EECS 211(B) – Object-Oriented Programming with C++

Winter 2012 Quarter

Instructor: Larry Henschen

Contact: henschen@eecs.northwestern.edu

Class meets: MTW F 2:00-2:50PM – at M345, Tech.

Office hours: MTW F if I'm in an not busy or by appointment

Detailed TA info: TBA

Course description: The main objective of this course is to expose the students to the foundations of the Object-Oriented programming model and make them proficient in the C++ language, as a particular example of the languages from that paradigm. The course will attempt to strike a balance between the two fundamental aspects of studying any field: **– *depth,*** via studying both the syntax and semantics of the important programming concepts, along with providing programming assignments and tests related to them; and **– *breadth***, bringing to awareness the "bigger picture" of good programming style and tying it with problem-solving, along with designing relevant test-cases and developing a set of broader-skills to come handy in the subsequent courses focusing on software design/engineering. Part of the course will provide a brief introduction to C/C++ programming in Unix/Linux environments, thereby introducing the students to some aspects that will be subsequently studied in greater detail in courses like introduction to systems programming, operating systems, networking, etc.

I. Prerequisites: CS 110 or knowledge of any programming language. The course begins with a review of data types and basic flow-of-control, so your prior knowledge of a programming language need not be deep and need not be specific to C/C++.

II. Required text: "Big *C++"*, by Cay Horstmann and Timothy Budd (publisher: Wiley)

> NOTE: In Section 20 the C++ book is used for reference. In fact, any up-to-date C++ book is probably adequate. Also, there are plenty of examples available on (various forums on) the web that you should feel free to consult.

III. Course Outcomes: After finishing the course, the students should be able to:

a. Use (most of) the syntactic elements of the C++ programming language in a fluent manner.
b. Develop well organized object-oriented code in C++ code for solving problems of interest in various applications' settings.
c. Provide their code with various test-cases and "guards" for the purpose of preventing unwanted behavior.

d. Understand code of existing applications and be able to add new features, correct existing errors and incorporate their novel solutions. An implicit benefit is increasing the broad flexibility in terms of being comfortable with adjusting to new programming environments.

IV. Tentative Course Outline:

Week 1: Introduction to programming, the stages of program development, introduction to a typical Integrated Development Environment (IDE).

Week 2: Data types, variables and constants, arithmetic and comparison operators and their precedences, expressions and assignment.
Simple I/O with cin and cout.
Introduction to flow of control – sequential, test-and-branch, looping, nesting control structures.

Week 3: Flow of control continued.
Functions, call by value/address/reference, return values. Scope of variables. Call stack and stack frames. Arrays and pointers.

Week 4: Arrays and pointers.
Introduction to classes - class declarations, public/protected/private members, data members and function members, constructors. Interface vs. implementation.

QUIZ

Week 5: Classes continued. Dynamic creation of objects – **new** and **delete** and the heap. Deep vs. shallow structure. Copy constructors. Destructors. Operator overloading.

Week 6: Review and catch up.

MIDTERM EXAM

Week 7: Introduction to derived classes and inheritance. Virtual functions and polymorphism. Pure virtual classes and abstract data types. ISA vs. HASA relationship.

The exact material in the last three weeks depends to some extent on our progress in the first seven weeks but may include:

- Exception handling
- Brief introduction to UNIX
- Dynamic data structures – linked lists and trees
- Features of other object-oriented languages, like JAVA and C#

V. Grading

Your grades will be based on:

- 2-3 homeworks – 2.8%
- 1 Quiz – 7.2%
- 8 Projects – a 40% (see the corresponding paragraph below)
- Midterm – 25%
- Final – 25%

NOTE: the distribution given above is approximate and may be subject to some *very minor* changes. The firm policy will be announced during the last week of classes.

Programming Assignments: There will be eight programming assignments. The first two introduce the students to basic C++ programming and the use of a modern, integrated development environment (IDE). The remaining six assignments form a single medium scale application. Each assignment adds a piece to the overall project. The various assignments in the project cover common programming applications such as character processing, linked data structures, classes and inheritance, etc. Sample applications from previous quarters include internet message passing, computer memory management system, sensor network simulator, and XML parser.

Students generally do the programming assignments on their laptops, although the course does include demonstrations and lectures on using UNIX for programming. A separate document contains information on how to obtain free software for C++ software development.

Awareness, Academic Responsibilities and Closing Remarks:

Please be advised that it is each student's *individual responsibility* to keep him/herself up-to-date with the announcements *made in class, distributed via email,* or *otherwise posted*. Although you are encouraged to discuss class-related issues with your classmates, it is your individual responsibility to ensure that the work is done individually. For example, you are encouraged to discuss the algorithmic aspects of solving a particular programming assignment, and even the high-level design approach. The source code that you will submit, however, must be typed *individually* in its entirety. The policies for cheating are well-defined and there will be no exceptions made for any excuse whatsoever – if caught cheating (e.g., copying on an exam, borrowing someone else's code as well as allowing someone to borrow your code), you will automatically fail the class and face a possible expulsion from the University. In addition, notwithstanding our willingness to be understanding for the students' commitments and time-constraints, please do not attempt to obtain an incomplete grade for the course based solely on your poor performance – it is against the University regulations. Lastly, please note that a substantial part of your grade is based on the programming assignments. Hence, in addition to keeping yourself up-to-date with the materials lectured, you should also make it a habit to start working on the programs as early as possible. You should not allow yourself to fall behind with the topics, as the new ones will be building incrementally upon the older ones, and it will be very hard to catch up. Plan your time wisely and good luck.