**Reginald Reed**

**9/22/2016**

**String Class Assessment Documentation**

1. **Requirement Documentation**

**Name:**

String Class

**Description:**

Create a string class that works with character arrays

**Specifications:**

The ability to query the string 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

**2. Design Documentation**

**System Architecture:**

Design contains a header file and two .cpp files. Inside of the header file is a class that contains the declaration of all of the required functions. In the first cpp

File is the definitions of the functions that were declared in the header. There is also another cpp file where the functions are called.

**Functions and Purpose:**

**Name:** Length()

**Description:**  This function has the ability to find the length of a string.

.

**Name:** Compare(MyString other)

**Description:** This function has the ability to compare one instance of a string to another inside the class. In my function i needed to create a new string. Now attempting to compare the strings.

**Name:** Append(MyString other)

**Description:** This function has the ability toattach one string to the end of another.

**Name:** Return()

**Description:** This function has the ability to return a string as a constant char C-style

**Name:** ConvertLow()

**Description:** This function has the ability to convert a string to lowercase

**Name:** ConvertUp()

**Description:** This function has the ability to convert a string to uppercase

**Name:** SetCStyle()

**Description:** This string has the ability to set string to input C-style string have not attempted this function yet

**3. Source Code**

**MyString.h**

#pragma once

class MyString

{

public:

MyString() {};

MyString(char w[]);

//return string length as an integer

int Length();

//compare if one string is the same as another

bool Compare(MyString other);

//append one string to another

MyString Append(MyString other);

//return basic constant C-style string

const char\* Return();

//convert string to lowercase

char ConvertLow();

//convert string to uppercase

char ConvertUp();

//set string to input C-style string

const char\* SetCStyle();

private:

char m\_string[255];

};

**MyString.cpp**

#include <iostream>

#include"MyString.h"

using namespace std;

MyString::MyString(char newString[])

{

int i;

for (i = 0; newString[i] != '\0'; i++)

{

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

}

m\_string[i] = '\0';

}

int MyString::Length()

{

// function finds the length of character array

// loops through array until it reaches an exit point then returns how many indexs it went through

int i;

for (i = 0; m\_string[i] != '\0'; i++)

{

}

return i;

}

bool MyString::Compare(MyString other)

{

int i;

for (i = 0; i != other.Length() && Length(); i++)//loops until not equal to either strings

{

if (other.m\_string[i] == m\_string[i])//compares other to m\_string

{

cout << "strings are the same";

return true;

}

else

{

cout << "strings are not the same";

return false;

}

}

}

MyString MyString::Append(MyString other)

{

char carray[255];

int totalSize = this->Length() + other.Length() + 1;

// assign the size of both strings to a variable and adding 1 to hold the place of terminating character

//since arrays start at 0 we subtract 1 from the total length

carray[totalSize - 1] = '\0';

for (int i = 0; i < this->Length() + other.Length(); i++)

{

if (i < this->Length())// if the length is less than i then assign m\_string to carray

carray[i] = m\_string[i];

else//else assign other string to carray

carray[i] = other.m\_string[i - this->Length()];

}

MyString ms = MyString(carray);

return ms;

}

const char\* MyString::Return()

{

return m\_string;

}

char MyString::ConvertLow()

{

int i;

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

{

if (m\_string[i] > 64 && m\_string[i] < 91)

//if characters on ASCII table are 64-91 add 32 then return the character

{

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

//adding 32 to the decimal associated with the characters in the array

cout << m\_string[i];

}

}

return m\_string[i];

}

char MyString::ConvertUp()

{

int i;

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

{

if (m\_string[i] > 96 && m\_string[i] < 123)

//if characters on ASCII table are 96-123 subtract 32 then return the character

{

(char)m\_string[i] = (int)m\_string[i] - 32;

//subtracts 32 to the decimal associated with the characters in the array

cout << m\_string[i];

}

}

return m\_string[i];

}

const char \* MyString::SetCStyle()

{

return m\_string;

}

**Main.cpp**

#include <iostream>

#include"MyString.h"

using namespace std;

int main()

{

MyString s = MyString("turtle");

MyString other = MyString("poop");

cout <<"the length of the string is: " << s.Length() << endl;

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

std::cout << s.Append(other).SetCStyle() << std::endl;

cout<<s.Return() << endl;

cout <<"is the string uppercase" << s.ConvertUp() << endl;

cout <<"is the string lowercase" << s.ConvertLow() << endl;

system("pause");

}