Computing & Software Systems 342:
Data Structures and Object-Oriented Programming I
Winter 2018
Syllabus

Description

This fast-paced course is intended to enable students to design solutions to programming problems using object-oriented techniques. The course integrates the fundamental discrete mathematics aspects of computing with detailed instruction in end-to-end software design. By the end of this quarter, you will be familiar with much of the C++ language and the basics of object-oriented programming. You will understand how to analyze a problem and design a solution. You will understand the basic data structures and algorithms which are the basis for much of computer science. You will also be able to analyze the trade-offs among memory, execution time, and implementation complexity associated with them. Topics include: recursion, computational complexity and algorithm analysis, logic, induction, lists, stacks, queues, sorting and searching. Also covered are object oriented fundamentals such as abstraction, encapsulation, hierarchy, and polymorphism.

Details

Instructor: Bob Dimpsey, dimpsey@uw.edu
Office Hours: MW 4:30 – 5:30 PM, UW1-271G
Course Website: http://courses.washington.edu/css342/dimpsey
Lectures: MW 5:45 – 7:45 pm, UW2 340

Grader: Fahad Alsheri, fahaduwb@uw.edu

Textbooks

Books Utilized in Class


Available at the campus library here.

An Active Introduction to Discrete Mathematics and Algorithms (version 2.6), Charles Cusack, David Santos, GNU Free Software, 2016:
It is also recommended that you obtain a good C++ book.

**Recommended C++ Books**

*C++: The Core Language*, Gregory Satire and Doug Brown, O'Reilly. (Good if you are coming from a C background)


*Effective C++: 50 Specific Ways to Improve Your Programs and Designs (2nd Edition)*, Scott Meyers, Addison-Wesley.

*More Effective C++: 35 New Ways to Improve Your Programs and Designs (2nd Edition)*, Scott Meyers, Addison-Wesley.

C++ Notes from Professor Zander: [http://courses.washington.edu/css342/zander/css332/](http://courses.washington.edu/css342/zander/css332/)

**Grades**

This course is not graded on a curve. That is, there is not a pre-defined percentage of the class to be assigned to different grade ranges (e.g., x% 4.0, y% 3.5, etc…). It is feasible (although unlikely) that all students can receive the highest grade.

Also, it is not possible to pre-determine what scores on the exams and tests correspond to what grade ranges.

Grades consist of two major components in the percentage defined by the table below and will be determined as objectively as possible. Coding guidelines are an important part of the objective grading of programming assignments. There will not be a subjective component based on class participation.

Decimal grades are computed for each assignment/quiz/exam by linearly interpolating between the grade boundaries. Results of exams, quizzes, and assignments are combined corresponding to the weights shown below.

There will be two major exams in the quarter: a midterm and a final. There will also be one quiz which will reinforce current material being covered and provide students with an early exposure to the level of mastery required to succeed.

Assignments will consist mainly of programming problems – there will be five major programs written. The last will also have a design phase grade associated with it.

At any time I will be happy to consult with you on how you are doing in the class relative to grade
Assignment Submissions

Assignments are to be submitted electronically to Canvas. A link is available on the course Web Site.

Late submissions will penalized unless there has been previous agreement due to extenuating circumstances.

Class Attendance

Class attendance is not mandatory and there is not a grade based upon attendance or participation. However, I strongly encourage you to come to class as there is generally a direct correlation between attendance and your grade. All tests will be given during class. Finally, you will be held responsible for all material covered in class, regardless of its presence (or lack thereof) in the textbook.

Programming Assignments

Programs will be graded upon the follow aspects:

- **Style / Syntax**
  - Following Coding Guidelines
  - Clarity
- **Factoring / Design**
  - Modularity
  - Interface Design
- **Correctness**
  - Determined through test cases run against code
- **Efficiency**

You can use any IDE for coding and testing your laboratory and programming assignments. I will be using Microsoft's Visual Studio 2017 Community Edition and I recommend that you do as well. This version is available free—a link is provided on the class webstie.

Independent of what version you use to develop your programs, there are only two options available for what environment we will use to grade the assignment. The program will be compiled with g++ and executed on a Linux operating system OR on Windows using the Visual Studio 2017.
community edition compiler. You can choose which of these two options you would like the grader to utilize.

If your program does not compile or does not run in either of these two chosen environments it will be deemed incorrect. This is true even if it works with others OS's or compilers. This means that if you use a different development environment you should port and test your code to the prescribed environment before submitting it. The Linux lab is available for this purpose if you do not have a prescribed local development environment. A link to usage of the Linux lab can be found on the class website.

Programs are turned in as soft copies to canvas. Each soft copy must be a zipped file—only zip will be accepted. Each homework specification will indicate what needs to be included in the .zip. However, generally the zip file contains source code (in ASCII Text), the executable, and some results of the program in a report format. Please check each homework specification about what you to write in your report.

For grading the correctness aspect of the assignment the grader will unzip each file, compile the code and execute a number of test cases.

For Our Veterans

If you are a student who has served in our nation’s military forces, thank you for your service. I hope that you feel comfortable enough to confidentially self-identify yourself to me so I can help you make a successful transition from the military to higher education.

For Our Parents

Parenting students are encouraged to take advantage of the resources provided on campus. These resources include the Parent Union at UWB, the Childcare Assistance Program, on-campus Family Friendly Spaces, priority access at Bright Horizons Bothell and Bothell KinderCare, and back-up/sick care at one of these locations. On campus resources include lactation rooms and baby changing stations. For more detailed information, visit https://www.uwb.edu/studentaffairs/resources/student-parents or contact the Parent Union on Facebook: https://www.facebook.com/PUUWB.

