Intro to Cpp

Devin Wiggins

9/21/16

1. **Requirement documentation**

I.1 **Description of problem**

Name: Create a String Class

Problem: Create a string class that makes working character arrays easier to manage

Problem description: string class with these functions:

The ability to query the string’s length, returning an integer

The ability to access a character at a certain index within the string class

The ability to compare if the string is the same as another string class

The ability to append one string to another

The ability to prepend one string to another

The ability to return the string as a basic constant C-style string (const char\*)

The ability to convert the string to a duplicate containing all lowercase letters

The ability to convert the string to a duplicate containing all uppercase letters

The ability to find a sub-string within the string class

The ability to find a sub-string within the string class, starting from a certain index within the string

The ability to replace a sub-string found within the string with a different sub-string

The ability to set the string to an input C-style string

I.2 **Input Description**

N/A

I.3 **Output** D**escriptio**n

The program manipulates the string in whichever way is specified and displays to console

I.4 **User interface**

Console

1. **Design Documentation**

II.2 **system architecture**

* **Class Name** (MyString)
* **In Private:**
* character pointer – m\_String
* **In public:**
  + (default constructor)
  + (constructor with the parameter of a pointer character named a that is assigned m\_word)
* **Name**: Size(int)
* **Parameters**: none
* **Description**: returns the size of a string excluding the null character
* **Name**: index(char)
* **Parameters**: index(int)
* Description: accesses a specific character within the index of a string
* **Name**: Compare(bool)
* **Parameters**: w(MyString)
* **Description**: checks whether two strings are the same or not
* **Name**: Append(char\*)
* **Parameters**: cat(Mystring)
* **Description**: Attaches the parameter’s string to the end of the object string, and returns the new object string
* **Name**: Prepend(char\*)
* **Parameters**: tac(MyString)
* **Description**: attaches the parameter’s string to the beginning of the object string, and returns the new object string
* **Name:** ToConstant(const char\*)
* Parameters: none
* Description: returns the object string as a constant
* **Name:** ToLower(void)
* **Parameters**: none
* **Description**: coverts the object string to lowercase
* **Name:** ToUpper(void)
* **Parameters**: none
* **Description**: converts the object string to uppercase

Name

* **Name**: FindSub(bool)
* **Parameters**: sub(MyString)
* **Description**: checks if argument string is within object string
* **Name**: FindIndexSub(char)
* **Parameters**: index(int), sub(MyString)
* **Description**: checks if sub string exists by index
* **Name**: SwapSub(Void)
* **Parameters**:
* **Description**:
* **Name**: ToC\_str(char)
* **Parameters**:
* **Description**:

1. **Implementation Documentation**

III.1 **Source Code**

**MyString.h**

#pragma once

class MyString

{

private:

char m\_word[255];

int size;

public:

//Todo

MyString() {};

MyString(char \*a);

//Constructors

int Size(); //queries the string's length returning an integer

char Index(int index); //accesses a character at a certain index

bool Compare(MyString w); //takes the sizes of each string and returns true or false

char\* Append(MyString cat); //takes the object string attaches the parameter string

char\* Prepend(MyString tac); //takes the parameter string in and attaches the object string

const char\* ToConstant(); // returns the object as a constant character

void ToLower(); //converts the object string to a lowercase version of itself

void ToUpper(); //converts the object string to a lowercase version of itself

bool findSub(MyString sub); //checks if argument string is within object string

bool findIndexSub(int Index, MyString sub); //checks if sub string exists by index

};

//unable to find a solution to incompleted functions. i could understand how they work

// i could not figure out how to optimize what ive learned to make those functions work

// i have also come to the conclusion that alot of the functions required a higher level \

// of knowledge to complete but because i spent so much time trying to figure it out

// on my own the time that i asked for help was too late. i try to

// do what i can without as much help in order to understand better.

MyString.cpp

#include <iostream>

#include <string>

#include "MyString.h"

MyString::MyString(char \*a)

{

int i;

int len = 0;

while (a[len] != '\0')

{

len++;

}

size = len;

for (i = 0; i < size; i++)

{

m\_word[i] = a[i];

}

}

int MyString::Size()

{

return size;

}

char MyString::Index(int index)

{

/\*as function is called the input index being searched will loop through

the string and return the index \*/

//char i;

if (index >= 0 && index < Size()) // if index(the variable) is greater than or

// equal to 0 and less the the size the index will be returned

return m\_word[index];

return '?'; // if the input is invalid the functiopn returns '?'

// no less than 0 no greater than size

};

bool MyString::Compare(MyString w)

{

int x = 0;

if (Size() != w.Size())

return false;

for (x = 0; x < Size(); x++)

{

if (m\_word[x] != w.m\_word[x])

return false;

}

//Check if words are the same length.

//Loop through all characters of both words until a character is different at

// that index.

