CSS 332: Programming Issues with Object-Oriented Languages

Lab1: Type conversion, references, and file I/O issues

For this lab, we will be using the following code to experiment with the basics of the C++ toolchain, primitives, type conversion, and pass by value/pass by reference. Please pick *one* of the following tasks to complete. There is no need for you to rush to do everything; instead, focus on whatever you're least comfortable with and try to get to the point where you now feel good about that concept. Of course, you should feel free to do more than one thing if you have time.

`BuggySoftware.CPP`

As a coding refresher, start with `BuggySoftware.cpp.` Follow the instructions embedded in code. Once you have debugged this file, you can then proceed with the following instructions.

`Primitive1.CPP` and `Primitive2.CPP`:

We have two programs to explore numeric primitives. `Primitive1.cpp` code is very simple. The `Primtive2.cpp` examines some aspects of implicit and explicit type conversion of primitives, in particular between integer and floating point types. Study the code, follow the instructions embedded in code, and be sure you understand how it works. For this section y**our job is to:**

1. Write a program to clearly and definitively show how the various conversion methods from floating point to integers handle floating point numbers halfway between integers, i.e., which end in ".5".

2. To avoid bias in rounding, floating point numbers that end in ".5" should be rounded to even numbers. This will result in rounding up half the time and rounding down the other half, and so there won't be a systematic bias in rounding. Write a function that rounds in this manner and write a program that demonstrates that it works properly.

`ReadNumber.CPP`

This exercise lets us experiment with compiling and executing our code in one file or implement our software in both a .cpp file and an .h file.

1. Download `ReadNumber.cpp` file.
2. The instructions for this section are embedded in the code. Please follow each step carefully and be sure your final version compiles and executes correctly.

ReferenceDemo.CPP

In this section, we will look at references and function calling using `ReferenceDemo.cpp`.

1. Download and try to compile the file.

2. Why doesn't it compile? I'm not just looking for which line(s) have the problem, but rather what is wrong with those line(s).

3. What is happening when onePlus() is called? In other words, what exactly does the function receive as an argument? Why isn't the value of the variable in main() altered?

4. Conversely, consider the call to increment(). What is the argument here and why *is* the value of the variable in main() changed?

5. Finally, consider the code related to value(). First of all, make sure you understand whatPair is and what the declaration of pairs means.

6. Next, consider a call to value(). What does the function receive as an argument? What is the meaning of const in this context?

7. What does value() return? How can it be legal and what does it mean to apply the ++operator to its return value?