array of m\_Data

//Ends loop once reaches null

}

m\_Data[i] = '\0'; //Sets end of array to null

m\_length = i;

}

// Problem 1

int MyString::getLength()

{

return m\_length;

}

//Problem 2

//Functions name: index

//Takes a single argument of type interger

//Takes in an interger which is set as "input"

char MyString::index(int input)

{

m\_Data[input];

return m\_Data[input];

}

//Problem 3

//Functions name: compare

//Takes a single argument of type MyString

//Comapares each character that is inside of the string to check if they are equal

bool MyString::compare(MyString str)

{

/\*int equal = strcmp(m\_first, m\_second);

if ((equal) == 0)

{

printf("They are the same \n \n");

}

else

{

printf("They are not the same \n \n");

}

if (strcmp(m\_first, m\_second) == 0)

{

printf("%s is equal %s \n \n", m\_first, m\_second);

}

else if (strcmp(m\_first, m\_second) == 1)

{

printf("%s is after %s \n \n", m\_first, m\_second);

}

else if (strcmp(m\_first, m\_second) == -1)

{

printf("%s is before %s \n \n", m\_first, m\_second);

}

return 1;\*/

// Above is an small example, created by Matthew Williamson

bool equal = true;

for (int i = 0;; i++) //Loops through the characters in the string and compares it to the second string

{

equal = (m\_Data[i] == str.m\_Data[i]) ? true : false; //If characters are equal then variable equal with type bool is set true

if (equal == false) //If equal comes out false it breaks out of the loop

{

break;

}

if (equal == true) //If equal comes out true it breaks out of the loop

{

break;

}

}

return equal; //Returns the equal out to console (true/false)

}

//Problem 4

//Functions name: append

//Takes a single argument of type MyString

//Appends the second string to first string

char \* MyString::append(MyString str)

{

int oldLength = this->m\_length; //Creates a variable called oldLength with type interger that is now equal to the length

int i;

for (i = 0; i < str.m\_length; ++i) //Loops through the second word entered until it hits the end of the word

{

m\_Data[oldLength + i] = str.m\_Data[i]; //Sets the second word to now append to the first

}

m\_Data[oldLength + i] = '\0'; //Ends loop once it hits end of string

m\_length = i + oldLength;

return m\_Data;

}

//Problem 5

//Functions name: prepend

//Takes a single argument of type MyString

//Prepends the second string to the appended string above

char \* MyString::prepend(MyString str)

{

char newString[255]; //The array of characters that will copy the current string

int i;

for (i = 0; i < m\_length; ++i) //Loops through the second word entered until it hits the end of the word

{

newString[i] = m\_Data[i];

}

newString[i] = '\0'; //Ends loop once it hits end of string

int oldLength = str.m\_length;

for (i = 0; i < getLength(); ++i) //Moves the current first string

{

m\_Data[oldLength + i] = newString[i];

}

newString[oldLength + i] = '\0'; //Ends loop once it hits end of string

for (i = 0; i < str.m\_length; ++i) //Places the second string in front of the first string

{

m\_Data[i] = str.m\_Data[i];

}

m\_Data[i + m\_length] = '\0'; //Ends loop once it hits end of string

m\_length += str.m\_length;

return m\_Data;

}

//Problem 6

//Functions name: constCStyle

//Does not take in any arguemnt

//Returns the string as a basic constant C-Style

const char \* MyString::constCStyle()

{

const char \* constCString = m\_Data;

return constCString;

}

//Problem 7

//Functions name: uppercase

//Does not take in any arguemnt

//makes all letters uppercase in the string

char \* MyString::ToUpper()

{

for (int i = 0; i < getLength(); ++i)

{

if ((int)m\_Data[i] > 96 && (int)m\_Data[i] < 123) //Uses the Ascii Table to check if each character is lowercase or not

{

m\_Data[i] = m\_Data[i] - 32; //If they are then change its integer value to Ascii Tables value for uppercase

}

}

return m\_Data;

}

//Problem 8

//Functions name: lowercase

//Does not take in any arguemnt

//Makes all letters lowercase in the string

char \* MyString::ToLower()

{

for (int i = 0; i < getLength(); ++i)

{

if ((int)m\_Data[i] > 64 && (int)m\_Data[i] < 91) //Uses the Ascii Table to check if each character is uppercase or not

{

m\_Data[i] = m\_Data[i] + 32; //If they are then change its integer value to Ascii Tables value for lowercase

}

}

return m\_Data;

}

//Problem 9

//Functions name: findSubString

//Does not take in any arguemnt

//Searches the string for characters

bool MyString::findSubString()

{

bool found = false;

const char \* sub = { "ee" }; //The charcters I set to be searched for

int x = 0;

for (int i = 0; i < getLength(); ++i)

{

if (m\_Data[i] == sub[x]) //Checks if the characters at index [i] is the same as the set characters above

{

x++;

if (x == 2) //Checks if x == 2 if it is the value of found is now true

{

found = true;

break;

