Course Information

Lectures

Lectures are **MWF 10:00–10:50** in 217 Art.

Discussion Sections

Discussion sections are **M 5:10–6:00** in 204 Art, and and **W 4:10–5:00** in 7 Wellman Hall.

Discussion sections begin next week. Go to one discussion section each week. The same material will be covered Monday and Wednesday. Early discussion sections will give you help in C++, while later discussion sections will deal more with lecture material.

Instructor

Prof. Phillip Rogaway. Eng. II, #3063. Phone: 752–7583. `rogaway@cs.ucdavis.edu` Office hours: **R 9:10–11:30** in my office.

Teaching Assistants

Ted Krovetz. `tdkrovetz@ucdavis.edu`. Office Hours: **F 1:00–3:00** in 55 Eng. Unit II (the `TA room`).

Janine Taylor. `taylorj@cs.ucdavis.edu`. Office Hours **M 11:15–1:15** in 55 Engineering Unit II (the `TA room`).

Midterms

*Important: There will be two **evening** midterms, to be held Thursday 10/26 and Thrusday 11/16.* Midterms are tentatively schedule for 7:00–9:00 pm. To compensate for the evening exams there will be no class meetings on the following Fridays: 10/27 and 11/17.

I am trying the evening exams because I have found that students can feel overly time-preassured during what I intend to be a 50 minute exam. The evening exams are meant to relieve time preassure. I hope most students will prefer this format. I will attempt to make the exams no longer than an in-class exam (I'll pretend I'm writing a 50 minute exam). Exams will be closed book, closed notes.

Important: If you have an irreconcilable conflict with either of the Thursday exams then you must see me during the first week (10/2–10/6) of the term. I won't be too accommodating after that!

Prerequisites

ECS 40 or equivalent.

Textbooks

The following is the *required* text:

E. Horowitz, S. Sahni and D. Mehta, *Fundamentals of Data Structures in C++.* W.H. Freeman and Company, 1995.

Your programs will be written in C++. I won't teach you much C++. To learn it you'll need to buy some C++ text and study it on your own.

There are now an incredible number of books on C++. Which one you choose from this multitude is entirely up to you. I have ordered two good ones, to suit different tastes and learning styles. These two books are in many ways opposites.

- **B. Stroustrup**, *The C++ Programming Language*, second edition, Addison-Wesley, 1994.

  This book is deep, comprehensive, terse, intelligent. The definitive C++ text, by the inventor of the language, it belongs in the library of every professional C++ programmer. But this text will be regarded as difficult for the average student reading his first description of the C++ language.

- **G. Perry**, *Moving from C to C++.* Sams Publishing, 1992.

  This is an easy, enjoyable read. It has a sort of you-can-do-this, PC-mass-market attitude. You should be able to absorb the material of parts I–III (which is roughly the part of C++ I expect you to use) in about a week. Learning the same "core" of ideas and syntax out of Stroustrup will take you significantly longer.

Other well-regarded texts include: **S. Lippman**, *C++ Primer*, Addison-Wesly, 1991. *A well-liked book on the language, it has often been used for this class.* And **S. Meyer,** *Effective C++: 50 Specific Ways to Improve Your Programs and Designs. A great book on how to improve your C++ programming once you know the basics of the language.* Finally, the first chapters of your course textbook include a substantial amount of C++ material. (Beware that some of this is styalisitically marginal.)

Material

This is a course in *data structures.* This isn't a course in C++ or Object Oriented Design. We'll cover "standard" material on data structures, like stacks, binary trees, graphs, heaps, etc.. At the end of this course you should be able to choose the "right" data structures for a programming task; analyze the behavior of basic algorithms which use data structures; and implement in a clean manner algorithms which make use of nontrivial data structures.

Programming Assignments

You will be doing four programming assignments, all in C++. These are designed to help you bridge the "theory" and the "practice." They also aim to give you experience with increasingly large or interesting programming problems. I intend that at least two of the programs be quite challenging. Start early.

Computing Resources

All students registered in this class should have accounts already setup. Use the instructional workstations in the basement of Engineering II. Most of these are DEC workstations, but there's also some Silicon Graphics machines. These use a different flavor of UNIX, but you should have little problem getting your work done in either environment. If you didn't have an account on these machines already setup then you should find your name on the list in the basement, together with the new account name. Change your password with `yppasswd` right away; it is initially the last 8 digits of your student id number. Accounts with passwords not changed within a week will be disabled.

