Syllabus for MATH 6767 Fall, 2013

Instructor Information

Stephen Demko's office is Skiles 265.
Phone Number: 404-894-2713.
e-mail: demko@math.gatech.edu
web site: www.math.gatech.edu/~demko
Office Hours: Tuesday and Thursday *2:30 to 4:00* and by appointment.

Math 6767 Design & Implementation of Systems to Support Computational Finance meets Tuesday and Thursday from 12:05 to 1:25 in Room 1 of the Weber SST3 building.

PREREQUISITES AND GENERAL CONTENTS OF THE COURSE

The student need not have had previous exposure to C++; however, some exposure to a computer language will be very helpful. The formal description of the course can be found here.

After introducing the core features of the language, including the standard template library, we will introduce the facilities for abstraction through template functions and inheritance and polymorphism. However, the goal is not to become an expert in C++ (although a talented, dedicated person can go a long way alone starting from this course), but to become acquainted with it and to become able by your own self-study to deepen your knowledge of it and improve your abilities. The hope is that the students will become able to learn how to advance their skill in the language if the opportunity or need arises in life beyond the current semester.

We will cover in a non-rigorous way the development of some financial models and the underlying mathematics. The students will implement in the C++ language several models from quantitative finance. This development will preview, reinforce, or review (depending on your experience) topics that appear in other courses in the QCF program. Several (about 4 or 5) projects will be assigned. These are meant to provide the student with the opportunity to exercise specific features of the language as well as to give the student an opportunity to explore the underlying models.

A word about the typical student: from past experience, I can say that there is no typical student. Student backgrounds are very diverse. Some have years of experience in programming in an industrial or research environment and some have almost no experience in programming or have forgotten much of what they knew. Some have taken courses in Quantitative Finance or Majored in Mathematical Finance and some have little exposure to the area. My goal is to try to serve all students enrolled. We will start with writing the simplest C/C++ programs and aim to finish with designing and writing object oriented programs in C++. The financial applications are a major unifying point of the course.

EXPECTATION and ADVICE The way one learns to program is to sit down with a computer and write programs (and, of course, run them and verify that they are producing the right sort of results). Initially, one can start with a program that works and play with it by modifying various parts or extending it to do more things. This is generally a self-checking process in that if the code compiles and runs, the checking of the results is generally simple because one can often calculate special cases by hand or in a spread sheet. If your programming skills are weak, it is important to start this process on the first day of class.

RESOURCES FOR MATH 6767

Text Books

The first and third books below should be in the book store. The first book, Koenig and Moo is "required". It provides a pretty quick introduction to C++. The second book is available in e-form for free; there is no reason not to have a link to it. The third book, by Capinski and Zastawnial, is "recommended". We will use it mainly as a *potential* source for descriptions of two of the main models we will implement: the Binomial Model and the Monte Carlo Model.

Accelerated C++: Practical Programming by Example by Andrew Koenig and Barbara E. Moo. You can download code examples from the book's web site and see reviews at Amazon.

Thinking in C++: Introduction to Standard C++, Volume One (2nd Edition) by Bruce Eckel. An e-copy can be found at many author authorized web sites, for example this one. You can read

reviews at Amazon.
We will refer to this book for certain examples and illustrations. You should set up a link to it or download it.

The book of Koenig and Moo gives a basic introduction to the language. I expect to use it throughout most of the course. There is no reason not to have an e-copy of Eckel's book. I will also use parts of it and make assignments from it.

Numerical Methods in Finance with C++ by Maciej J. Capinski and Tomasz Zastawniak.
You can download errata and code examples from the book's web site and see reviews at Amazon.

Additional Reference Books

C++ Design Patterns and Derivatives Pricing by M. S. Joshi.
I have used this book in the past in the second half of the term. It assumes familiarity with C++ and presents some useful constructions in Object Oriented Programming. You can get updates and code from the author's website and see reviews at Amazon.

The C++ Standard Library: A Tutorial and Reference by Nicolai M. Josuttis.
Supplementary materials and code are available at the author's web site and reviews can be found at Amazon. This is an excellent reference for the C++ library.

