

# 1 Introduction

Each exercise should be its own separate project. The exercises are from previous assignments that you will update to use functions.

**Remember to follow the coding and documentation standards for the class listed on the MyClasses pages.**

When you are ready to submit your work create a folder called Homework04 in that folder have separate folders for each project, one folder per project. Put all the code files needed for that project in its respective folder. Do not include the files that the IDE creates, I just want the code files. Zip the entire Homework04 folder up into a single zip file and submit it.

## 2 Exercises

1. In this exercise you will be creating a few useful functions for strings. For each of the following put a prototype at the top and the function implementation below the main.

- (a) `to_uppercase` — brings in a string by reference and changes it to uppercase. Note that the following loop will convert the string `str` to uppercase. The function `toupper` converts a single character to uppercase in a similar manner as the character arithmetic we saw in some of the examples.

```
for (int i = 0; i < str.length(); i++)
    str[i] = toupper(str[i]);
```

- (b) `to_lowercase` — brings in a string by reference and changes it to lowercase. The above loop will do that if you change `toupper` to `tolower`.
- (c) `to_uppercase_copy` — brings in a string by value and returns the uppercase version of it.
- (d) `to_lowercase_copy` — brings in a string by value and returns the lowercase version of it.
- (e) `ltrim` — brings in a string by reference and left trims it by removing all whitespace at the beginning. The following line of code will replace the string `str` with one that has the beginning whitespace removed.

```
str.erase(0, str.find_first_not_of("\t\n\r\v\f "));
```

The way this works is that the `find_first_not_of` function of the string class will find the first character that is not a space, tab, new line, return, vertical tab, or form feed. That is, the first character that is not a whitespace character. The `erase` command will then erase all characters from index 0 up to but not including the first non-whitespace character.

- (f) `rtrim` — brings in a string by reference and right trims it by removing all whitespace at the end. The following line of code will replace the string `str` with one that has the ending whitespace removed.

```
str.erase(str.find_last_not_of("\t\n\r\v\f ") + 1);
```

This works in a similar fashion by finding the last non-whitespace character and then erasing all characters after it.

- (g) `trim` — brings in a string by reference and trims it by removing all whitespace at both the beginning and the end.
- (h) `ltrim_copy` — brings in a string by value and returns the left trimmed string.
- (i) `rtrim_copy` — brings in a string by value and returns the right trimmed string.
- (j) `trim_copy` — brings in a string by value and returns the trimmed string.

Use the main program below,

```
int main() {
    string str = "Hi There 123 &*^#$$%#";
    string str2 = "";

    cout << str << endl;
    to_uppercase(str);
    cout << str << endl;
    to_lowercase(str);
    cout << str << endl;
    cout << endl;

    str2 = to_uppercase_copy(str);
    cout << str << endl;
    cout << str2 << endl;
    cout << endl;

    to_uppercase(str);
    str2 = to_lowercase_copy(str);
    cout << str << endl;
    cout << str2 << endl;
    cout << endl;

    str = "  Hi There  ";

    cout << "*****" << str << "*****" << endl;

    ltrim(str);
    cout << "*****" << str << "*****" << endl;

    str = "  Hi There  ";
    rtrim(str);
    cout << "*****" << str << "*****" << endl;

    str = "  Hi There  ";
    trim(str);
    cout << "*****" << str << "*****" << endl;
    cout << endl;

    str = "  Hi There  ";
    str2 = ltrim_copy(str);
    cout << "*****" << str << "*****" << endl;
    cout << "*****" << str2 << "*****" << endl;
```

```

    str2 = rtrim_copy(str);
    cout << "*****" << str << "*****" << endl;
    cout << "*****" << str2 << "*****" << endl;

    str2 = trim_copy(str);
    cout << "*****" << str << "*****" << endl;
    cout << "*****" << str2 << "*****" << endl;

    return 0;
}

```

With your functions, this should create the following output.

```

Hi There 123 &^#$%#
HI THERE 123 &^#$%#
hi there 123 &^#$%#

hi there 123 &^#$%#
HI THERE 123 &^#$%#

HI THERE 123 &^#$%#
hi there 123 &^#$%#

*****  Hi There  *****
*****Hi There  *****
*****  Hi There*****
*****Hi There*****

*****  Hi There  *****
*****Hi There  *****
*****  Hi There  *****
*****  Hi There*****
*****  Hi There  *****
*****Hi There*****

