COMP 2160 - Programming Practices

Calendar Description: Introduction to issues involved in "real-world" computing. Topics will include memory management, debugging, compilation, performance, and good programming practices (Lab required).
Prerequisite: COMP 1020.
Recommended for: COMP 3290 and COMP 3430
This course is a prerequisite for: COMP 2150, COMP 2280 and COMP 4430

Outline

1. Introduction to Unix and C (3-4 weeks)
   Unix is 1 class but is shown via demos throughout the year. C if simply doing equivalent Java stuff from 1010/1020 (up to pointers for ADTs). Introduction of the notion of stack versus heap, discussion of the idea of process context.
2. Testing (2 weeks)
   Design by contract, assertions, scaffolding and TDD (intro only -- no xUnit).
3. Separate Compilation and Modular Design (1 week)
   Header files and compilation issues. The use of make for compiling – with a basic introduction to IDEs for projects and debugging (shown with debugging). How to break programs into modules using ADTs (and a brief introduction to coupling and cohesion).
4. Debugging, Profiling and Optimization (1 week)
   This is actually distributed within separate compilation and memory management as they end up being part of 3 labs: debugging (with gdb), profiling (with gprof) for speed optimization and storage optimization. Each requires a class for lab preparations. We then use the techniques (especially debugging) for working on pointers and memory management.
5. Pointers and Memory Management (2 weeks)
   A look at pointer/array equivalencies, pointer arithmetic and the use of void pointers for generic memory management. Includes a discussion on the notion of heap management (as an array of bytes with pointers to individual objects) and the basics of garbage collection.
6. C++ (1 week)
   A basic introduction to the differences between C and C++. How to make basic classes (e.g. a linked list) and standard features such as templates (for vectors, queues, etc) strings and I/O. Absolutely ***no*** OO stuff is included!
7. Advanced C (2 weeks)
   Building shared libraries, using getopt, advanced storage classes (register and volatile), etc. End with a discussion on building objects in C: unions for polymorphism and function pointers for embedding routines in structs and/or building function tables (ala vtables in C++ or for running state machines – which they won't see until 2280...).

Text: Jon Bentley, *Programming Pearls, Second Edition,* Addison Wesley, 2000.
Recommended Text: Brian Kernighan and Dennis Ritchie, *The C Programming Language, Second Edition,* Prentice Hall, 1988.