Lab 1

Resources

- Discussion Slides - Copy of the presentation given by the TA
- Unix Commands - A list of some unix commands you will find useful for this lab
- g++ Commands - A list of some g++ options. Not all of these will make sense to you right now. We will cover them more in depth in later labs
- More Unix - A more comprehensive list of 30 useful unix commands
- Google C++ Style Guide - Quite a lengthy guide on various areas of C++. You don't need to know it all but it is usually a good place to start. For now, look over some of the commenting and formatting sections

Idea

The goal of this lab is to get familiar with the basics of C++ syntax and the compliation process.

In-lab assignment

Preliminary

These steps will walk you through how to compile a basic Hello World program from the Unix command line. Please ask questions if you have trouble.

1. Log into your CISE account if you have not already (How are you reading this?)
2. Use the command line to create a new directory called "lab01" inside your home directory.
   `mkdir ~/lab01`
   Use the command line to make this your working directory.
   `cd ~/lab01`
3. Using your favorite text editor (emacs, gedit, etc.), create a file called lab01.cpp
4. Write a program that will print "Hello World!" to the console whenever it runs.
5. Compile this program from the command line. To do so, make sure you are in the correct directory. You can check this by typing `pwd` which stands for present working directory. You should see "/cise/homes/your_name/lab01/". You can ensure you saved your lab01.cpp file correctly by entering the `ls` command which will give you a listing of all files and folders inside your pwd. You should see "lab01.cpp" listed here. Get familiar with these commands and how they flow together.
6. To compile the program, use the following command:
   `g++ lab01.cpp`
7. If you were successful, there should not have been any output. If you had errors in your code, they will show up here.
8. When you get your code to compile correctly, you should now have a file called "a.out" in your directory. Remember, `ls` can check this for you. This is your executable file. To run it, use the following command:
   `./a.out`
9. You should see Hello World printed to your screen. Yay! If you don't (Aw!), you have some errors in your code.
10. Once you have your Hello World program working, we are going change the name of the executable from "a.out" to "hello". The way this is done is by adding the following to your compilation line
    `g++ lab01.cpp -o hello`

11. You should now have a "hello" in your directory. Run it with
    `./hello`

Requirements/Deliverables

This next set of instructions you are going to do on your own. They involve adding some logic to your program you have made. You are going to write a number guessing game. The idea is that the computer will pick a number between 1 and 100 and the user will have to guess it. When the user guesses too high or too low, the program notifies them. Program flow is as follows:

1. 2 pts - The program displays a prompt to the user explaining the rules of the game
2. 2 pts - The program picks a random number using the given random function found here
3. 3 pts - The program prompts the user to guess a number and correctly reads the users input
4. 3 pts - The program tells the user whether they guessed too high or too low
5. 3 pts - The program determines that the user enters the correct number
6. 3 pts - The program allows the user to play the game multiple times, asking them each time they guess a number right if they would like to play again. Reading in a simple character (y for yes, n for no) will suffice here
7. 2 pts - Use good programming style in your code. This includes proper structure, readability, and useful variable names
8. 2 pts - Do all of the compilation from the command line

Optional Enhancements

If you finish the game and have spare time, consider the following additions (completely optional but can be

valuable experience).

1. Modify the random function to pick a random number between a given minimum and maximum value instead of just 1 to 100
2. Allow the user to input the min and max values the program chooses from
3. Implement the above option using command line arguments
4. Each time the user makes an incorrect guess, display the possible bounds for the number as well as the high/low information. For instance, if the computer picks between 1 and 100, and the user's guess of 25 is too low, then we know the number is between 26 and 100
5. Check for valid input for the user. Ensure your program is robust to any entered data. For instance, if the user enters "abc" as a guess, your program should not crash or go into an infinite loop. It should print some sort of error message and carry on.
6. Anything else you can think of! Really, take this chance to familiarize yourself with the basic input/output and control statements of C++ as well as the unix command line. Remember: practice makes perfect!

Submission

When you finish, just raise your hand and the TA will come around to grade you according to the Requirements section. The grading will consist of the following:

- Navigate to your lab01 folder using the command line
- Compile and run your program from the command line
- TA will check off the requirements you have met

Please make sure you are ready to do these things before asking for grading. The TA will let you know the latest you can be graded based on time constraints.