//If no characters are different the words are the same.

return true;

}

char\* MyString::Append(MyString cat)

{

char \*arr = new char[cat.Size() + Size() - 4]; //allocates dynamic memory of strings

int it = 0;

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

{

arr[i] = m\_word[i];

} // sets arr pointer to word

for (int i = Size(); i < cat.Size() + Size(); i++)

{

arr[i] = cat.m\_word[it];

it++;

}

return arr;

}

char\* MyString::Prepend(MyString tac)

{

char \*arr = new char[tac.Size() + Size() - 4];

int it = 0;

for (int i = 0; i < tac.Size(); i++)

{

arr[i] = tac.m\_word[i];

}

for (int i = tac.Size(); i < Size() + tac.Size(); i++)

{

arr[i] = m\_word[it];

it++;

}

return arr;

}

const char\* MyString::ToConstant()

{

return m\_word;

}

void MyString::ToLower()

{

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

// i is supposed to go through the string checking each letter

{

if (m\_word[i] >= 'A' && m\_word[i] <= 'Z')

{

m\_word[i] = (int)m\_word[i] + 32;

}

else

break;

}

// this is supposed to return the value of

// the letter that i's address is equal to, plus 32.

// the 32 takes the ascii character's place value on the ascii table

// and adds it by 32 in order to match its lowercase counterpart

}

void MyString::ToUpper()

{

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

{

if (m\_word[i] >= 'a' && m\_word[i] <= 'z')

{

m\_word[i] = (int)m\_word[i] + 32;

}

else

break;

}

}

bool MyString::findSub(MyString sub)

{

int i = 0;

for (int it = 0; it < sub.Size() && i != ' '; it++)

{

for (i; i < Size(); i++)

{

if (m\_word[i] == ' ') // this probably is redundant but it basically skips check

// of space characters

{

continue;

}

if (m\_word[i] == sub.m\_word[it]) // if the characters match the nested for loop

// is broken and the outer for loop continues

{

break;

}

}

if (m\_word[i] != sub.m\_word[it] && m\_word[i] == '\0') // if the string has ended and the

// string was not found it returns

// false

{

return false;

}

}

return true;

}

bool MyString::findIndexSub(int Index, MyString sub)

{

int it = 0;

{

for (int i = Index; i < Size() && i != ' '; i++)// increments i to check continuous character spaces unless it

//hits a space character

{

if (m\_word[i] == sub.m\_word[it])

{

it++;

continue;

}

if (m\_word[i] != sub.m\_word[it] && sub.m\_word[it] != '\0') // if the letter do not match at any point it returns false

// null character check statement cannot be executed if it is false

{

return false;

}

if (sub.m\_word[it] == '\0')// after full check if the words are equal the loop breaks and returns true

{

break;

}

}

}

return true;

// this function will return true or false whether the argument string is the sub string specified by Index

}

Main.cpp

#include <iostream>

#include <string>

#include "MyString.h"

using namespace std;

int main()

{

int inputint;

MyString d = MyString("JAMIN MUSIC");

MyString s = MyString("programmin");

printf("I have two strings that i will be manipulating. my first string is");

printf("'JAMIN MUSIC' and my second string is 'programmin'");

printf("first I will reveal the sizes of each string");

cout << "the size of the first string 'JAMIN MUSIC' is "<< d.Size() << endl;

cout << "the size of the second string 'programmin' is " << s.Size() << endl;

printf("Type an index number of a letter that you would like to present from the string\n");

printf("'JAMIN MUSIC': ");

cin >> inputint;

cout << d.Index(inputint) << endl;

printf("the two strings will be compared. if they are different a 0 will print if they\nare");

printf(" the same 1 will print\n");

system("pause");

cout << d.Compare(s) << endl;

printf("'programmin' will be appended to 'JAMIN MUSIC'\n");

cout << d.Append(s) << endl;

printf("programmin will be prepended to 'JAMIN MUSIC'\n");

cout << d.Prepend(s) << endl;

printf("JAMIN MUSIC will be converted to lower case letters\n");

d.ToLower();

printf("programmin will be conveted to upper case letters\n");

s.ToUpper();

printf("if the string 'MUSUC' is in the 'JAMIN MUSIC' string a 1 will print else, 0 \n");

cout << d.findSub("MUSUC");

printf("type a number within the index range of the string 'JAMIN MUSIC' if the string 'MUSIC'");

printf("exists there 1 will print if not a 0 will print: ");

cin >> inputint;

cout << d.findIndexSub(inputint, "MUSIC");

system("cls");

printf("And thats my mystring class assessment");

system("pause");

return 0;

}

1. **READ ME**

In order to access this file, copy and paste the following link into browser:

<https://github.com/devinwiggins/String-class>

The page will display the files

* String class folder
* String class.docx
* String class.exe

You can either click clone or download, and download the zip file or you can go to each individual folder or file and download them separately.

The string class program will first ask for a number representing an index number for one of the strings after entering the number, you will have to press any key, then repeat.