CS170 Week 5 Lecture 1 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. Here, you will only see the minor details Prof. Mead spoke about which were not on his topic notes.

Pre-Class Mentions/Q&A

- ➔ Download EVEditor for an easier (better-looking) way to change your Windows system environment variables
- ➔ This class is unique in that we are encouraged to learn how to test software, rather than be given all explicit tests
- ➔ 50% of the people who turned in Assignment #1 had perfect Doxygen and DrMemory
- ➔ Valgrind and DrMemory detect the same issues, either are okay to use
- ➔ Today, Mead will show us how to run Valgrind (only if you have Mac, Linux, Virtual machine) and DrMemory for those with Windows
- ➔ Visual Leak Detector is easy to use and free (student suggested)
- ➔ Dr Memory works with MinGW (student suggested)
- ➔ Always running your program under a memory debugger as you develop (and with warnings turned up all the way) is the way to go
- ➔ If you aren't compiling with debug information ON (in Valgrind) it will only give you an address, no line number information etc.
- ➔ Printing out your index is a good way to find out where you overwrote memory
- ➔ The biggest problem in Assignment #1 was when people wrote into the boat array because boat ID starts at 1 and not 0
- ➔ The good news is that there's a new assignment

Classes Continued

Arrays of Objects
To construct an object, a constructor must be called. Period. Always.
If you don't have one, the compiler will give you one. Same goes for destructors.
User defined types require default constructors unless you explicitly initialize every member.
This ends the introduction to classes. We now have everything we need for turning WarBoats into an object oriented version for Assignment #2.

More Classes
We will be able to write our own array class to add arrays.
The main goal of overloading operators is to make tasks very obvious and easier to understand.

Overloading the binary + Operator
You cannot overload for built in types.
You cannot change precedence or associativity of any operators.
Typically you just have the function in the header file.

Overloading More Operators in the StopWatch Class
Every operator is unique, so if you overload the '+' operator, it will not affect the '+=' or '++' operators.
Adding more than two *StopWatch*es works because operator+ returns a *StopWatch* type.
Every time you hear the word “convert” or “copy” (mostly for user defined types), you should think about how expensive the operation is.

Overloaded Operators as Member Functions
There's only one operator now because the left hand side operator is implied.
(To see where the above note applies, Ctrl+f: *StopWatch operator+(const StopWatch& rhs) const;*)

~15 min break~

The friend Keyword
Functions that are friends of the class have access to the privates.
We can list as many functions as we want as friends of the class.
There's no way to copy a stream so we pass them by pointers (using references).
This is where references begin to shine, in overloaded functions.
Friend status is one way only. Friends do not affect each other.

Automatic Conversions and User-Defined Types
A lot of the automatic converting was put into place for backwards compatibility.
There will possibly be quiz questions on ambiguous conversions.
We will often overload the subscript operator.
There is a keyword called explicit, which is how we will tell the compiler to suppress automatic conversions.
Normally, when you have one argument constructors (conversion constructors) you should always mark them explicit.
With C++11 you can put the explicit keyword in front of your conversion functions.
We won't use C++11 until CS225 / CS280.

Overloading Operators with Side-Effects
With built in types, we won't measure performance.
For user defined types, this becomes a big deal especially when it comes to operators with side-effects.
Some of the objects we use will become complicated and take up lots of space.
The cost of constructing those objects is something we now need to think about.

Member, Friend, or Non-member?
Friends will generally only be used for the overloaded output operator.
Quiz/Midterm questions could possibly ask about whether functions are friend or member functions.
We'll do an assignment at some point where we do a lot of operator overloading.

Restrictions on Operator Overloading
“Keep in mind, how would the ints react to this?”
(Don't do anything that isn't unintuitive or easy to follow).

Assignment #2 Info/Q&A

- ➔ The first parameter to all of the functions was the ocean, but now the ocean is an object
- ➔ Outputs should all be exactly identical
- ➔ 85-90% of the code is done
- ➔ You aren't really adding any code, so it shouldn't take more than two hours (if you had the right output in the last assignment)
- ➔ Assignments are expected to be in the return folder on Monday
- ➔ Don't just copy the Doxygen comments from the last assignment because many comments will be changing
- ➔ We'll be turning header files as well as cpp files pretty consistently from now on

To better understand the topics in CS170:

- ➔ Before next Friday, be sure to review the online notes
- ➔ Do the textbook assigned homeworks
- ➔ Do the assigned reading in the textbook