

ENG EC327 INTRO SOFTWARE DESIGN

2008-2009 Catalog Data:

Prereq: ENG EK 127. The goal of this course is to introduce engineering students to advanced programming techniques and basic data structures concepts. The course will start with a fast-paced introduction to the fundamentals of object-oriented programming, dynamic memory allocation, and file input/output operations. The stress in this introduction will be on practical programming issues, such as proper programming style and optimization, debugging techniques and compilation, and graphical user interfaces. Students will also be introduced to fundamental data structures, such as linked lists, queues, trees, hash tables, and graphs, and algorithmic analysis techniques in the context of searching and sorting methods. Throughout the course, students will utilize industry-standard programming tools, and examples for theoretical concepts will be provided from contemporary applications. Duplicate credit with CAS CS 111; cannot take both for credit. 4cr.

Class/Lab Schedule:

LEC: 4 hrs/wk (TR12-2pm)

Lab monitoring 8 hrs/wk (M2-4pm, T2-4 , R2-4pm, R4-6pm)

Status in the Curriculum: Required

Textbooks and other required materials:

Y. Daniel Liang, Introduction to Programming with C++ (comprehensive version), Prentice Hall, 2007.

Reference:

Mark Allen Weiss, Data Structures & Algorithm Analysis in C++ (2nd edition), Addison-Wesley, 1999; The C++ resources network <http://www.cplusplus.com/>.

Central Consulting, <http://www.bu.edu/cc>: free tutorials on UNIX and other fundamental systems.

Coordinator:

Ari Trachtenberg, Associate Professor, ECE Department

Prerequisites by topic:

Elementary mathematics and programming

Goals:

To provide students with:

- basic knowledge of C and C++ program and software design.
- introductory knowledge of simple data structures and algorithms.
- skills in solving simple engineering math: e.g. linear algebra and root finding
- fundamentals of Unix and Microsoft programming environment.

Course Outcomes:

As an outcome of completing this course, students should be able to:

1. Understand program development cycle
2. Understand data primitives and declarations
3. Understand control flow: selection and loops
4. Understand function, call stack, scope, parameters
5. Understand arrays, strings and pointers
6. Utilize program development: Makefiles
7. Develop good programming style: variable names , comments
8. Utilize IDE, version control, libraries, GUI
9. Ability to debug using gdd, ddd, Visual Studio
10. OOP : classes, inheritance, polymorphism
11. Data structures and algorithms: sorting, linked lists, stacks, queues
12. Understand basic for trees and recursion
13. Complexity bounds and the example of the FFT.
14. Work in a team on programming development
15. Understand the many-core trend and the GPU technology

Course Outcomes mapped to Program Outcomes:

Program Outcomes:	a	b	c	d	e	f	g	h	i	j	k
Course Outcomes:	1-13	11-13		14	6-13		14	12	1-14	15	8-13
Emphasis:	5	3	1	2	5	1	2	2	5	3	5

1=not at all; 5=a great deal;

Topics in Project Assignments

Draw simple graphics on the screen using a Windows graphics application in Visual Studio; Programming in the CUDA the Nvidia GPU chip for simple image manipulation.

Contribution of Course to Meeting the Professional Component:

Engineering topics: 70%

Math & Basic Science: 25%

General Education: 5%

Prepared by: Prof. Richard C. Brower **Date:** June 10, 2009