ECE2036 Course Syllabus

ECE2036

Engineering Software Design (3-0-3-4)

CMPE Degree
This course is Required for the CMPE degree.

EE Degree
This course is Elective for the EE degree.

Lab Hours
0 supervised lab hours and 3 unsupervised lab hours

Course Coordinator
Davis,Jeffrey A

Prerequisites
ECE 2020/2030 [min C] and ECE 2025/2026* [min C] * Prerequisites indicated with an asterisk may be taken concurrently with ECE2036

Corequisites
None

Catalog Description
Object-oriented software methods for engineering applications. Numerical analysis methods; simulations and graphical presentation of simulation results; analysis of numerical precision. Programming projects.

Textbook(s)
Deitel and Deitel, *C++, How to Program* (10th edition), Prentice Hall. ISBN 9780134448848 (required)

Eckel, *Thinking in C++: Introduction to Standard C++, Volume One* (2nd edition), Prentice Hall, 2000. ISBN 0139798099, ISBN 9780139798092 (required) (comment: text is available free on line at http://original.jamesthornton.com/eckel/)

Course Outcomes
Upon successful completion of this course, students should be able to:

1. Write a syntactically and semantically correct program using the object-oriented programming language chosen, such as C++.
2. Define and develop a computer program to implement specific tasks associated with a well-defined engineering application.
3. Choose the appropriate data structures to maintain the system state, and explain why the selected one is appropriate.
4. Choose appropriate algorithms to maintain the progression of the system state, and explain why the selected algorithm is appropriate.
5. Implement those data structures and algorithms using an object-oriented programming language such as C++.

6. Use the developed program to analyze performance and behavior of the engineering application being studied.
7. Create a class hierarchy of software objects with public inheritance and virtual functions to enable the use of polymorphism.
8. Create class templates and function templates to be used in a generic programming style.
9. Analyze the numerical error in floating-point calculations of linear equations of at least one multiplication and one addition.
10. Use commonly available tools for software development, source code maintenance, and debugging.
11. Implement their program using multiple threads in a shared memory, multiprocessor environment when appropriate.
12. Recognize when and how an algorithm will benefit from parallelization.

Student Outcomes

In the parentheses for each Student Outcome:
"P" for primary indicates the outcome is a major focus of the entire course.
“M” for moderate indicates the outcome is the focus of at least one component of the course, but not majority of course material.
“LN” for “little to none” indicates that the course does not contribute significantly to this outcome.

1. (P) An ability to identify, formulate, and solve complex engineering problems by applying principles of engineering, science, and mathematics
2. (M) An ability to apply engineering design to produce solutions that meet specified needs with consideration of public health, safety, and welfare, as well as global, cultural, social, environmental, and economic factors
3. (LN) An ability to communicate effectively with a range of audiences
4. (LN) An ability to recognize ethical and professional responsibilities in engineering situations and make informed judgments, which must consider the impact of engineering solutions in global, economic, environmental, and societal contexts
5. (LN) An ability to function effectively on a team whose members together provide leadership, create a collaborative and inclusive environment, establish goals, plan tasks, and meet objectives
6. (LN) An ability to develop and conduct appropriate experimentation, analyze and interpret data, and use engineering judgment to draw conclusions
7. (M) An ability to acquire and apply new knowledge as needed, using appropriate learning strategies.

Topical Outline

Required Topics:
1. Review of C basic syntax, compilation, linking, libraries, etc
2. Defining and implementing classes, constructors, destructors e
3. Member functions, virtual functions, pure virtual functions
4. Argument passing variations (by value, by pointer, by referenc
5. Managing dynamic memory (new, delete)
6. Inheritance and subclassing
7. Using common tools, gdb, make, gprof, valgrind, emacs etc.
8. Floating Point precision and numerical analysis
9. Introduction to Templates, including data structures and algor
10. Parallel processing and concurrency

Optional Topics:
11. Exceptions
12. Smart Pointers

Typical Programming Projects:

1. One dimensional and two dimensional Fast Fourier Transforms, u
2. Matrix multiplication using dynamic memory allocation for arbi
3. Single variable numerical methods to solve first-order differe
4. Embedded programming projects using sensor feedback, interrupt
5. Optimal (and non-optimal) search for solving problems with