CS170 Week 2 Lecture 2 Notes

Disclaimer: These notes are meant to be read in parallel with Professor Mead's online notes if you have missed class. Topics will not be covered extensively. You will only see the minor details Prof. Mead spoke about which were not thoroughly expressed in his topic notes.

Pre-Class Mentions/Q&A

- ➔ The lab (which will be discussed at the end of class) will be due on Tuesday, January 19th at midnight.

Namespaces (Continued)

Namespace Aliases:
Local variables are okay to be short while global variables should be long(enough)/descriptive.
These aliases may only be used preceeding the scope resolution operator.

Using Directives:
Don't use “using namespace” very often
The purpose of namespaces: to not pollute the file with all these symbol names which are in the namespace. After using the using directive, it is no longer necessary to qualify the members of the namespace with the scope resolution operator (in the scope which the directive lies in).

DON'T DO THIS: Using namespace std; //at the top of the file
(especially don't do this in a header file)

No warnings are emitted upon trying to access a variable that may be in more than one current namespace.

The Standard Namespace (and using declarations):
Using declarations are more effective than writing “using namespace std;” at the beginning of your code.
The std namespace is a part of iostream.
The nice thing about C++ : you can overload operators to do whatever you want.
To access nested namespaces, we use the scope resolution operator in the same way that we accessed members of a struct in C.
For example: *“ Digipen::CS170::Assignment1::Sort(); “*
Digipen is the top level namespace. Assignment1 is a nested namespace within Digipen.
You could possibly set an alias to make this less verbose: *“ D1701 = Digipen::CS170::Assignment1; “*
Boost is a type of training grounds for C++. When functions do well in Boost, they may become part of the std namespace.

Simple I/O:

Overview of Formatted Output:
For now, objects are an instantiation of a struct, rather than a function.
In regards to cin and cout: c stands for console.
Remember, we no longer need to specify the data type when printing with cout.
Insertion operator is overloaded for each type. The operand on the right of an insertion operator is what the compiler looks at to decide what to print.
User defined types: to print these, you have to tell the compiler how to do so.

Lab #2 Info/Post Lecture Mentions

- ➔ Due at midnight on Tuesday January 19, 2016
- ➔ There will be no practice labs this semester due to a lack of student interest in the past
- ➔ For the lab, you need to look at all of the given files to see what each is dependent upon
- ➔ This is a cozy 30 minute task
- ➔ Going forward, all of the assignments will be using doxygen