CMSC 160: Introduction to Algorithmic Design I
Spring 2024

http://marmorstein.org/~robert/Spring2024/cs160.html

Instructor: Robert Marmorstein (marmorsteinrm@longwood.edu) **Phone:** (434) 395-2185
Lecture: 12:00pm – 12:50pm MWF, Rotunda 354 **Lab:** 2:00pm – 3:15pm R, Hiner G11
Office Hours: 2:00pm – 3:15pm MTWF *or by appointment*, Stevens 109

For an appointment outside regular office hours, contact me on Slack at least twenty-four hours in advance.

Course Description: An introduction to problem solving and algorithmic design using an object-oriented programming language. Topics include programming logic, iteration, functions, recursion, arrays, memory management, user-defined data types, abstraction, and complexity analysis.

Prerequisites: This course has no prerequisites. (Previous programming experience is helpful, but not absolutely required.)

Course Student Learning Outcomes: By the end of the course, the successful student will be able to:

– Given a problem, design an algorithm to solve it

– Use variables and control structures (such as if statements, loops, and functions) to implement an algorithm in C++

– Describe appropriate programming practices for ensuring that programs are robust, maintainable, and efficient.

Communications Policy: The best way to reach me is to use **Slack**. Slack is a chat utility with clients for mobile devices and desktop computers. It will allow you to easily send me code snippets. Also, since I get notifications when a Slack message comes in, I am more likely to reply to your message quickly than if you send me an e-mail. Slack also allows me to easily set up a Zoom meeting (or Google Hangout) if we need to video chat.

When you send me a Slack message, I instantly get a notification on my computer, tablet, and phone. Typically, I reply to Slack messages within 24 hours (often sooner) on weekdays and sometimes even on weekends. I am much slower at replying to e-mail (since I do not get a notification and have to explicitly check it). Typically, you can expect a reply to an e-mail within 48 hours (longer on weekends).

Slack is also a good way to communicate with other members of the class. You will be invited to a public #cmsc-160 channel in which you can discuss the projects and other course topics with other students in the class. Feel free to ask for and give help on this channel, but please stick to general answers rather than posting code.

You can also reach out to me by e-mail to marmorsteinrm@longwood.edu. However, please do not send me large files by e-mail. They take up space toward my limited quota on the mail server and cause me all sorts of headaches. **E-mail messages containing large files will be deleted unread**.

Asking for help
If you are asking for help with a project or homework problem, you can send me a direct message through Slack. You should attach your code or your work to a Slack message so that I can see where you are at. You should do this by using the "paperclip" icon to attach the file directly to your message or by copy/pasting the particular snippet of code you are working on to the body of the message.

Please do NOT attach screenshots or pictures taken on your phone. They are hard to read and do not allow me to compile your code without retyping it. Instead, attach the .cpp or .h file directly to the Slack message. You will probably need to do this from a browser running inside your Linux virtual machine (or other Linux system).

One last suggestion: don't "ask to ask". I am delighted to answer questions about the projects and homework assignments and you should feel free to ask questions at any time. Asking permission to ask a question wastes my time and yours.

Course Structure and Student Expectations: You should expect to spend roughly **four and a half hours each week** attending class. You should expect to spend an additional **nine to twelve hours** each week working on projects, reading the textbook, preparing for quizzes and exams, and completing the homework assignments. You may find that you spend significantly less than nine hours on some of the easier topics at the beginning of the course and significantly **more than twelve hours** on some of the harder projects in the middle and at the end of the course. Budget your time carefully so that you are able to complete your work and earn the grade you want.

Textbook: The textbook for this class is "C++20 for Lazy Programmers" by Will Briggs, Second Edition, Apress, ISBN: **978-1-4842-6305-1**. I highly recommend you use the print version of this book rather than the e-book. This will allow you to bring your book to class and is also much more convenient for using it as a reference while programming.

This book introduces many innovations that have been added to C++ in recent revisions of the language. It also contains many examples and details that will go beyond the material covered in lecture. However, in class, I will occasionally introduce a different, simpler, or more common way to do things. For this reason, it is important both to come to class AND to read the textbook.

In addition to the textbook, we will make use of the free Unix Programmer's Manual (sometimes called the "man pages") and the TexInfo documentation (accessible through the "info" command on any Linux system).

These can be retrieved directly from the command line in any Linux system and provide information about the standard programming libraries and the UNIX programming environment we will be using to develop software. There are also online versions of these resources, but they may differ from the actual software installed on your laptop, so it is better to learn to use the command line versions.

