CSE30 : Lab #1 - Introduction to C++

Overview

Welcome to CSE30! This lab will help you be familiar with the C++ programming language and the tools available to you on the Linux systems in the lab. In this exercise, you should do your work on the lab systems directly so to minimize any headaches in the beginning of this semester. After you are comfortable with the system and the language would be a better time for exploring your other options. The goal of this lab is to create simple programs that will display output to the console, read input from the user, use if statements and for loops in C++, and perform some simple error handling.

Note: You need to have a separate program for each of the parts of this lab. When you submit your assignment through CatCourses, make sure that ALL PARTS are included.

Getting started

In this class, we will use the Linux terminal extensively, so you should get familiar with it as much as possible (there is plenty of information online). We recommend setting up a smart directory structure to use throughout the class, but it is up to you how you setup your programs. We would setup a CSE30 directory on the Desktop, a directory for the lab (i.e. Lab_01) inside it, and a directory inside the lab directory for each part of the lab (i.e. part1, part2, etc...).

(Exercise) Tutorial in Linux

If you are not familiar with the Linux operating system or not sure how to use command lines in a Linux terminal, take a look at the tutorial from the following link during your lab hours. Otherwise, skip to the next exercise.

http://linuxcommand.org/lc3_learning_the_shell.php

(Exercise) Create – Lab_01 directory

Open a terminal and access your Desktop directory. To create a new directory, use ***mkdir*** command. Make sure you are in your Desktop directory, type ***mkdir CSE30*** and press enter in the terminal. To verify if the directory is created, type ***ls*** (lowercase L, not uppercase i) and press enter. You should see **CSE30** among a list of directories and files under your Desktop directory. Go into the newly created CSE30 directory by typing ***cd CSE30*** and press enter. Now, create a new directory named **Lab_01** under your current (CSE30) directory. Go into the newly created Lab_01 directory and start working on your lab assignment.

(Exercise) Create – main.cpp

Make sure you are in your Lab_01 directory, type ***nano main.cpp*** and press enter. The **nano** text editor will be displayed (see the *Working from a Windows/an Apple Computer* guide on how to use nano). Feel free to use any other text editors (vi, gedit, etc.) of your choice.
Note: If you see "***bash: nano: command not found***" after typing ***nano main.cpp***, it means that nano was not installed in your machine. You will need to install it by typing "***sudo apt-get install nano***" and wait for the computer to finish the installation. Ask your TA if you are not sure about this.

Copy the following code to your file:

```
#include <iostream>
using namespace std;
int main()
{
    cout << “Welcome to CSE030!” << endl;
    return 0;
}
```

To save the file, press CTRL + x (if you use nano), it will ask if you want to save the file. Type y and it will confirm with you the file name. Press enter to save the file and exit nano.

Congratulations, you've just written your first C++ program! The first line, **#include<iostream>**, includes a library that allows you to use simple Input/Output (I/O) operations. In this program, it allows you to use the **cout** statement. Everything inside the **iostream** library uses the **std** namespace, which would require you to write **std::cout** every time you would want to print something to the screen. In order to avoid this waste of time, we use the statement, in the second line, **using namespace std**, so that **cout** can be written directly (as opposed to having to write **std::cout**). In the next line, **int main()**, we start the main function, which is a mandatory function for any C++ program. This is the function that first executes when the program is launched. The contents of the function's code need to be encompassed by curly braces (lines 4 and 7). On line 5, we are printing the text **Welcome to CSE030!** to the screen. The **endl** statement stands for **end of line** and will put the next print statement on the next line. Finally, the last line returns from the main function, which will stop execution of the main program.

Once you have created this file and understood its content, you want to compile the source file (main.cpp) into an executable so that you can run it on your computer. You will do this from the command line by running:

g++ <source> -o <executable>

where **g++** is a program (already installed on your Linux system) that compiles C++ source files, **<source>** is the source file for your program (main.cpp in this example), **-o** tells the compiler that you want to give the executable its own name, and **<executable>** is the name you want to give your program. In the terminal (still under Lab_01 directory), compile your first program by typing ***g++ main.cpp -o main***

Assuming that your program compiled successfully (i.e. no errors were found), you can run your program by typing **./main** in the terminal.

(Assessment) Logic Check

It is crucial that you gain a thorough understanding of creating and compiling programs. Answer the following questions (either try to answer them now or in your own time, but make sure you can answer them):

1. How would you change the executable name from **main** to **cselab**?
2. What happens when you compile using **g++ <source>** only?
3. How can you remove the line using **namespace std**; from your program and get it to compile.
4. Add comments to your program.
5. Play around with "cout" statements and make sure you understand what "endl" does.

