New York University Tandon School of Engineering

Department of Finance and Risk Engineering
FRE6811 – Financial Software Laboratory – C++ (1.5 Units)
Fall 2021
Professor Jason Yarmish
Tuesdays 6:00pm-8:41pm; 11/02/21 – 12/21/21

To contact professor: Yarmish@nyu.edu

Course Pre-requisite: Graduate Standing

Course Description: This course will teach the fundamentals of procedural programming using C++. It is intended for those with little programming background, although prior programming experience will make it easier. The course emphasizes writing programs and becoming proficient with the language and its use in financial applications.

With C++, we will illustrate the core concepts; what it takes to program well and to really understand how the computer works behind the scenes. As opposed to learning a software package, mastering C++ is an experience which will enable mastery of almost any other programming language.

Aside from the fact that an understanding of C++ helps in programming almost every other language, some of the most famous applications are written in C++ such as most major applications of Adobe systems, many Google applications, Mozilla Firefox, Safari, Chrome, MySQL, Bloomberg foundational libraries, and virtually all of windows applications including Word, Excel and the Microsoft operating system - in fact most existing operating systems. Facebook, Youtube, amazon and Paypal all have major C++ components. The list (gathered from a simple google search) goes on...

We will learn: Writing algorithms, debugging and testing programs, loops and conditional control structures, functions and parameter passing, arrays, memory allocation, program tracing, classes and more. Examples of programming applications include implied volatility and Monte Carlo simulation.

By the end of this course, you should be able to:

- Understand and use the basic programming constructs of C++
- Manipulate various C++ data types.
- Isolate and fix common errors in C++ programs
- Write, read and understand C++ programs on paper using the skills learned
- Understand how C++ may be used in financial problem solving

Coding Environment: Software developers generally work in an integrated development environment (IDE), which have built-in tools for editing, compiling, running, and debugging programs. We will use the IDE known as Code::Blocks, which is freely available. Click here to go to the download page for Code::Blocks. Within installation, if asked, please choose MinGW for the compiler.

Course Structure: Lectures, lab, discussion, course readings, programming assignments, program critique/peer review, hand written exams. Our online classroom contains all class content. (Additional Codility Challenges and online discussion among students are available to those who want wish to challenge themselves and their skill as they learn.)

Readings: There is no textbook for this course, however you may find some of the following online materials helpful. Required readings will come from these sources as well:

C++ Language Tutorial
Wikiversity: Introduction to C++
Learn about C++ Programming

Requirements: Readings, attendance, class participation, assignments, exams.

Assessment: There will be assignments/projects at the end of each lecture. You are encouraged to collaborate, but any code and write-ups you hand in must be your own. Plagiarism will not be tolerated. Review and critique of your fellow students' code is required after the completion of each assignment. There will be two exams and a final.

Grading Policy

Exam 1:	25%
Exam 2:	25%
Final :	30%
Assignments/Projects:	20%

<u>Course Overview</u>:

The lecture notes produced in class will be provided after each class.

Please review the programs done in class. Make sure you can go through them line by line and "trace" the programs as we did in class, knowing the value of each variable and the output on the screen at any point in the program.

Week	Topics
1	Overview from source code to executable file Structure of compiler Standard input/output Variables/ The assignment operator/ Comparison operators Concept of Data Types/ basic types (int, float, bool…) Implicit type conversion Introduction to escape characters/commenting/integer division Type Casting Operator Iteration statements: **for** and **while** loops and their relationship Increase and decrease operators/ Compound assignment operators
2	Break statement Selection statements: **switch** and **if-else** and their relationship Logical operators (&&, ||, …)/ Short circuit evaluation Arrays (declaring/accessing/ manipulating/ structure in memory…) More on nested loops/ Pattern building with loops Introduction to scope / Introduction to namespaces Return value of an assignment Type mismatching Chaining input/output streams
3	Scope and Stack Functions (Prototypes and Definition) Using libraries (example using library functions to estimate π using Monte Carlo simulation) More on creating your own functions with what you know (Basic Average, Max, Present Value, Black Scholes and Implied Volatility functions) **void** type / **int&** type Recursion **Exam1**
4	Pass by reference vs Pass by value Pass by array Preprocessor definitions vs const Efficiency considerations and const-references Two-dimensional arrays/ higher-dimensional arrays Data structures Introduction to file streams Introduction to pointers, the address operator and the dereference operator

5	Arrays & averaging averages (an instance of Simpson's Paradox) File streams continued Standard input/output vs ifstream/ ofstream Passing streams to functions Output formatting Potential to overwrite library functions The Ascii Table Nesting functions Pointers' relationship with arrays and order of operations **Exam2**
6	Introduction to modulus arithmetic/ The C++ modulus operator uses in creating Fibonacci sequence and with bool, functions and testing multiples Conditional/ Ternary Operator Character Arrays String Class (comparisons/ functions/ manipulation/ find, insert, erase, replace…) getline vs cin/ String stream Controlling text files
7	Function overloading Default function arguments More on pointers/ Double pointers Introduction to the heap/ Dynamic Memory Introduction to classes/ public and private variables and functions set and get functions/ writing operators/ separating code into .h, .cpp files Templates **Final**

Moses Center Statement of Disability

If you are student with a disability who is requesting accommodations, please contact New York University's Moses Center for Students with Disabilities (CSD) at 212-998-4980 or mosescsd@nyu.edu. You must be registered with CSD to receive accommodations. Information about the Moses Center can be found at www.nyu.edu/csd. The Moses Center is located at 726 Broadway on the 2nd floor.

Inclusion statement

New York University values an inclusive and equitable environment for all our students. We hope to foster a sense of community in our class and consider it a place where all individuals will be treated with respect. The needs of all students will be addressed both in and out of class. Note that the diversity that students bring to this class is a resource, strength, and benefit. As with all class questions or issues, if you feel these standards are not being upheld, please let me know. Looking forward to learning together this semester.

NYU School of Engineering Policies and Procedures on Academic Misconduct

A. Introduction: The School of Engineering encourages academic excellence in an environment that promotes honesty, integrity, and fairness, and students at the School of Engineering are expected to exhibit those qualities in their academic work. It is through the process of submitting their own work and receiving honest feedback on that work that students may progress academically. Any act of academic dishonesty is seen as an attack upon the School and will not be tolerated. Furthermore, those who breach the School's rules on academic integrity will be sanctioned under this Policy. Students are responsible for familiarizing themselves with the School's Policy on Academic Misconduct.

B. Definition: Academic dishonesty may include misrepresentation, deception, dishonesty, or any act of falsification committed by a student to influence a grade or other academic evaluation. Academic dishonesty also includes intentionally damaging the academic work of others or assisting other students in acts of dishonesty. Common examples of academically dishonest behavior include, but are not limited to, the following:

    1. Cheating: intentionally using or attempting to use unauthorized notes, books, electronic media, or electronic communications in an exam; talking with fellow students or looking at another person's work during an exam; submitting work prepared in advance for an in-class examination; having someone take an exam for you or taking an exam for someone else; violating other rules governing the administration of examinations.
    2. Fabrication: including but not limited to, falsifying experimental data and/or citations.
    3. Plagiarism: intentionally or knowingly representing the words or ideas of another as one's own in any academic exercise; failure to attribute direct quotations, paraphrases, or borrowed facts or information.
    4. Unauthorized collaboration: working together on work that was meant to be done individually.
    5. Duplicating work: presenting for grading the same work for more than one project or in more than one class, unless express and prior permission has been received from the course instructor(s) or research adviser involved.
    6. Forgery: altering any academic document, including, but not limited to, academic records, admissions materials, or medical excuses.