

CS 361: Advanced Data Structures and Algorithms
Summer 2005
Course Outline

Instructor: Ravi Mukkamala
Office: E&CS 3317
Phone: 683-3901 (Voice) 683-4900 (Fax)
e-mail: mukka@cs.odu.edu
Lecture: Tuesday, Thursday 1:00-2:45 PM (Hughes 1110)
Office Hours: Tuesday, Thursday 3-4 PM (E&CS 3317)
Prerequisites: CS 250 & Math 163
Text: *Data Structures in C++ (REQ)* by Timothy Budd
Programming in C++ (REC) Schaum's Outlines 2nd Ed

University Catalog Course Description

Lecture 3 hours; 3 credits. Laboratory work required. Prerequisites: MATH 163, CS 252 and either CS 250 or 333. Common abstract data types, including vectors, lists, stacks, queues, sets, maps, heaps, and graphs. Standard C++ interfaces for these ADTs. Generic programming via iterators and templates. Choosing data structures and algorithms to implement ADTs, via analysis of their time and space requirements.

Course Objectives

This course explores fundamental data structures, algorithms for manipulating them, and the practical problems of implementing those structures in real programming languages and environments. Heavy emphasis is placed upon the analysis of algorithms to characterize the expected worst and average case requirements for running time and the anticipated size of memory requirements. Perhaps more than any other course, CS361 should expand the students ``toolbox" of basic techniques for manipulating data at both the conceptual and the concrete level. At the conceptual level, the student will see a broad selection of standard practices and approaches used in program design. At the concrete level, the student will begin what should be a career-long practice of accumulating useful, reusable code units. Upon successful completion of this course, the student will have gained an appreciation of some general *programming costs* associated with the use of certain data structures and the common algorithms used to manipulate those structures.

Grading Criteria

Number	Type	Points
10	Weekly Programming/Written Assignments	400
	Examination I	125
	Examination II	175
	Examination III	200
	Total	900

Grading Scale

Percentage	Letter Grade
95-100	A
90-94	A-
87-89	B+
84-86	B
80-83	B-
76-79	C+
72-75	C
69-71	C-
0-68	F

Policies

Electronic mail on the CS Dept. network will be used for timely communication, especially for clarification/corrections/changes to homework or projects. Students should check their e-mail on a regular basis.

The directory `~mukka/361sum05` will serve as a repository of information related to this course. This will include a weekly set of agenda files, lab assignments and programs related to course projects, and examples and handouts.

Students should expect and plan for a short assignment almost every week. Most of these will be programming assignments; the rest will be written problem sets. Students are presumed to be familiar with basic programming techniques, including the use of pointers, functions and procedures, selection, loops and recursion. Also assumed is facility with basic algebra.

Do I Need to feel comfortable in writing programs in C++? Absolutely! For all programming assignments students will use C++; it will be used in the lectures and examples. All students will eventually need to be proficient in writing C++ source code. However, it is possible that some students may enter this class with a background in other programming languages like Pascal.

Do I Need to feel comfortable with the UNIX operating system environment? Not necessarily. You have the choice of working with Windows based C++ compilers such as DEV C++ or Visual Studio. You may also use GNU C++ on Unix. Students who are unfamiliar with UNIX must make it their own responsibility to learn how to use it.

CS 361 is not, however, a programming language class. Students are responsible for teaching themselves (if necessary) the more basic C++ constructs (i.e., those parts of the language that have direct analogies in Pascal, including expressions, assignment and control-flow, and functions). More advanced C++ constructs will be covered in class as needed. The topics in this course are scheduled in such a way that no more than the

ability to read basic algorithms in C++ is required in the first two to three weeks.

For all examinations, you will be allowed to use as resource materials any computer science text, your class notes, computer documentation and your projects.

Honor Code

The Honor System at Old Dominion University is based upon the integrity of each student. Any form of dishonesty or deception such as lying, cheating, and plagiarism constitutes a violation of the Honor System. Since each student has signed an honor pledge on the course registration form (BAR Form), all material submitted for grading by a student during the course is to be the student's own work.

