**Project 1**

1. **Problem Statement**

Write a program that contains a string class called myString. This class should store a string of up to 25 characters. The class should include methods to manipulate the string, as well as methods to set the string to a string data value and display the string value to the end user. The actions performed in the class should be logged in an output file, which will not only show the actions performed, but the original value of myString, parameters, results, successes, and error messages, too. The main function of this program should test the functions of the myString class and display the actions to the end user, utilizing a method to display the results.

1. **Requirements**
   1. **Assumptions**

* There should be a limit of 25 characters for the myString class input
* The user will only input string
* Only use command line input/output
  1. **Specifications**
* The program’s class will store only up to 25 characters
* The class will be called myString
* The user will input 25 characters to be stored
  + Only string input will be valid
  + An error message will appear if the 25-character limit is exceeded: “Error: Only 25 characters can be stored! Please reduce the number of characters in your input”
* The class should include a method to display the number of characters within the string
  + The class will be called “size()”
  + An empty string will display zero
* The class should also include various methods to manipulate the stored string
  + addStart(myString)
    - add the string as the input parameter to the beginning of the current string
      * Cannot exceed 25 characters
        + Error message if 25-character limit is exceeded: “Error: Only 25 characters can be stored!”
  + addEnd(myString)
    - add the string in the input parameter to the end of current string
      * Cannot exceed 25 characters
        + Error message if 25-character limit is exceeded: “Error: Only 25 characters can be stored!”
  + partString(startPos, length)
    - displays string from startPos for length given
      * if startPos < 0, then null string will be returned
      * throw exception if startPos > size
  + replPartString(myString, startPos)
    - will replace starting characters at startPos with input string
      * Error message if 25-character limit is exceeded: “Error: Only 25-characters can be stored!”
      * Utilize a length error exception
  + replWholeString(myString)
    - will replace what is currently in the string with the input parameter string
  + compareString(myString)
    - will compare the value of the current string with the value of the input parameter string
  + initString()
    - will initialize the string in myString to the null string
  + setString(string)
    - will assign a string to the myString data value
  + getString(string)
    - will return the data from the setString method
  + printStringScreen()
    - displays the myString data value on the screen to the end user
  + numericString()
    - will tell the end user if the string is an integer or real
  + alphabeticString()
    - will tell the end user if the string is composed of all alphabetic characters
* Input cannot be stored as a string variable
* The c++ string class variable can only be utilized for screen input and the setString() method
* An output file must be created
* The main function will test the methods’ functions and write the actions into the output file, as well as display the results on the screen to the end user

1. **Decomposition Diagram** (Used to break program down into components visually. Can have as many components as needed. Defines functionality that will solve the problem – does NOT define a flow )

Main

Write all output to an output file

Replace starting characters with input string

Assign string to myString data value and return the value

Replace current string with input

Place the string input at the end of the string

Place the string input at the beginning of the string

Print out output file contents and results of each method

Print out error message if 25 character limit is exceeded

Check if input exceeds 25 characters

User inputs string via command line

Input

Output

Process

1. **Test Strategy**

* Output file existence
* Invalid data
* Valid data
* Input within 25 character limit

1. **Test Plan Version 1**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Output file existence | 1 | User attempts to open file |  |  |  |  | |
| Input within 25 character limit | 4 | User inputs more than 25 characters |  |  |  |  | |
| Input within 25 character limit | 4 | User inputs less than 25 characters |  |  |  |  | |
| Invalid data | 2 | User inputs integer value |  |  |  |  | |