Write your answers to questions 1-3 in a text file for submission.

(Exercise) Create – hello1.cpp

In this part of the lab, you should create a program that outputs **Hello World!** to the screen. Please note that this program should be very similar to the main.cpp. When you run your program, it should display the following output:

```
Hello World!
```

(Exercise) Create – hello2.cpp

In this part of the lab, we will introduce user input. More specifically, we want to welcome a specific student to the CSE030 class. In order to do so, you should start by asking the user for his name, by outputting **Please enter your name:** to the screen. You will then read the student's name, which could be his first name, full name, both, or any other string that is typed before pressing enter (see examples in the next sections). After you have read and stored the student's name, you will output **Welcome to CSE030, <name>!** where **<name>** is the name that the user inputted. Refer to the online tutorial on how to use ***string*** (under *Variables. Data types*. section) and ***user input*** (under *Basic Input/Output section*).

Web address of the tutorial:

http://www.cplusplus.com/doc/tutorial/

Here are the examples of the output (input is in italic and bold):

Please enter your name: ***Name***
Welcome to CSE030, Name!

Please enter your name: ***First_Last***
Welcome to CSE030, First_Last!

(Exercise) Create – punishment1.cpp

A common punishment for school children is to write out the same sentence multiple times. Write a C++ program (**punishment1.cpp**) that will write out the following sentence a number of times inputted by the user: **I will always use object oriented programming.** The program will first ask the user to input the number of times the sentence should be written, by outputting the following sentence to the screen: **Enter the number of lines for the punishment:** . You should check to make sure that the value entered is correct (think about what would constitute an incorrect value). If the value entered is incorrect, output the following sentence to the screen: **You entered an incorrect value for the number of lines!** and stop the program. If a correct value was entered, display the sentence **I will always use object oriented programming.** the number of times inputted by the user. You may want to read the online tutorial on how to use loops and if statements (under *Control Structures* section)

Here are the examples of the output (input is in italic and bold):

Enter the number of lines for the punishment: ***5***
I will always use object oriented programming. I will always use object oriented programming. I will always use object oriented programming. I will always use object oriented programming. I will always use object oriented programming.

Enter the number of lines for the punishment: ***-5***
You entered an incorrect value for the number of lines!

(Exercise) Create – punishment2.cpp

Since we want to make the punishment more realistic, we will add a typo in one of the lines of the punishment. This is very similar to what you have already done in punishment1.cpp, so it would be a good idea to copy your punishment1.cpp and start from there. Your new program (punishment2.cpp) will ask the user for the number of lines for the punishment: **Enter the number of lines for the**

punishment: . Once again, if an incorrect value has been entered, your program should output **You entered an incorrect value for the number of lines!** and stop executing (up to this point, you should not have to change anything from your last exercise. Next, the program will ask on what line a typo should be made by outputting **Enter the line for which we want to make a typo:** to the screen. Once again, you should check that the value entered by the user is correct (think about what would constitute an incorrect value). If the value is incorrect, display **You entered an incorrect value for the line typo!** and stop program execution. If both inputs are correct, you should then display the punishment sentence the correct number of times (**I will always use object oriented programming.**), making sure to change it to **I will always use object oriented programing.** (the typo) for the line number defined by the user/input.

Here are the examples of the output (input is in italic and bold):

Enter the number of lines for the punishment: ***4***
Enter the line for which we want to make a typo: ***1***
I will always use object oriented programing. I will always use object oriented programming. I will always use object oriented programming. I will always use object oriented programming.

Enter the number of lines for the punishment: ***6***
Enter the line for which we want to make a typo: ***3***
I will always use object oriented programming. I will always use object oriented programming. I will always use object oriented programing. I will always use object oriented programming. I will always use object oriented programming. I will always use object oriented programming.

Enter the number of lines for the punishment: ***-8***
You entered an incorrect value for the number of lines!

Enter the number of lines for the punishment: ***3***
Enter the line for which we want to make a typo: ***-2***
You entered an incorrect value for the line typo!

What to hand in

When you are done with this lab assignment, you are ready to submit your work. Make sure you have included the following ***before*** you press Submit:

- Your main.cpp, hello1.cpp, hello2.cpp, punishment1.cpp, punishment2.cpp, answers to Assessment questions (1-3) in a text file, and a list of Collaborators.