McCormick
Northwestern Engineering
Department of Electrical Engineering and Computer Science

EECS 230 – Programming for Engineers

Winter 2013 Quarter

Instructor: Goce Trajcevski

Contact: g-trajcevski@northwestern.edu

Class meets: Mon. (Tue.) Wed., Fri. 1:00-1:50PM – at L361 Tech.

Office hours: Mon./Wed. 2:00-3:00 (or, by appointment)

TAs: Leonidas Spinoulas and Bing Zhang

TAs Contact: LeonidasSpinoulas2015@u.northwestern.edu; BingZhang2017@u.northwestern.edu

TAs Office Hours: TBA

I. Course description: The main goal of this course is to expose the students to the fundamentals of programming in general, and focus on developing their skills for using the C++ language in particular. In addition to its "popularity", given its C-foundation, the C++ language provides an opportunity to introduce both the *procedural* (aka imperative) as well as the *object-oriented* programming paradigms. Time-wise, the course will have two major topics:

1. **C+** (or, C++ "as a better C") – This part of the course will focus on introducing the basic syntactic constructs (e.g., types and variables, branching, iterations, arrays, dynamic memory allocation, I/O, etc.) from the *imperative/procedural* standpoint. The pedagogical advantage of this part of the course is that the transition to another language from the same paradigm (e.g., Fortran, Basic) in the future, should be easier.
2. **C++ as an Object-Oriented language** – This part of the course will stress the key aspects of the O-O paradigm (classes and encapsulation, inheritance, polymorphism) as well as its impact on the overall software design stage, using C++ as a model-language. One benefit of this part of the course is that the students will gain understanding of some software engineering aspects that play an important role in the overall development process. This part should enable the students to have an easier adjustment to programming in other "purely-OO" languages, e.g., Java.

The course will attempt to strike a balance between the two fundamental aspects of studying any field: **– *depth,*** focusing on detailed examples when studying both the syntax and semantics of the important abstractions; and **– *breadth***, illustrating as many concepts as possible, for the purpose of generating a more complete picture of the different "players" (e.g., from algorithmic solutions' development, through translating them into a robust C++ code; to the global application-view).

II. Required text: "*C++ for Everyone*", by Cay Horstmann, (publisher: Wiley)

III. Recommended text and/or other materials:

a. "*C++: How to Program",* by Deitel&Deitel (publisher: Pearson (Prentice Hall), 6th or 7th edition);
b. Handouts that will be provided as part of the course.
c. NOTE: There required textbook is as close of a fit as possible for this course. There is a variety of books related to (introduction to) C++ programming. The reason for selecting the textbooks above are:
    i. The required text is pedagogically easier to follow for a first-time programmer;

ii. The recommended text, while possibly not being a subject to "immediate-benefits", has an extensive collection of example-codes. Hence, after some initial familiarization, the students should not have any problems using the recommended text as a "quick-lookup" source.

IV. Course Outcomes: After finishing the course, the students should be comfortable with:

a. Devising C++ code for solving problems of interest in various engineering disciplines;
b. Devising an overall code-organization plan, for the purpose of integrating their code in larger applications;
c. Getting involved in existing applications and incorporating their solutions;
d. Reading programming texts/code and adjusting to new development environments.

V. Tentative Course Outline

Week1	- Introduction/Motivation/History of programming; - The basics of C++ program "lifetime"; • Visual C++ IDE (Integrated Development Environment); - Basic Data Types and Variables;
Week2	- Program Flow • Comparisons, Relational Operators, `if-then-else` branching; - Nested Branching; - Boolean type and Operators; *Homework #1 Assigned;*
Week 3	- Looping constructs • `while`; `for` and `do` loops - Introduction to Functions and parameters/arguments; *Homework #1 due; Project #1 assigned*
Week4	- Functions (cont.) • Scopes of Variables and References; • Recursion; **Quiz#1** *Project #1 due; Project #2 assigned;*
Week5	- Arrays and Vectors; • Strings; - Pointers and Dynamic Memory Allocation; *Project #2 due; Project #3 assigned;*
Week 6	- Arrays and Pointers; - Streams (Files I/O vs. command line arguments) **Midterm** - Introduction to Classes
Week 7	- Classes and encapsulation; • Object-Oriented paradigm and C++ - Data members and Member functions; - Objects and their lifetime • constructors and destructors; - Access privileges and `friend`-ship - Compilation with classes *Homework #2 posted; Project #3 due;*
Week 8	- Nested classes; - Inheritance and Derived classes; • Inheriting data members (w. access) and overriding member

	functions; **Quiz#2** *Homework #2 due; Project #4 posted;*
Week 9	- Virtual functions and polymorphism; - Abstract classes and pure virtual function; - Multiple inheritance in C++; *Project #4 due; (possibly, Project #4.5 assigned);*
Week 10	- Possible-Porpourri: • Templates (functions and classes); • Namespaces/STL; *Project #4.5 due;*
Week 11	- **Final exam (material after the midterm only)**

VI. Grading

Your grades will be based on:

- Two homeworks (6%)
- Project #1 (6%)
- Project #2 (7%)
- Project #3 (8%)
- Project #4 (8%)
  - Project #4.5 (4% – team)
- Quiz #1 (8%)
- Quiz #2 (8%)
- Midterm (22%)
- Final (23%)

Please note: the distribution given above is approximate and may be subject to some very minor changes. However, the firm-policy will be announced during the last week of classes (Week #10 of the Winter quarter).

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*. As indicated, the last "0.5" programming assignment will most likely be done in teams, however, if a particular assignment is indicated as *individual* – although you are encouraged to always discuss class-related issued with your classmates – it is your responsibility to ensure that the work is done individually. 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, 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 (both in terms of 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 assignment. Hence, you really need to keep yourself up-to-date with the material lectured and 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.