1. **Initial Algorithm**
2. Create a string class called myString that will accept string input that does not exceed 25 characters and store it as an array of characters.
   1. Throw length exception if character limit is exceeded and display an error message to the end user: “Error: Only 25 characters can be stored!”
   2. Initialize the myString string data value
   3. Create a function that writes the actions of the class into an output file
   4. The string class should also have several methods to manipulate the string input
      1. Create a setter method called setString()
         1. Assign the string input to the myString data value
      2. Create a getter method called getString()
         1. Return the string assigned to the myString data value in setter method
      3. Create size() method
         1. Return the number of characters input by the end user
         2. Return zero if string is empty
      4. Create printStringScreen() method
         1. Return myString data value to end user
      5. Create numericString() method
         1. Return whether string is an integer or real number
      6. Create alphabeticString() method
         1. Return whether string is composed of all alphabetic characters
      7. Create a method that will compare the string input to the string stored in the class called compareString(string)
         1. This method should have the input as the parameter
      8. Create a method that will replace the current stored string with the new string input called replWholeString(myString)
         1. The input parameter will replace current stored string
      9. Create a method that will place the end user’s input at the beginning of the string stored in the class called addStart(mystring)
         1. Throw length error exception if exceeds 25 character limit
            1. Error message to the end user: “Error: Please do not exceed 25 characters!”
         2. The input parameter is the myString data value initialized and set with the setter method
      10. Create a method will place the end user’s input at the end of the string stored in the class called addend(myString)
          1. Throw length error exception if exceeds 25 character limit
             1. Error message to end user: “Error: please do not exceed 25 characters!”
          2. The input parameter is the myString data value initialized and set with setter method
      11. Create a method that returns the string from the starting position within the length specified by the user called partString(startPos, length)
          1. Initialize length variable of integer data value
          2. Initialize startPos variable to integer data value
          3. Ask end user to enter amount of characters they wish to have returned from the string class
             1. Set input to the length variable
          4. The startPos data value and the length are the input parameters for the method
          5. If startPos < 0, return null string
          6. If startPos > size, throw length error exception
      12. Create a method that will replace part of the stored string with the end user input using the myString data value and the startPos data value as parameters called partString
          1. Initialize startPos variable as integer data value
          2. Ask end user what position would they like to begin replacing the stored string at
          3. Throw length error exception if exceeds 25 character limit
          4. Return the string to end user with the replacement made
      13. Create a method that will reset the array to the null string called initString()
3. In main function
   1. Initialize character variable called “reset”
   2. Ask end user to enter a 25 character string input
   3. Set input as array contents in myString class
   4. Call each method of the myString class beginning with the size() method to return the size of the input to the end user
   5. Ask user if they wish to clear the string input (y or n)
      1. Set the input to the reset character variable
      2. If y, call initString() method
      3. If n, prompt user to choose a function of the myString class
   6. Call the method the user selects until user exits program
   7. Return the output file data as the program returns the function output, as well
4. **Updated Test Plan**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Output file existence | 1 | User attempts to open file | User opens file | The file should exist |  |  |
| Input within 25 character limit | 4 | User inputs more than 25 characters | The input is at least 26 characters | The error message will warn the user to remain within the 25 character limit |  |  |
| Input within 25 character limit | 4 | User inputs less than 25 characters | 24 character or less string input | The program saves the input into the array and proceeds with the size method |  |  |
| Invalid data | 2 | User inputs integer value | Any integer 1-infinity | The program will prompt the user to input string data |  |  |
| Invalid data | 2 | User inputs integer value for character value | Any integer 1-infinity | The program will prompt the user again to input whether they wish to reset the string array |  |  |
| Valid data | 4 | User inputs string data within the 25 character limit for addStart method | The user inputs string that will not exceed the 25 character limit once added to the stored string | The program will display the new string with the added input |  |  |
| Invalid data | 2 | User exceeds 25 character limit for addStart method | The user’s input, when added to the stored string, will be over 25 characters | The program will give an error message that the input cannot exceed 25 characters |  |  |
| Invalid data | 2 | User exceeds 25 character limit for addend method | The user’s input, when added to the stored string will be greater than 25 characters | The program will display the error message that the input cannot exceed 25 characters |  |  |
| Valid data | 4 | User inputs string within the 25 character limit for addEnd method | The user’s input, when added to the stored string, is within the 25 character limit | The program will display the stored string with the added string input |  |  |

1. **Code**

//Assignment One

//Author: Mariah Diaz

//This program utilizes a class named myString to manipulate a string and

//tests the functions of said class

#include<iostream>

#include<string>

using namespace std;

//Mystring class

class MyString {

//Class variables

private:

string myString;

//Class functions

public:

//Default constructor

MyString() {

myString = "";

}

//Return size of the string

int size() {

return myString.length();

}

//Add string to the start

void addStart(string mystring) {

if (size() > 25) {

cout << "You can't add string at the front.String already in length 25" << endl;

}

else if ((size() + mystring.length()) > 25) {

cout << "You can't add string at the front.String has maximum length 25" << endl;

}

else {

myString = mystring + myString;

cout << "You successfully added new string." << endl;

}

}

//Add string to the end

void addEnd(string mystring) {

if (size() > 25) {

cout << "You can't add string at the end.String already in length 25" << endl;

}

else if ((size() + mystring.length()) > 25) {

cout << "You can't add string at the end.String has maximum length 25" << endl;

}

else {

myString = myString + mystring;

cout << "You successfully added new string." << endl;

}

}

//Display only part of the string

void partString(int startPos, int length) {

if (startPos < 0) {

cout << "Error!!! String index start from 0." << endl;

}

else if (startPos > 25) {

cout << "Error!!! Maximum string length is 25." << endl;

}

else if (startPos + length > 25) {

cout << "Error!!! Maximum string length is 25.." << endl;

}

else {

for (int i = startPos; i <= length; i++) {

cout << myString[i];

}

cout << endl;

}

}

//Replace the string

void replPartString(string mystring, int startPos) {

if (startPos < 0) {

cout << "Error!!! String index start from 0." << endl;