Access and Accommodations

Your experience in this class is important to me. If you have already established accommodations with Disability Resources for Students (DRS), please communicate your approved accommodations to me at your earliest convenience so we can discuss your needs in this course.

If you have not yet established services through DRS, but have a temporary health condition or permanent disability that requires accommodations (conditions include but not limited to; mental health, attention-related, learning, vision, hearing, physical or health impacts), you are welcome to contact DRS at 425-352-5307 or uwbdrs@uw.edu. DRS offers resources and coordinates reasonable accommodations for students with disabilities and/or temporary health conditions. Reasonable accommodations are established through an interactive process between you, your instructor(s), and DRS. It is the policy and practice of the University of Washington to create inclusive and accessible learning environments consistent with federal and state law.
Honor Code

Work is to be done independently unless directed otherwise; collaboration of work is not acceptable.

You may discuss the problem statement with each other and help debug, but any actual work to be turned in, must be done without collaboration. In other words, all designing and coding is to be done independently. This class is run by the honor code. By taking this class, you agree that you will not collaborate inappropriately on any work. In some cultures, family relationships and their loyalty are considered above all others. In this course, we are an academic family and you betray the instructor’s and university’s trust should you violate the honor code.

Academic Conduct

Work you submit must be the product of your own efforts. Plagiarism is one of the most common violations of academic integrity. Plagiarism, cheating, and other forms of academic dishonesty, including dishonesty involving computer technology, are prohibited. You are responsible for knowing what constitutes a violation of the University of Washington Student Code, and you will be held responsible for any such violations whether they were intentional or unintentional, so please pay attention to canvas information, assignments, and discussions in class regarding these topics. See http://www.uwb.edu/studentservices/academicconduct. The library also has an extremely useful website with resources at http://libguides.uwb.edu/ai.

UWB Alerts

Sign up for UWB Alerts, to be made aware of possible school closures. For more information on the alert process, see http://www.uwb.edu/alert. If school is closed and class cancelled, I will notify the class with arrangements to handle coursework for any missed classes.
# Tentative Class Schedule

The following is a tentative list of subjects, assignments, and tests that we will cover through the quarter. This will undoubtedly change as we make our way and adjust through the quarter.

<table>
<thead>
<tr>
<th>Date</th>
<th>Topics / In-Class Tests</th>
<th>Reading</th>
<th>Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/3</td>
<td>Introduction, Development Environment, C++ Fundamentals</td>
<td>Carrano Ch. 1, Appendix A</td>
<td>Program 1 assigned</td>
</tr>
<tr>
<td>1/8</td>
<td>Classes, Objects, Abstraction, Encapsulation and OOP in C++</td>
<td>Carrano Ch. 1, C++ Interlude 1.1-1.2</td>
<td></td>
</tr>
<tr>
<td>1/15</td>
<td>Objects, Classes, Encapsulation and OOP in C++, Op Overloading</td>
<td>Carrano Ch. 3.1-3.2, Appendix D, C++ Book</td>
<td></td>
</tr>
<tr>
<td>1/15</td>
<td><strong>Martin Luther King Day</strong></td>
<td></td>
<td>Program 2 assigned</td>
</tr>
<tr>
<td>1/17</td>
<td>Templates and STL</td>
<td>C++ Interlude 1.3, C++ Book</td>
<td></td>
</tr>
<tr>
<td>1/22</td>
<td>Recursion &amp; induction</td>
<td>Carrano, Ch 2, Appendix E</td>
<td></td>
</tr>
<tr>
<td>1/24</td>
<td>Recursion &amp; induction</td>
<td>Carrano Ch 3.3, Carrano Ch 5.3-5.4, Cusack Ch 8.1</td>
<td>Program 3 assigned</td>
</tr>
<tr>
<td>1/29</td>
<td>Stacks</td>
<td>C++ Interlude 2.1-2.3, Carrano Ch 4, 6.1-6.2, 6.4-6.5</td>
<td></td>
</tr>
<tr>
<td>2/5</td>
<td>Arrays / Pointers / memory allocation / Linked-Based Implementations</td>
<td>Carrano Ch 7</td>
<td></td>
</tr>
<tr>
<td>2/12</td>
<td>Algorithms Analysis / Program Complexity</td>
<td>Carrano Ch 10</td>
<td></td>
</tr>
<tr>
<td>2/14</td>
<td>Algorithms Analysis / Program Complexity</td>
<td>Cusack Ch 7</td>
<td></td>
</tr>
<tr>
<td>2/19</td>
<td><strong>Presidents’ Day</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2/21</td>
<td>Sorting Algorithms</td>
<td>Carrano Ch 11</td>
<td>Program 5 assigned</td>
</tr>
<tr>
<td>2/26</td>
<td>Sorting Algorithms</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2/28</td>
<td>Lists</td>
<td>Carrano Ch 12</td>
<td>Program 5 Design Due</td>
</tr>
<tr>
<td>3/5</td>
<td>Trees (BST)</td>
<td>Carrano Ch 15</td>
<td></td>
</tr>
<tr>
<td>3/7</td>
<td>Queues (not Priority Queues)</td>
<td>Carrano Ch 13, 14</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Propositional Logic (time permitting)</td>
<td>Cusack Ch 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Exceptions / Iterators</td>
<td>C++ Interlude 3, 7</td>
<td></td>
</tr>
<tr>
<td>3/12</td>
<td><strong>Final Exam</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>