CS 333: Syllabus (Spring 2016)

Chuck Cartledge

Last modified: Jan 07, 2016

Contents:

1. Basic Course Information

1.1 When and Where

This course is online and has no regularly scheduled lectures. There will be a limited number of attendance-optional network conferences (announced on the website).

1.2 Objectives

This course covers basic C++ programming and the software development issues that arise in practical programming projects. Topics include C++ syntax and semantics, principles of design and basic software engineering skills.

This course satisfies the requirements of both CS 150 and 250. It is intended for the student who has already been introduced to programming, possibly in another programming language. Students who lack prior programming experience may attempt the course, but they should be aware that, because *this is* an *accelerated course* that covers the *equivalent of two other courses*, students may find that this course requires far more work than a normal single course.

This is a web-delivered course with no scheduled class meetings, except for a few meetings conducted via network conferencing. These will be recorded for the convenience of students unable to attend live.

This course offers students considerable flexibility in scheduling their work, but students without the maturity to take advantage of the available learning resources or the self-discipline to maintain a regular work schedule run the risk of falling too far behind to succeed.

1.3 Required Text

Malik, *C++ Programming: from Problem Analysis to Program Design*, 7th Edition, Thomson Course Technology, ISBN 1285852745 8–1285852744

This book is frequently sold with accompanying CDs. We won't use them, so don't worry about it. If you can get a used copy that is missing the CDs, or if you can get a better price on a new one without the CDs, go for it!

2. Communications

2.1 Instructor

Chuck Cartledge	Virtual office
(757) 633–2581	ccartled@cs.odu.edu

Make sure to include the course name **CS333** in the subject line of any email related to this course.

Office Hours

My general office hours are available at http://www.cs.odu.edu/~ccartled

/Teaching. Note that my office hours include both live and web-conference based appointments.

General questions about course content and reports of website problems should normally be asked in the Forums on Blackboard.

Questions about grades, how to solve assignments and other graded activities should be send to ccartled@cs.odu.edu.

For more discussion on course communications, please refer to the Communications policy.

3. General Organization of the Course

The course is divided into three major parts. Part I corresponds roughly with the portion of CS150 that does not overlap with CS250. The emphasis here is on the basic C++ language. For this mostly technical material, your major source of information will be the textbook.

Parts II and III correspond roughly to CS250. Although new C++ language features continue to be introduced, the emphasis in these sections is to place the process of writing code within the larger context of programming, where coding is preceded by design and followed by testing and debugging. In these parts of the course, you will find that a much higher portion of your information comes from the instructor's lecture notes, which attempt to give a sense of perspective that the text, with its narrow emphasis on the C++ language itself, lacks.[1]

Within each part of the course are several topics. Each topic is addressed by readings (from the text or the online lecture notes) and accompanied by a variety of activities including:

- Self-tests - ungraded online quizzes
- Labs - ungraded activities that introduce or practice techniques that you will use in the assignments
- Assignments - graded activities, most of which involve programming
- Unix Assignments - references back to the co-requisite course CS252. These indicate that you should have completed the CS252 curriculum up to and including the listed assignment, because you will need or, at least, might benefit from, the skills taught in that course.

  If you have already taken and passed CS252 in a prior semester, you will not need to (and not be able to) actually repeat these assignments. But it

may be a good idea to return to the CS252 website and review the lecture notes related to the listed assignment.

In addition, there will be an exam at the end of each of the three parts, with the final exam after part III being cumulative.

There will also be a semester *term* project in which you will apply the techniques of design, coding, testing, and debugging to a larger problem than is tacked in the assignments.

This being a web-course, you have considerable freedom in scheduling your time. You need to take care, though, to keep up the pace (and, because this *is* an accelerated course covering two semesters' worth of material, that pace will be challenging).

4. Course Pre-requisites and Co-requisites

The prerequisites for this course are:

- CS 150, Problem Solving and Programming I, or an equivalent course in any high-level programming language
- Math 163, or an equivalent course

The co-requisite for this course is:

- CS 252, Introduction to Unix for Programmers

  A co-requisite is a course that must be taken before or during the same semester as this course.

- Although CS252 is, itself, a self-paced course, students in CS 333 will be expected to have completed specific portions of it in preparation for some labs and assignments, as indicated on the course Outline page.
- All students are enouraged to complete CS 252 during Part I of the course–i.e., the first 3 weeks of the semester.

5. Assignments

Assignments for this course will include numerous small assignments and a term project. Both may involve programming in C++.

5.1 C++ compiler

The "official" compiler for this course is the Free Software Foundation's g++

(also known as gcc or GNU CC), version 4.8.1 or higher. This is the compiler that the instructor and/or grader will use in evaluating and grading projects. If you have access to other compilers, you may use them, but *you* are responsible for making sure that their projects can be compiled by the instructor and/or the course's grader using the official compiler.

You may want to develop your programs on the most convenient compiler and then port it over to the official environment. Please do not underestimate the amount of time that may be involved in coping with subtle differences among compilers.

You can do all work in this course using g++ on the CS Dept Unix servers via ssh/X or via the CS Dept's Virtual PC Lab. If you like, however, you can obtain the g++ compiler for free from a variety sources. Links will be provided on the course Library page.

5.2 Computer Accounts

Students will need an account on the CS Dept. Unix network to participate in this class. This account is unrelated to any University-wide account you may have from the ODU's computing services (ITS).

If you have had a CS Unix account in the recent past, you should find it still active with your login name, password, and files unchanged. If you have had an account and it has not been restored, contact the CS Dept systems staff in the lab in Dragas Hall, Room 1111K or email root@cs.odu.edu requesting that it be restored.