}

else if (startPos > 25) {

cout << "Error!!! Maximum string length is 25." << endl;

}

else if ((size() - startPos) + mystring.length() > 25) {

cout << "Error!!! Maximum string length is 25.." << endl;

}

else {

string temp = "";

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

if (i == startPos - 1) {

for (int j = 0; j < mystring.length(); j++) {

temp += mystring[j];

}

myString = temp;

}

else {

temp += myString[i];

}

}

cout << "Successfully replaced!!!" << endl;

}

}

//Replace the string with a new string

void replWholeString(string mystring) {

if (mystring.length() > 25) {

cout << "Can't replace.Max length of the string is 25." << endl;

}

else {

myString = mystring;

cout << "Successfully replaced." << endl;

}

}

//Compare strings

bool compareString(string mystring) {

if (size() != mystring.length()) {

return false;

}

else {

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

if (myString[i] != mystring[i]) {

return false;

}

}

return true;

}

}

//Initialize the string

void initString() {

myString = "";

}

//Setter

void setstring(string mystring) {

if (mystring.length() > 25) {

cout << "Cannot add string.Max string length is 25." << endl;

}

else {

myString = mystring;

}

}

//Getter

string getString() {

return myString;

}

//Print the string

void printStringScreen() {

cout << "My string = " << myString << endl;

}

//Check for numerics

bool numericString() {

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

if ((myString[i]) >= '0' && (myString[i]) <= '9' || (myString[i]) == '.')

i++;

else

return false;

}

return true;

}

//Check for alphabetics

bool alphabeticString() {

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

if (!isalpha(myString[i]))

return false;

}

return true;

}

};

int main()

{

MyString mstring;

mstring.setstring("World!!Good Morning");

mstring.printStringScreen();

//Add at start and print

mstring.addStart("Hello ");

mstring.printStringScreen();

//Add end and print

mstring.addEnd(".Nice Day.");

mstring.printStringScreen();

//Add end check

mstring.addEnd(".");

mstring.printStringScreen();

//Print the string size

cout << "Size of the string= " << mstring.size() << endl;

//Part of the string

mstring.partString(1, 10);

//Replace the string

mstring.replPartString("Day", 19);

mstring.printStringScreen();

//Repalce the entire string

mstring.replWholeString("How are you?");

mstring.printStringScreen();

//Compare string

if (mstring.compareString("How are you?") == 1)

{

cout << "Both strings are equal" << endl;

}

else {

cout << "Both strings are not equal" << endl;

}

//Check for numeric

if (mstring.numericString()) {

cout << "String is numeric" << endl;

}

else {

cout << "String is not numeric" << endl;

}

//Check for alphabetic

if (mstring.alphabeticString()) {

cout << "String is alphabetic string" << endl;

}

else {

cout << "String is not alphabetic string" << endl;

}

//Check for numeric

mstring.setstring("123.45");

if (mstring.numericString()) {

cout << "String is numeric" << endl;

}

else {

cout << "String is not numeric" << endl;

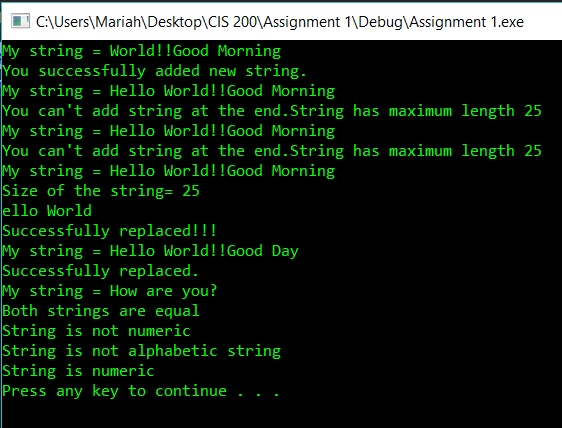
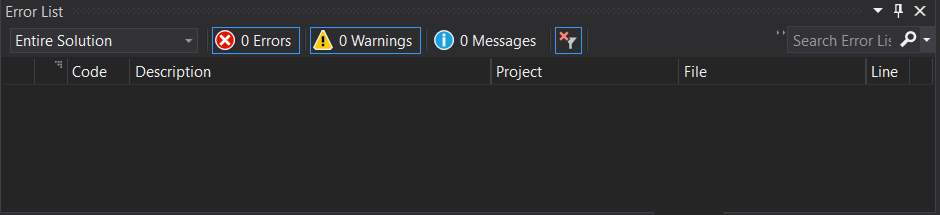
}

system("pause");

