CSC 150 LAB 2-1
STRING CLASS OPERATIONS

PROFESSOR GODFREY C. MUGANDA
DEPARTMENT OF COMPUTER SCIENCE

The purpose of this lab is to explore various member functions of the string class.

Remember that a type consists of, not only a set of values, but also a set of operations that can be performed on those values. This lab studies the set of operations that can be performed on string values.

NOTE: The output of the your program must match exactly the example output shown, down to line structure.

1. String Input and Output

You can read string values from the keyboard, from some other input stream, int a string variable. This can be done two ways:

- By using the `iostream` extraction operator `>>` to read a single "word":
- By using the string library function `getline` to read up to the end of the line:

Create a project called `CSC150Lab2_1`. Add a new source file, which you should name `CSC150Lab2_1.cpp`, prefixed by your name. For example, if you were me, you would use the name `gcmugandaCSC150Lab2_1.cpp`.

Type in the following code to ask the user to enter in two lines of input, read the two lines, and echo them back to the user:

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
    string str1, str2;
    // Prompt for a string and read the string
    cout << "Enter a string on a line: ";
    getline(cin, str1);
    // Prompt for the second string and read the string
    cout << "Enter a second string on a line: ";
    getline(cin, str2);

    // Echo the strings entered
    cout << "You entered the strings: \n";
    cout << str1 << "\n";
    cout << str2 << "\n";
}
```

When you run this program, you should see output like this:

```
Enter a string on a line: North Central College
Enter a second string on a line: Department of Computer Science
You entered the strings:
North Central College
Department of Computer Science
```

2. Outputting the Strings in Reverse Order

Add code to your program that prints the two strings entered in the reverse order. Here is a sample output:

```
Enter a string on a line: Barack Obama
Enter a second string on a line: Joseph Biden
You entered the strings:
Barack Obama
Joseph Biden
The reversed order of the strings is:
Joseph Biden
Barack Obama
```

3. Comparing String Objects

The *relational operators*

```
<   <=   >   >=  ==   !=
```

are overloaded to work with objects of the `string` class. They compare strings using alphabetic, or dictionary ordering, to see whether two strings are equal or unequal, or to see which string should come before or after the other.

Add code at the end of your program that prints the strings entered in alphabetical order, with the first string printed prefixed by (1) and the second by (2).

```
Enter a string on a line: Where you are central
Enter a second string on a line: North Central College
You entered the strings:
Where you are central
North Central College
The reversed order of the strings is:
North Central College
Where you are central
The two strings in alphabetic order:
(1) North Central College
(2) Where you are central
```

NOTE: From now on, you will still be modifying your program by adding code at the end. However, the sample output may show only the input strings and the output from the newly added code.

Next, add code that determines whether the two strings entered are equal on a character by character basis, or whether they are different. Here is a sample run:

```
Enter a string on a line: Donald Trump
Enter a second string on a line: Ben Carson


The strings are not equal.
```

Here is a second sample run:

```
Enter a string on a line: It was the worst of times
Enter a second string on a line: It was the worst of times


The strings are equal.
```

4. Appending to a String

To append to a string means to extend the string by adding another string to the end of it.

The `string` class has a member function

```
string append(string str)
```

For example, the code

```
string str = "mom";
str.append(" and dad");
cout << str << endl;
```

will print the phrase `"mom and dad"`.

5. Inserting into a String

You can insert a string into another string at a specified position. To do this, you use the `string` class member function

```
string insert(int pos, string str)
```

For example, the code

```
string str = "mom";
str.insert(0, "dad and ");
cout << str << endl;
```

will print the phrase `"dad and mom"`.

As another example, the code

```
string str = "mom";
str.insert(1, "ary is a wonderful m");
cout << str << endl;
```

will print the phrase `"mary is a wonderful mom"`.

Modify your program by adding code to do the following. First, the code asks the user to enter three strings, with each string being entered on its own line. Second the code reads in the three strings and stores them into three string variables.

Finally, the code inserts the second string at position 4 of the first string, and then appends the third string at the end.

We will assume that the first of the three strings entered has at least 5 characters.

Here is a sample run:

```
Enter 3 strings, each string on its own line:
The force awakens
sun in full
 the dawn.
The sun in full force awakens the dawn.
```

There the three strings entered by the user are

1. `"The force awakens"`
2. `"sun in full "`
3. `" the dawn."`

6. Finding a String Within another String

The member function `find()` comes in two varieties. The first variety is

```
size_t find(string str)
```

which returns the position of the string `str` within the original string object.

You can think of the type `size_t` as integers that indicate a position within a string. For example, the code

```
string mystr4 = "All dogs growl bark and eat";
size_t pos = mystr4.find("dogs");
```

assigns the value 4 to the variable `pos`, because the string `"dogs"` is found at position or index 4 within `mystr4`.

Similarly, the code

```
string mystr4 = "All dogs growl bark and eat";
size_t pos = mystr4.find("ogs");
```

would assign the value 5 to the variable `pos`.

The second variety of `find()` takes a single character and its position within the string:

```
size_t find(char ch)
```

As an example, the code

```
string mystr4 = "All dogs growl bark and eat";
size_t pos = mystr4.find('g');
```

will assign the value 6 to `pos`. Notice that there are two occurrences of the character 'g' in the string. The `find` function always returns the position of the first character or substring that matches.

There is one other thing we need to know about `find`. If the searched for substring or character is not found, the function returns a special `size_t` value called `string::npos`, for “no position.” It is a good idea to test the returned value `pos` before you use it in your program. To test it, use the boolean expression

```
pos == string::npos
```

in an *if statement.* If the above boolean expression is true, then you print a message saying the searched-for substring was not found. But if it is false, then print a message indicating the position at which the substring was found.

For the final step of this lab, add the statement

```
string mystr4 = "All dogs growl bark and eat";
cout << "Here is a string: ";
cout << mystr4 << endl;
```

at the end of your program. This statement just initializes a string variable with the sentence shown above, and then displays the string so the user can see it. Make this modification and run the program.

Next, add a variable

```
string search_str;
```

to your program. This variable will be used to store a string that the user enters, so you can search `mystr4` for an occurrence of the string. Add code that asks the user to enter a string, and then have your program read the string and either tell the user the string is not found in `mystr4`, or print the position at which it is found.

Your program should give the user three opportunities to enter a search string. After that, your program should terminate.

Altogether, a sample run of your program should look like this

```
Enter a string on a line: It was the best of times
Enter a second string on a line: Do or do not, there is no try

You entered the strings:
It was the best of times
Do or do not, there is no try

The reversed order of the strings is:
Do or do not, there is no try
It was the best of times

The two strings in alphabetic order:
(1) Do or do not, there is no try
(2) It was the best of times

The strings are not equal.

Enter 3 strings, each string on its own line:
Watson, come here
I am busy
Well then you are fired.
```

```
WatsI am busyon, come hereWell then you are fired.

Here is a string:
All dogs growl bark and eat

Enter a string to search for: growl
growl is found at position 10

Enter a string to search for: cats
cats is not found

Enter a string to search for: bark
bark is found at position 16
```

Due date : Thursday of Week 2 at midnight.

The deadline for submission is Friday night at midnight.