Programming II

Assignment 0

Review of Programming 1

1 Programming Concepts

- Standard I/O
- Selection Constructs
- Loop Constructs

2 Task Summary

This programming task is a review of Programming I concepts up to defining functions. In the spirit of this course (and to introduce some basic game programming techniques), we will be implementing a very simple interactive game. For this assignment you will learn and demonstrate how to:

- Write and read data from the standard input and output streams,
- Utilize a loop construct
- Utilize multiple types of selection construct
- Declare and define variables

In this homework you will also be writing code that maintains a text-based GUI and updates a very simple game state.

3 Game Overview

This game is Hangman. The Instructor has provided a list of 30 different countries for you to use. The objective is to program the game's user interface and logic.

4 Rules of the Game

1. The player has to input a single character
2. The character from the player is case-insensitive
3. The player has at most 4 guesses
4. The game is over if the player has correctly guessed the country or they have ran out of guesses

5 Graphical User Interface

5.1 During Gameplay

During Gameplay, the player will begin guessing letters of the alphabet. The program should accept those letters, determine which characters in the string match and then print the user interface. The user interface is a box with plus signs in the corners. There is space between the secret word and the sides of the box. There is a `padding` variable to dictate how many spaces on each side. Underneath the secret word is an underline.

Underneath the user interface, the number of tries is printed and a message indicating to enter a guess.

```
+------------+
|   ******   |
|   ______   |
+------------+
NUMBER OF TRIES: 0
Enter a Guess!:
```

Figure 1: Example of the startup screen during gameplay

```
+-------------+
|   somal*a   |
|   _______   |
+-------------+
NUMBER OF TRIES: 1
Enter a Guess!:
```

Figure 2: Example of the typical screen during gameplay

5.2 After Gameplay

The game ends when the player has ran out of guesses or when they have successfully guessed the word. Notice how the congrats and failure text start at the beginning and not in the middle.

```
+-------------+
|   somalia   |
|   _______   |
+-------------+
CONGRATS
```

Figure 3: Example of the success screen

```
+------------+
|   ******   |
|   ______   |
+------------+
YOU FAILED
```

Figure 4: Example of the failure screen

6 Tasks

The instructor has provided you with the main game loop as well as all the necessary function prototypes and state variables. Your job is to fill in the function definitions to correctly implement the program as described by the rules and game play scenarios given above.

Below is a recommended list of tasks to complete the assignment.

6.1 Task 0: Download Base Files

1. Click on the link to the repository in Blackboard and a copy of the repository will be created for you.
2. You will use GitHub Desktop to clone the repository to your computer

6.2 Task 1: Read the Code

The code is very straight forward and should be readable from the top down. Sometimes it helps to draw a mindmap or talk to a rubber duck (I'm serious)

6.3 Task 2: Draw the Box

First try to print the box with padding. There is a variable `correctCountry` that will contain a random country name that you can use to make sure everything is correct.

6.4 Task 3: Implement fillLetters

The function `fillLetters` is a bit tricky. I would recommend either commenting out the functionality in main or use onlinegdb.com to isolate everything. The idea is to test and implement each function out incrementally. The fillletters needs to modify a global variable based on two parameters.

6.5 Task 4: Implement Missed Guesses

After you are sure that fillLetters is working, now add some functionality to increment `currentGuessNumber` when the user input a character that doesnt fit.

6.6 Task 5: Finish everything and submit

Drawing the user interface and implementing fillLetters should be the hardest part. Comparatively, everything else should be a walk in the park.

7 Grading Rubric

Your job is to design and implement a C++ procedural paradigm program that performs the tasks according to this specification (and any supplemental specifications provided to clarify program tasks). As would be true in a real game development scenario, you will be evaluated primarily on your codes ability to implement all of the requirements correctly. A secondary consideration, but also important, will be the structure, readability, and documentation of your code.

8 How to go about writing this code

1. Read this programming task specification carefully.
2. Download and build the project.
3. Run the base code.
4. Examine the source code. Particularly, study which variables the functions (those required for the programming task) take as arguments and compare them with what these functions are to do.
5. Write down the steps that each function should perform as simple as possible.
6. Use this plan to write the code.
7. Write the code incrementally, compiling often. Save your work!
8. Save your progress often by committing your code and pushing to github.

9. Ask questions early and often!