```

- Write a program that will take a filename from the user and a target string that can either be a single word or a phrase of several words. The program is to find all of the occurrences of the input word or phrase in the file. The searching must be case insensitive. Also, keep in mind that the word or phrase could be present several times on one line. If the phrase is broken up between two lines in the file you do not need to worry about counting it. I have included two text files for this exercise, `DATHT.txt` and `HGGT.txt`. These files contain the Hitchhikers Guide to the Galaxy trilogy, which is actually 5 books. In the file `DATHT.txt` the text is broken into over 26,000 lines as one would expect to read and the file `HGGT.txt` is the same words but is on a single line. Your program must work for both files without alteration. An example run is below.

```

Input the input filename: DATHT.txt
Input the word or phrase to be found: Arthur

The file contains 1576 occurrences of "Arthur"

```

In the Hitchhikers Guide to the Galaxy the main characters are Arthur Dent, Ford Prefect, Zaphod Beeblebrox, Trillian, Marvin, and Slartibartfast. Do searches on both

files for Arthur, Arthur Dent, Ford, Zaphod, Beeblebrox, Zaphod Beeblebrox, Trillian, Marvin, and Slartibartfast and report the number of occurrences of each.

In this exercise create a function called `count_strings` that takes in two string parameters, the filename and the target string, and return the number of occurrences of the target string in the file. So the function will open the file, read through the file and count the occurrences, and finally return that number. If the file does not exist the function should return `-1`. The main should simply take the input, call the function and display the output. If the function returns `-1` the main should not display output of the number of occurrences. The checking should be case insensitive.

3. Create a program to play the following word scramble game. This is a two player game. Player 1 inputs a word while player two is not looking. This is a single word. The program then randomly scrambles the letters of the word and prints out the scrambled word for Player 2. The program gives Player 2 three guesses to get the word. If they are incorrect the program tells them and asks for another guess. If Player 2 guesses correctly then the program congratulates them and ends. If Player 2 does not guess correctly in three tries the program tells them that they lost and gives them the correct answer. The checking should be case insensitive and you should trim the whitespace from the front and back of all inputs. In this program you will create a function called `scramble_string` that will take a single string parameter by value and return a string with the letters of the original scrambled. There are lots of ways to do this but one way is to pick two character locations at random and then interchange the two letters. Do this about 100 times and the word should be scrambled well. The original word should be converted to all lowercase and the comparing must be case insensitive. A couple runs of this program are below.

```
Player 1: Input a single word: Committee
```

```
Player 2: The scrambled word is tmicemeto
You have three guesses at the original word.
```

```
Guess the original word: timcomeet
That is not correct, make another guess.
Guess the original word: COmmIttEE
Congratulations, you guessed the word correctly.
```

---

```
Player 1: Input a single word: COMMITTEE
```

```
Player 2: The scrambled word is eomtcimet
You have three guesses at the original word.
```

```
Guess the original word: committees
That is not correct, make another guess.
Guess the original word: therefore
That is not correct, make another guess.
Guess the original word: theorem
You have exhausted all three of your guesses.
The correct word was committee.
```

---

```
Player 1: Input a single word: longitude

Player 2: The scrambled word is itonduleg
You have three guesses at the original word.

Guess the original word: longitude
Congratulations, you guessed the word correctly.
```

4. Write a program that will find anagrams of an input word. An anagram is a word, phrase, or name formed by rearranging the letters of another, such as cinema, formed from iceman. The program should take a single word from the user and produce all possible anagrams of that word. Make sure that this is case insensitive. The file `words.txt` that can be downloaded from this lab contains 466,551 English words, names, and abbreviations, one word per line. Also note that the contents of the `words.txt` are not all uppercase. In this exercise you will create a function named `is_anagram` that will take as parameters two strings and return a `bool` of `true` if the two are anagrams and `false` if not. The main will take in the original word, run through the `words.txt` file and output all anagrams that were found. Determining an anagram can be done in a number of ways. One suggestion is to first check that the sizes are the same, if not then they cannot be anagrams. Then take the first word character by character and if that character is in the second word replace that character in the second word by a symbol, such as an `*`. When you are done with all the characters in the first word the second word should be all `*` if it is an anagram and not if not. A couple runs are below.

```
Input a single word: sector
Anagrams of SECTOR
CORSET
CORTES
COSTER
CROSET
ESCORT
RECAST
RECTOS
SCOTER
SECTOR

Input a single word: committee
Anagrams of COMMITTEE
COMMITTEE
```