}

}

else //Or if it is not the value is now 0 and found is now false

{

x = 0;

found = false;

}

}

return found;

}

bool MyString::findSubString(char \* sub) //x

{

MyString tmp = MyString(sub);

bool match = false;

for (int i = 0; i < this->m\_length; i++)

{

if (this->m\_Data[i] == sub[0])

{

for (int j = 0; j < tmp.getLength(); j++)

{

if (this->m\_Data[i + j] == tmp.m\_Data[j])

{

match = true;

}

else

{

match = false;

}

}

}

}

return match;

}

//Problem 10

//Functions name: findSubStringIndex

//Takes a single argument of type interger

//Searches the string for characters starting at a index

bool MyString::findSubStringIndex(int input, char \* sub)

{

MyString temp = MyString(sub);

bool foundIndex = false; //True or false to say if its starting at the certain index

int x = 0; //Variable x with type interger set to be a place hold for the index of the const char \*

for (int i = input; i < m\_length; ++i) //Makes a type interger = to my set input and loops to end of word length

{

if (m\_Data[i] == sub[x]) //Checks if the characters at index [i] is the same as the set characters above

{

x++; //Increments the x to go to next letter of string

if (x == temp.m\_length) //Checks if x == 2 if it is the value of found is now true

{

foundIndex = true; //Continues the loop

break;

}

}

else //Or if it is not the value is now 0 and found is now false

{

x = 0;

foundIndex = false; //Stops the loop

}

}

return foundIndex;

}

//Problem 11

//Functions name: replaceSubString

//Does not take in any arguemnt

//Searches the string for characters and if found said characters replaces them

void MyString::Replace()

{

bool foundReplace; //True or false to say if its starting at the certain index

const char \* subString = { "xx" }; //The characters I set to be searched for

const char \* replaceSubString = { "ll" }; //The characters I set to replace

int x = 0; //Variable x with type interger set to be a place hold for the index of both subString and replaceSubString

for (int i = 0; i < m\_length; ++i) //Loops that goes through length of word

{

if (m\_Data[i] == subString[x]) //Checks if the characters at index [i] is the same as the subString[i]

{

x++; //Increments the x to go to next letter of string

if (x == 2) //Checks if x == 2 if it is the value of found is now true

{

foundReplace = true; //Continues the loop

for (int i = 0; i < m\_length; ++i) //Loops that goes through length of word

{

if (m\_Data[i] == subString[x]) //Checks if the characters at index [i] is the same as the subString[i]

{

m\_Data[i] = replaceSubString[x]; //Replaces the subString with replaceSubString set string

x++; //Increments the x to go to next letter of string

}

else

{

x = 0;

}

}

break;

}

}

else

{

foundReplace = false; //Stops the loop

x = 0;

}

}

x = 0;

std::cout << "Replace SubString: ";

for (int i = 0; m\_Data[i] != '\0'; ++i) //States the new word

{

std::cout << m\_Data[i];

}

std::cout << "\n \n";

}

char \* MyString::Replace(char \* mine, char \* sub)

{

MyString m = MyString(mine);

MyString newString = MyString(sub);

MyString copy = MyString(m\_Data);

bool match = false;

int p = 0;

int follow = 0;

int copyIn = 0;

for (int i = 0; i < this->m\_length; i++)

{

copyIn = follow;

if (this->m\_Data[i] == sub[0])

{

for (int j = 0; j < m.getLength(); j++)

{

if (this->m\_Data[i + j] == m.m\_Data[j])

{

match = true;

}

else

{

match = false;

}

}

}

if (this->m\_Data[i] == m.m\_Data[p]) //Dont know why this works -> [0]

{

p++;

if (p == m.m\_length)

{

p = 0;

if (m.m\_length <= newString.m\_length)

{

m\_length += newString.m\_length - m.m\_length;

}

else

{

m\_length -= m.m\_length - newString.m\_length;

}

copyIn++;

for (int j = 0; j < newString.getLength(); j++)

{

m\_Data[(i + j) - (m.getLength() - 1)] = newString.m\_Data[j];

match = true;

}

if (i <= m\_length - newString.m\_length)

{

int holder = copyIn;

for (int x = i - m.m\_length + newString.m\_length + 1; x < m\_length; x++)

{

m\_Data[x] = copy.m\_Data[copyIn];

copyIn++;

}

copyIn = holder;

}

if (m.m\_length < newString.m\_length)

{

i += newString.m\_length - m.m\_length;

}

else if (m.m\_length > newString.m\_length)

{

i -= m.m\_length - newString.m\_length;

}

}

}

else if (this->m\_Data[i] != m.m\_Data[0])

{

{

match = false;

}

}

else

{

match = false;

}

m\_Data[m\_length] = '\0';

follow++;

}

return m\_Data;

}

**-Read Me -**

**You can access this file via this link:** [**https:github.com/wdonray/String-Class**](https://github.com/wdonray/String-Class)

**An executable to start the program and test it is also enclosed in the repository.**

**Once you run the program enter a string then press enter once, repeat this once more.**