The instructor reserves the right to question a student orally or in writing and to use his evaluation of the student's understanding of the assignment and of the submitted solution as evidence of cheating. Violations will be reported to the Honor Council for consideration for punitive action.

By the Computer Science Policy, students found to be in violation of this rule will, at the very least, receive a failing grade in the course and may be subject to additional penalties. Students who contribute to violations by sharing their code/designs with others are subject to the same penalties. Students are expected to use standard UNIX protection mechanisms (chmod) to keep their assignments from being read by their classmates. Failure to do so may result in grade penalties.

This policy is not intended to prevent students from providing legitimate assistance to one another. Students are encouraged to seek/provide one another aid in learning to use the operating system, in issues pertaining to the programming language, or to general issues relating to the course subject matter. Student discussions should avoid, however, explicit discussion of approaches to solving a particular programming assignment, and under no circumstances should students show one another their code for an ongoing assignment, nor discuss such code in detail.

If examinations are missed, they will receive a grade of **ZERO** unless prior permission has been given by the instructor.

Late Work is Not Acceptable

Late papers and projects, and make-up exams will not normally be permitted. I will give appropriate consideration to documented emergencies, but such arrangements must be made prior to the due date for all situations where the conflict is foreseeable.

Extensions to due dates will not be granted simply to allow "porting" from one system to another. "But I had it working on my office PC!" is not an acceptable excuse. In a similar vein, anyone who has achieved junior-level or higher status in a CS program should by now be aware that the network and workstations generally get overloaded just before assignments are due, and that machine crashes and down-time are a fact of life in this field. Students are expected to plan for these problems. Except in extreme cases, these are not grounds for extension of a due date.

C++ is a language that is still undergoing rapid change. Consequently, code accepted by one C++ compiler may fail to compile in another. You may use either the Dev-C++ or the MS Visual .Net C++. Both are available on the lab machines. You may also download a free version of the dev-C++ on to your laptop or personal PC by visiting www.bloodshed.net. It is your responsibility to make sure that your programs run on either one of these compilers. Please don't underestimate the amount of time that may be involved in coping with subtle differences among compilers.

Class Attendance

Because the class period is important and discussions cannot be reproduced, absences cannot be made up. Excessive absences may have a negative effect on a student's learning and performance. Any student who must miss a class is expected to have the initiative necessary to properly cover the material missed (i.e. assignments given or modified, due dates established or modified and any handouts, etc.). Students are required to meet all course deadlines and be present for all examinations.

Tentative Lecture Schedule

Date	Topic	Chapters	Assignments
May 17	Fundamentals	Chapter 1	
May 19	Classes and Object-oriented Programming	Chapter 2	HW 1 assigned
May 24	Algorithms	Chapter 3	
May 26	Analyzing Execution Time	Chapter 4	HW 1 due
June 2	Increasing Confidence in Correctness	Chapter 5	HW 2 assigned
June 7	STL: General Introduction	Chapter 6	Hw 2 due; HW3 assigned
June 9	Vectors	Chapter 8	
June 14	Vectors (Cont.)	Chapter 8	HW 3 due; HW4 assigned
June 16	Exam I		Chapters 1-8
June 21	Lists	Chapter 9	HW 4 due
June 23	Lists (Cont.)	Chapter 9	HW 5 assigned
June 28	Stacks and queues	Chapter 10	
June 30	Deque	Chapter 11	HW 5 due; HW 6 assigned
July 2	Sets and multisets	Chapter 12	
July 5	Sets and multisets (cont.)	Chapter 12	HW 6 due; HW 7 assigned
July 7	Exam II		Chapters 9-12
July 12	Trees	Chapter 13	HW 7 due; HW 8 assigned
July 14	Searching	Chapter 14	
July 19	Priority Queues	Chapter 15	HW 8 due; HW 9 assigned
July 21	Maps and Multimaps	Chapter 16	
July 26	Hash tables	Chapter 17	HW 9 due; HW 10 assigned
July 28	Matrices	Chapter 18	
August 2	Graphs	Chapter 19	HW 10 due
August 4	Exam III		Chapters 13-19