Sandesh Ghimire

CSCI 301 section 3

Computer Science 2

Project 1

1st September 2020

**Design Document**

**Introduction**

An Anagram is a word or a phrase formatted by rearranging the alphabet of a different word or a phrase in such a way that both contain the same number of letters. For example: “Hello World” and “hello?world” are anagrams, “It's my life” and “It's your life” are not anagrams. This program finds weather two phrases are anagrams or not. It considers any words or a phrase that contains spaces, numbers, and special characters. This program reads those two phrases of input from users with a maximum length of 60 characters, excludes the spaces, numbers, and special characters. After excluding them, the program sorts both phrases and check if they are equal or not. At last prints “are anagram” if they are equal and “are not anagram” if they are not equal via terminal.

**Data Structure**

The program uses two data structures. The first one is a string which is also an array of characters. It holds the two input phrases from the user. A validation is set on the string which only allows the user to enter phrase less than or equal to 60 characters. Those phrases are filtered and sorted calling the next function. The second data structure is a Stack. It is used to store alphabets when the program excludes numbers, spaces, and special characters.

**Functions**

The program uses two functions. One for excluding the spaces, numbers, and special characters as well as checking if the phrases are anagram or not. And another for sorting the given phrases. The function “areAnagram()” takes two strings and returns a Boolean value. It is called the main function. The function filters the phrase and take out numbers, special characters, and spaces and then stores them at two different variables using Stack (an ADT). This function also calls the other function i.e. “Sort()” which takes a string as a parameter and returns a string value. It sorts the phrase that was filtered and returns and replaces it to the previous variable. Now the function “areAnagram()” checks if both filtered and sorted strings are equal or not. If they are equal it returns true else returns false to the main function.

**The Main Program**

The main program askes two inputs for the user and stores it in variables. It starts with a loop for a validation check of entered data by the user. It checks if the input value is less than or equal to 60 characters or not. If the value is more than 60 then it gives an error message telling the user to enter again. Finally, it called “areAnagram()” function and then prints if they are anagram or not.

If the function “areAnagram()” returns true then the program displays a message telling that the phrases are anagram.

**Code**

//CSCI 301

//Computer Science 2

//This program asks user to enter two phrases and outputs if they are anagram or not

//The input is taken from the main function with the validation

//check. It checks if the input of user has characters more than 60 or not.

//It keeps on displaying the error message until the user inputs strings less than 60 characters.

//Each input is used to call a function that firstly deletes any

//spaces, numbers and special characters. After that this function calls another function to sort the

//strings. After sorting the function checks if the two strings are equal or not.

//If they are equal function returns true(a boolean value) to the main function and if they are

//not equal it returns false.

//In the end the main function displays if the entered strings are anagram or not.

#include <iostream>

#include<cstring>

using namespace std;

string Ssort(string);

bool areAnagram(string, string);

//This function sorts the formatted strings taken from user.

string Ssort(string abc1) {

for (int index = 0; index < abc1.size() - 1; index++) { //starting outer loop for sorting

for (int i = index + 1; i < abc1.size(); i++) { //Starting inner loop for sorting

if (abc1[index] > abc1[i]) // comparing two characters

swap(abc1[i], abc1[index]); //swapping if the condition is mate

}

}

return abc1; //returns sorted value to " areAnagram" function

}

//function to find if the lines are anagram or not

bool areAnagram(string abc1, string abc2) {

string ForStr1, ForStr2; //strings to store the phrase without special characters,numbers and spaces.

//taking out the spaces, numbers and special characters.

for (int i = 0; i < abc1.length(); i++) //starting the loop

{

if ((abc1[i] >= 'A' && abc1[i] <= 'Z') || (abc1[i] >= 'a' && abc1[i] <= 'z')) {

ForStr1.push\_back(tolower(abc1[i]));

}

}

for (int i = 0; i < abc2.length(); i++) {

if ((abc2[i] >= 'A' && abc2[i] <= 'Z') || (abc2[i] >= 'a' && abc2[i] <= 'z')) {

ForStr2.push\_back(tolower(abc2[i]));