If you do not yet have such an account, follow these directions to get set up. *New account creation for students enrolled in a future semester becomes available about one week before the start of that semester.*

6. Course Policies

6.1 Assignments and Grading

Assignments will be turned in through the CS submission system, rather than through Blackboard–more information is available here. Most of the assignments will be graded by an automatic grader. The results will be sent to your CS email account. Unless the assignment explicitly states otherwise, you may submit a total of three times per assignment; the instructor will take the last of the marks, although you may request that your score be "rolled back" to an earlier one. You may NOT submit after viewing the sample solution.

6.2 Due Dates and Late Submissions

There are a few hard due dates in the course.

- The dates for the three exams are fixed.
- The assignments in each of the 3 parts of the course are due before the exam at the end of that part.
- Because the semester project is submitted in phases, with the intent of providing you with feedback as you move into the next phase, there is a *fixed due date for each phase of the project*.

You can choose what days of the week to work on and what times of day to work, but you cannot choose to defer all the work to the last month of the semester.

All assignments (not including the term project phases) are due prior to the date of the exam at the end of the Part of the course in which they are given. Late submissions of assignments from part I or Part II will be accepted during the following part, subject to a 20% penalty.

Late submissions on the portions of the semester project and make-up exams will not normally be permitted.

> Exceptions will be made only in situations of unusual and unforeseeable circumstances beyond the student's control, and such arrangements must be made prior to the due date in any situations where the conflict is foreseeable.

"I have fallen behind and can't catch up", "I am having a busier semester than I expected", or "I registered for too many classes this semester" are *not* grounds for an extension. Extensions to due dates will not be granted simply to allow *porting* from one system to another. *But I had it working on my home PC!* is not an acceptable excuse.

6.3 Academic Honesty

Everything turned in for grading in this course must be your own work.

The instructor reserves the right to question a student orally or in writing and to use this evaluation of the student's understanding of the assignment and of the submitted solution as evidence of cheating. Violations will be reported to the Office of Student Conduct & Academic Integrity for consideration for possible punitive action.

Students who contribute to violations by sharing their code/designs with others may be subject to the same penalties. This includes showing material to other students in person and *posting code and designs in any public area, whether physical or on the internet. Students are expected to use standard Unix protection mechanisms[2] to keep their assignments from being read by their classmates.*

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. The same guideline applies to discussions, whether face-to-face or on-line, with anyone other than the course instructor and TAs – general aid on the subject matter of the course is OK. Specific discussions of solutions to any graded activity are forbidden.

> Students must avoid explicit discussion of approaches to solving a particular programming assignment. Under no circumstances should students show one another their code for an ongoing assignment, nor discuss such code in detail.

6.4 Grading

Assignments:	45%
Semester Project:	20%
Exam 1	10%
Exam 2	10%
Final Exam:	15%

Read these further notes on grading for a detailed explanation of how grades are computed.

6.5 Auto-Grader & Testing

Test driven development is a topic of particular import–not only in academia, but in industry.

You will be expected to make use of Blackbox Testing after it is covered in Part II of this course. This is a topic of particular import. You will also need to make use of white-box testing and unit testing–covered during Part III of this course.

Difficulties with the tests performed by the Auto-Grader should be addressed

after the first assignment submission. If a program fails a test, there is usually an edge--or corner--case for which you–the student– did not account.

All tests are designed by me–the instructor. The Auto-Grader runs tests that I use to evaluate my solution. These tests evaluate mechanics of import–e.g., dynamic binding and function overloading.

Be systematic in all changes to your assignment solution and modifications to your tests. Do not haphazardly make changes to an assignment and resubmit hoping for a better grade. Treat each submission attempt as your final submission. Ask for guidance before each subsequent submission.

6.6 Status Reports

You may be instructed to submit status reports. These reports may include: general progress reports or project status reports. These reports will be used to assess your progress in the course, address difficulties with course content, and provide general guidance on assignments.

Specific requirements for status reports will be discussed by the instructor.

7. Review Sessions

I conduct Review Sessions at selected times during the semester–approximately once every two weeks. Instructions regarding how to attend live sessions and view recorded reviews are available here. Attendance for live sessions is not mandatory. **You are expected to view the recordings of these Review Sessions.**

8. Educational Accessibility

Old Dominion University is committed to ensuring equal access to all qualified students with disabilities in accordance with the Americans with Disabilities Act. The Office of Educational Accessibility (OEA) is the campus office that works with students who have disabilities to provide and/or arrange reasonable accommodations.

- If you experience a disability which will impact your ability to access any aspect of my class, present me with an accommodation letter from OEA so that we can work together to ensure that appropriate accommodations are available to you.
- If you feel that you will experience barriers to your ability to learn and/or complete examinations in my class but do not have an accommodation letter, consider scheduling an appointment with OEA to determine if

academic accommodations are necessary.

The Office of Educational Accessibility is located at 1021 Student Success Center and their phone number is (757) 683–4655. Additional information is available at the OEA website http://www.odu.edu/educationalaccessibility/

---

1. Not that this textbook is unusual in that respect. Pick up almost any programming textbook, turn to the index, and look up the terms *design*, *testing*, and *debugging* and note what a small percentage of the total pages in the book are devoted to these activities. Then reflect on the fact that these activities typically account for the lion's share of most programmers' time, with actual code writing coming in a distant fourth. ↩

2. You will learn about these in the co-requisite course, CS252. ↩