Effective C++: 55 Specific Ways to Improve Your Programs and Designs (3rd Edition) by Scott Meyers.
This is somewhat advanced. See reviews at Amazon.

Effective STL: 50 Specific Ways to Improve Your Use of the Standard Template Library by Scott Meyers.
This is somewhat advanced. See reviews at Amazon.

The book of Koenig and Moo gives a basic introduction to the language. I expect to use it throughout most of the course. There is no reason not to have an e-copy of Eckel's book. I will also use parts of it and make assignments from it.

Internet Resources

cplusplus.com is a good online reference; it has a tutorial as well as a very good reference section which **we will use often; you should bookmark it now.** If you are new to C++, you should consider starting to work through the tutorial.

The boost libraries are peer reviewed C++ libraries. We will probably use some of them.

Computing Environment

You will have to write computer programs that I will test in the `g++` environment. One way to do this is to do the computing on a Unix or Linux computer by logging into such a computer from your PC via the program SecureCRT. In this mode of operating, the program WinSCP is also useful. Both are available from Office of Information Technology (OIT). In particular *SecureCRT* is a terminal emulation program that will enable Windows users to log into network computers, and *WinSCP* allows one to view the contents of a network computer as thought it were a disk drive on your own machine. Mac users can use the Mac's terminal program to access a network computer via `ssh`. Ask about this approach if you are interested. (In fact, Mac users should be able to install gcc/g++ on their computer via XTools. You will have to become a Mac Developer to get the software.) In order to view the network drive as a disk drive, Mac users can use the program *Fugu* from the above OIT link.

A better approach for PC users is probably to download Code::Blocks. This is a free development system that includes the g++ compiler. I will often use it in class. It has many more features than we need, but it works. If you have Visual Studio, you can use it; but YOU have to make sure that your code is not Microsoft specific.

COMMUNICATIONS

I will send e-mail concerning the class (e.g., announcements, notes) to your GT-Account e-mail address, so please check this account often. Please use the address demko@math.gatech.edu when you send e-mail to me. Finally, if you need help, especially early in the term, please let me or our TA know.

ASSESSMENT

This is subject to change. We will have two in class tests. The first one will be in the first week of October or the last week of September. The second test will probably be the Final Exam. **NOTE:**Your final letter grade can be no more than one letter grade higher than your scores on these tests. (For example, if you have a C-average on in-class tests, the highest grade you can get for the course is B.) I expect to assign 3 or 4 major projects of equal weight.

Concerning the projects. I expect almost everyone to be able to write correct programs and draw sensible conclusions. As a rule this is not enough to get a distinguished grade (like A). The work of an A-student is typically distinguished by some of the following: especially insightful analyses of the problen at hand, elegant coding, investigations beyond the original scope of the project (but not busy work). In short the work of an A-student will show a strong command of the math, finance, and computing as well as a scientific-like attitude to the investigation at hand.

HONOR CODE

It is very important that every student understand Georgia Tech's Honor Code and its addendum for graduate students.

All students are bound by the Georgia Tech Honor Code and will be required to sign a statement agreeing to accept their responsibilites under the code. Note that in addition to the basic code, graduate students are governed by the Graduate Addendum as well as by the guidance given in the next paragraph.

There are several ways that what happens in the classroom is complemented by your own experience. Working environments, other classes and readings, and discussions with your classmates or with others who have taken courses like this one are perhaps the most common. These are all good things and they are to be encouraged. However, when written assignments are due (in the form of essays, papers, or computer projects), it is required that what you hand in is your own work. The idea is that your evaluation be based on your performance and not that of another. Some tasks have only a small, finite number of natural ways of being carried out and these are well documented in many texts and web-sites. You may make use of these resources, but you may not violate any license, copyright, or patent privileges. If you use any source code that you did not write, you must give a proper acknowledgement. The key thing is that you understand the computer code you are using. If it has particularly elegant features, you should be able to explain them. If you wrote these features, you should document it. **The bottom line is that you must not only understand what you are doing, but be prepared to explain it, if asked. There may be short written quizzes or interviews required of you after a project is due.** It is expected that, unless otherwise specified when an assignment is given, all written assignments, quizzes, and tests, turned in by students will be the work of the student alone (or the student’s work group in the case of group assignments).