}

1. **Updated Algorithm**
2. Create a string class called myString that will accept string input that does not exceed 25 characters ~~and store it as an array of characters~~.
   1. Throw length exception if character limit is exceeded and display an error message to the end user: “Error: Only 25 characters can be stored!”
   2. Initialize the myString string data value
   3. ~~Create a function that writes the actions of the class into an output file~~
   4. The string class should also have several methods to manipulate the string input
      1. Create a setter method called setString()
         1. Assign the string input to the myString data value
      2. Create a getter method called getString()
         1. Return the string assigned to the myString data value in setter method
      3. Create size() method
         1. Return the number of characters input by the end user
         2. Return zero if string is empty
      4. Create printStringScreen() method
         1. Return myString data value to end user
      5. Create numericString() method
         1. Return whether string is an integer or real number
      6. Create alphabeticString() method
         1. Return whether string is composed of all alphabetic characters
      7. Create a method that will compare the string input to the string stored in the class called compareString(string)
         1. This method should have the input as the parameter
      8. Create a method that will replace the current stored string with the new string input called replWholeString(myString)
         1. The input parameter will replace current stored string
      9. Create a method that will place the end user’s input at the beginning of the string stored in the class called addStart(mystring)
         1. Throw length error exception if exceeds 25 character limit
            1. Error message to the end user: “Error: Please do not exceed 25 characters!”
         2. The input parameter is the myString data value initialized and set with the setter method
      10. Create a method will place the end user’s input at the end of the string stored in the class called addend(myString)
          1. Throw length error exception if exceeds 25 character limit
             1. Error message to end user: “Error: please do not exceed 25 characters!”
          2. The input parameter is the myString data value initialized and set with setter method
      11. Create a method that returns the string from the starting position within the length specified by the user called partString(startPos, length)
          1. Initialize length variable of integer data value
          2. Initialize startPos variable to integer data value
          3. Ask end user to enter amount of characters they wish to have returned from the string class
             1. Set input to the length variable
          4. The startPos data value and the length are the input parameters for the method
          5. If startPos < 0, return null string
          6. If startPos > size, throw length error exception
      12. Create a method that will replace part of the stored string with the end user input using the myString data value and the startPos data value as parameters called partString
          1. Initialize startPos variable as integer data value
          2. Ask end user what position would they like to begin replacing the stored string at
          3. Throw length error exception if exceeds 25 character limit
          4. Return the string to end user with the replacement made
      13. Create a method that will reset the array to the null string called initString()
3. In main function
   1. Initialize character variable called “reset”
   2. Ask end user to enter a 25 character string input
   3. Set input as array contents in myString class
   4. Call each method of the myString class beginning with the size() method to return the size of the input to the end user
   5. Ask user if they wish to clear the string input (y or n)
      1. Set the input to the reset character variable
      2. If y, call initString() method
      3. If n, prompt user to choose a function of the myString class
   6. Call the method the user selects until user exits program
   7. Return the output file data as the program returns the function output, as well
4. **Final Test Plan**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Output file existence | 1 | User attempts to open file | User opens file | The file should exist | Exists | Pass |
| Input within 25 character limit | 4 | User inputs more than 25 characters | The input is at least 26 characters | The error message will warn the user to remain within the 25 character limit | Error message | Pass |
| Input within 25 character limit | 4 | User inputs less than 25 characters | 24 character or less string input | The program saves the input into the array and proceeds with the size method | Saves | Pass |
| Invalid data | 2 | User inputs integer value | Any integer 1-infinity | The program will prompt the user to input string data | Prompt | Pass |
| Invalid data | 2 | User inputs integer value for character value | Any integer 1-infinity | The program will prompt the user again to input whether they wish to reset the string array | Prompt | Pass |
| Valid data | 4 | User inputs string data within the 25 character limit for addStart method | The user inputs string that will not exceed the 25 character limit once added to the stored string | The program will display the new string with the added input | Displays string | Pass |
| Invalid data | 2 | User exceeds 25 character limit for addStart method | The user’s input, when added to the stored string, will be over 25 characters | The program will give an error message that the input cannot exceed 25 characters | Error | Pass |
| Invalid data | 2 | User exceeds 25 character limit for addend method | The user’s input, when added to the stored string will be greater than 25 characters | The program will display the error message that the input cannot exceed 25 characters | Error | Pass |
| Valid data | 4 | User inputs string within the 25 character limit for addEnd method | The user’s input, when added to the stored string, is within the 25 character limit | The program will display the stored string with the added string input | Displays string | Pass |

1. **Screenshots**
2. **Error Log**

Any issues you had while testing your code are recorded in the error log as you perform testing of the “completed” code – that is, when you run through all of the test cases in the test plan.

|  |  |  |
| --- | --- | --- |
| Error Type | Cause of Error | Solution to Error |
| Log 2 types of errors:  Logic  Runtime | What specifically caused the error to occur | What did you do/change to fix the error |
|  |  |  |

Do not list any syntax errors or errors detected in unit testing as you build your program.

1. **Status**

The program works as it should, 100%.