}

}

ForStr1 = Ssort(ForStr1); //sorting the phrase using "Ssort" function

ForStr2 = Ssort(ForStr2); //sorting the phrase using "Ssort" function

if (ForStr1 == ForStr2) { //checking if two phrases are same or not

return true;

} // returning the boolean value to the main function if true.

else {

return false;

}

}

//the main function

int main() {

string abc1 = " "; //declaring and initializing the variable for input 1.

string abc2 = " "; //declaring and initializing the variable for input 2.

bool check = true; //declaring a boolean value.

do { //starting the loop.

cout << "Enter two line that might be anagram: " << endl; //asking the input from users

cout << "Line 1: ";

getline(cin, abc1); //storing the input to abc1 variable.

if (abc1.length() >= 60) { //validation check: if the string is greater then 60 keep not..

cout << "Error: Please enter only the phrase that is less then or equal to 60 characters" << endl; //error message

check = false; //set the value to false

} else { //if condition is false

check = true; //set value to true

cout << "Line 2: "; //asking another phrase

getline(cin, abc2);

if (abc2.length() >= 60) { ////validation check: if the string is greater then 60 keep not..

cout << "Error: Please enter only the phrase that is less then or equal to 60 characters" << endl; //error message.

check = false;

} else {

check = true;

} //if statement is false

}

} while (check == false); //ending the loop with condition.

if (areAnagram(abc1, abc2) == true) //calling the areAnagram() function

cout << "The two phrases are anagram with each other"; //print this value if true

else

cout << "The two phrases are not anagram with each other"; //print this value if false

return 0;

}

**User Document**

Anagram is a word or a phrase formatted by rearranging the alphabet of different word or a phrase in such a way that both contains the same number of letters. The program *IsAnagram.cpp* finds if the two phrases or words are anagram or not. The user enters two phrases or words whose length is less than 60 characters. In the end the program displays if the entered phrases are anagram or not.

The program’s name is *IsAnagram.cpp* ; to compile the program simply enter:

*g++ -o IsAnagram IsAnagram.cpp*

To run the program, enter. */IsAnagram,* then you can input two phrases. After entering the phrases, the program will display if the phrases are anagram or not.

For example, when you run the program it asks input:

*Prompt>./IsAnagram*

*Enter two line that might be anagram:*

*Line1: Hello world*

*Line2: hello?world*

If you enter those two lines the program displays,

*The two phrases are anagram with each other*

Note that no input should be greater than 60 characters else it will show the following error.

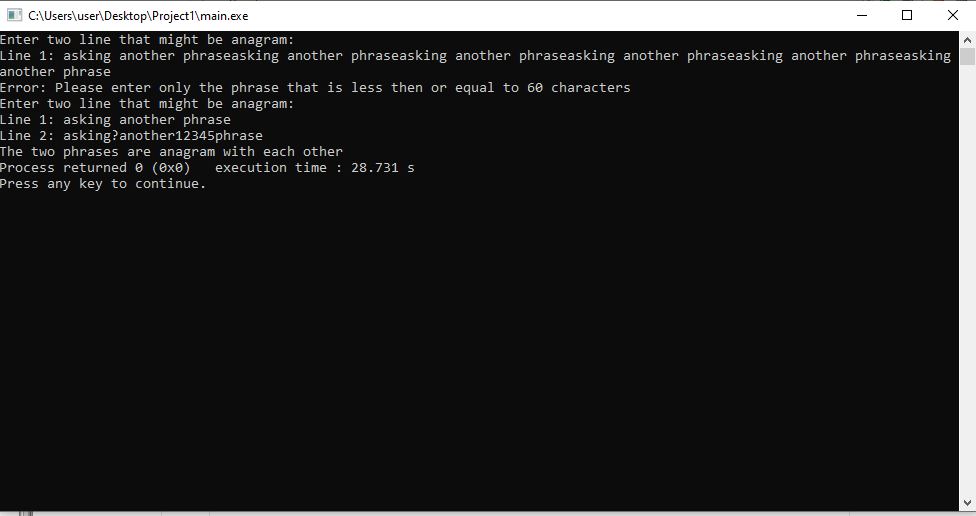
*Error: Please enter only the phrase that is less then or equal to 60 characters*