If there's a problem, see the student programmers first (they're in the basement of Engineering II). They keep long hours and are very helpful. The student programmers also respond to mail: the address is `support@cs`.

You are welcome to do the assignments on your own machine. For example, some of you might own a PC with a C++ compiler such as Borland C++. There are some differences across C++ implementations, but this is unlikely to get in your way if you stick to "standard" C++.

Newsgroups

Read `ucd.class.ecs110` regularly; you are responsible for everything posted there. There is also a `ucd.class.ecs100.d` discussion newsgroup; read this if you want to. This newsgroup is for you to talk among yourselves and answer one another's questions.

Learning C++

Many students are shocked by the "you're-on-your-own attitude" we take towards your learning a new programming language. Be assured that its not *that* hard to pick up a new programming language (even one as complicated as C++) just by reading a book. Mostly, you should learn C++ from your reading. But I suggest that you experiment a bit on the computer as you do the readings, making up some "toy" programming tasks before you tackle the first assignment. You will get some help learning C++ from the early discussion sections, but don't count on the discussion sections to teach you all you need to know. I strongly advise you to get started learning C++ *immediately.*

I plan to give a C++ quiz on Friday, 10/13. That's just two weeks from today ...

Grading

There will be *four assignments*, *two midterms*, a *final*, and a *C++ quiz*. The homeworks will count for about **32%** of your grade (8%, 8%, 8%, 8%); the midterms about **30%** of your grade (15%, 15%); the final about **35%** of your grade; and the C++ quiz for about **3%** of your grad. The midterms and final are closed book, closed notes. As stated earlier, the midterms will be in the evening. See the section "Midterms" for the dates.

Turning in the Assignments

We're doing this the low-tech way: turn in hardcopy in Eng. II, room 2131. Homeworks are due at 4:00 p.m. on the date specified as their due date (usually a Monday).

Securely staple your assignment together. Clearly indicate your name and e-mail address on the first page. I encourage you to highlight important function names and class names to help make your programs easier to read. Each assignment will have questions to turn in. These are an integral part of the assignment. You will also need to turn in an execution script which shows that your program correctly runs. Keep this short but convincing. Clearly identify any known bugs.

Read the section on Academic Misconduct concerning the implicit statement which accompanies each assignment you turn in.

You are required to retain a copy of each turned-in program until the end of the academic term.

Late Assignments

You may turn in assignments 1, 2 or 3 up to 3 calendar days late; doing this will cost you 15% of the value of that assignment. You may turn in assignments 1, 2, or 3 up to 7 calendar days late; it will cost you 30% of the value of that assignment. Assignment 4 may not be turned in late.

Write your own code

You are encouraged to discuss program design with fellows students, but all the code should be written *on your own.* Don't use a friend's code. Don't modify someone else's code. Don't use class libraries which you might dig up (except, of course, for standard libraries like `iostream`). This might not be the way that the programming works in the real world, but it's the way things work in this class. Build your programs from scratch (it's more fun that way, anyway).

When you discuss the ideas behind a program's solution with a friend, that is fine. But please indicate on your writeup with whom you have exchanged ideas. This has no effect on the grading of assignments.

Academic Misconduct

We have experienced problems with people turning in substantially identical programs. Contrary to students' expectations, it is rather easy to spot such things. We have also had incidents of people fabricating program output. If there's a bug in your program and you kludge program output, you will probably get caught. Both behaviors constitute academic misconduct.

Incidents of academic misconduct will be reported to the UC Davis Office of Student Judicial Affairs. The penalties are severe, including dismissal from the university.

When you turn in a program with its output you are certifying to me that (1) the turned-in program produced the turned-in output; (2) you wrote the code yourself; (3) you did nothing to intentionally facilitate someone else copying your program; (4) you have indicated all the people with whom you had substanative exchanges of ideas.

If you are having personal or academic difficulties which are tempting you to engage in such misconduct, please come speak to me instead.

Parting Thoughts

Programming non-trivial tasks is a process of planning, abstraction, and design. If you find youself spending less time engaged in such activities than you you spend in front of a machine typing in and debugging code it, then you're doing something wrong and you should try to adjust they way in which you work.

Data structures was one of my favorite undergraduate CS classes— it was here that I first got the computer to tell me things which I could not have figured out any other way. I hope you get the same sense of discovery.