*Enter two line that might be anagram:*

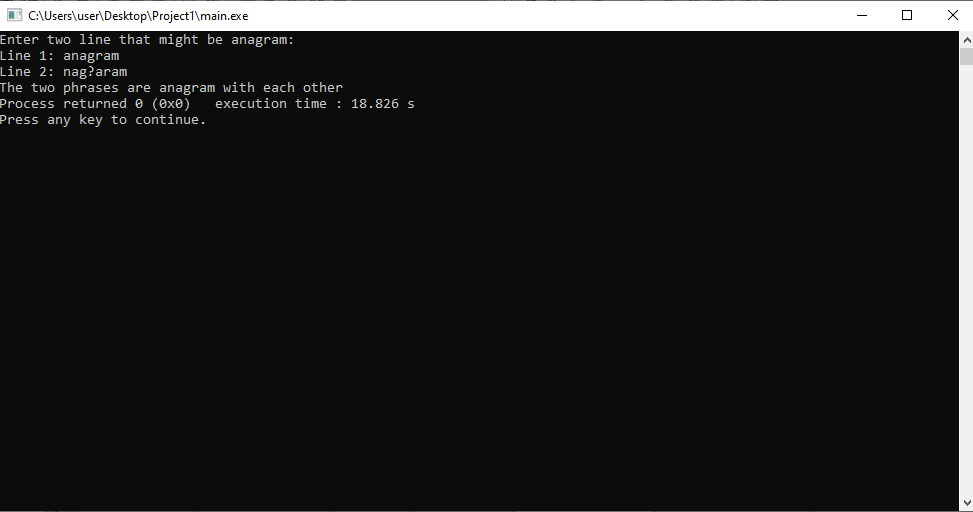
*Line1:*

**Testing**

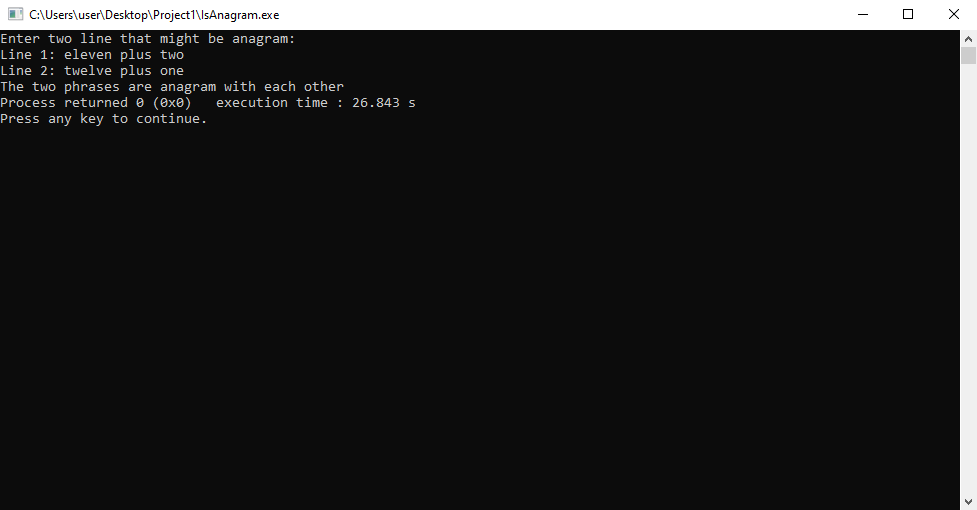
*Test 1*

****

*Test 2*

****

*Test 3*

****

**Summary**

In this project, we learned about anagrams. We entered a code that would let us find if the entered two phrases are anagram or not. It uses a lot of string functions from cstring library.

This program lets us workout on how to work with strings. Also understanding the sorting. This program can be modified furthermore. We could have used bits/stdc++.h library which will allow us to sort just using a line of code. We can also use an array of characters while formatting (deleting the numbers, special characters, and spaces from phrase) the phrases but instead we used Stack (ADT) which is easier to do.