Linux Environment: In order to complete the programming assignments for this course, you will need to use a Unix-based open-source operating system such as Linux or BSD. ***You are responsible for getting a development environment set up and working correctly on your system.***

If you already have a working Linux laptop, you are in great shape. If not, you will need to install Linux on your laptop or workstation. To do this, you have several options:

One option is to **install Linux in a Virtual Machine** such as Virtual Box. This is the option I recommend for most students. Virtual Box can be downloaded for free from http://www.virtualbox.org/) and allows you to install a complete Linux environment that can run in a separate window within your existing Windows or Macintosh system. One drawback is that you will need a significant amount of disk space to store the large "hard drive image" file. I recommend 50 to 60 Gb, and 20Gb is probably a bare minimum. Many modern systems use small SSD drives that simply do not have enough space to do this.

Another option is to **install Linux directly onto your hard drive** and set up your system in a "dual boot" configuration that will let you reboot to switch operating systems. If you have extensive computer experience, this is probably your best option. However, it requires skill and care. *Installing an operating system and configuring the boot environment is risky and you should carefully back up your files before installing Linux. Linux installers have become much more user friendly, but if you are not careful it is possible to accidentally wipe your hard drive, deleting all the files.* As with the Virtual Machine option, dual-booting requires a significant amount of disk space, since you will need to set up a second partition your hard drive large enough to hold a complete operating system.

A final option is to **use a Live USB disk** to run Linux without any modifications to your hard drive. This is a relatively safe option and does not require additional hard drive space. In some circumstances, it can be the easiest option. However, you will still need to configure your system's UEFI or BIOS settings to allow you to boot from the USB. In addition to adjusting the boot order, you will likely need to disable the "Secure Boot" setting and may need to turn off "Intel Rapid Start".

The main disadvantage to this approach is that USB drives are very slow (compared to a hard drive or SSD). They are also not designed for heavy usage and can be prone to failures, which could cause you to lose some or all of your work. If you use this option, I strongly recommend that you back your projects up onto another device at regular intervals.

No matter which of these methods you choose, you will need to select a Linux distribution to install and follow the installation instructions to get it working. I recommend Debian (https://www.debian.org/) unless you have previous Linux experience with a different distribution. Since the project handouts are geared toward Debian-based distributions, you will need to adapt the instructions slightly if you elect to install a different one.

Macintosh Computers: Unfortunately, if you have a Macintosh based on Apple's new "M1", "M2", or "M3" chips (i.e. "Apple Silicon"), you may experience difficulty getting a Virtual Machine to work. In the past, some students have reported significant trouble installing Linux on these systems. This was particularly true when the technology was new – and support may have improved as software vendors fix bugs and improve the product. However, if you are on a Mac, you are likely to experience headaches and may not be able to complete all the projects this way. Your best bet may be to borrow another computer from someone (such as the Longwood library).

Course Requirements and Major Assignments: Your grade will depend largely on completion of programming projects, which will comprise 50% of your grade. The remainder of your grade will come from homework assignments and quizzes (30%), the final exam(15%), and participation(5%).

Reading
It is important that you keep up with readings from the textbook. While the reading itself isn't graded, there is material in the textbook that enhances and extends what we cover in lecture. It is important that you master these details in order to be prepared for the homework assignments and (pop) quizzes. Reading assignments for each week are listed on the tentative course schedule (see below). I will expect you to have read the chapter for each week by the Monday of the following week.

Projects
This is a heavily project-driven course. There will be between four and six significant programming projects (plus one "warmup" project) which will constitute the majority of your course grade. For tentative due dates, see the schedule below.

Final Exam
The final exam for this class will be a comprehensive final covering topics from the beginning to the end of the course. It will consist largely of "coding problems" where I give you a short programming task and you give me C++ code that accomplishes that task. However, the exam may also contain short answer, matching, fill in the blank, vocabulary, or even essay questions.

Homework and Quizzes
In addition to projects and the final exam, you will be graded on homework assignments and quizzes. Many of the homework assignments will be in the form of "drills" – worksheets with several similar problems that will help you practice particular programming skills until you are proficient at them. In addition, there will be a test review packet for the final exam that will be worth a significant percentage of the homework grade.

Quizzes will be given in class as unannounced pop quizzes. It is important to keep up with the weekly readings (see tentative course schedule below) so that you are prepared for the quizzes. If you miss class due to an **excused absence** I will allow you to make up the quiz if you contact me within twelve hours of the absence to explain why you had to miss class.

Late Work and Grading Policy: Late work will not be accepted unless you have a serious medical or family emergency which prevents you from completing the assignment on time. In such cases, you do not need a doctor's note, but you must send me ***e-mail*** within twelve hours of the assignment due date to explain your circumstances and to make arrangements for the work to be completed. However, see the section on slip days, below.

Slip Days: You will be allocated a fixed number of slip days at the start of the semester. You may use your slip days to extend the due date of one or more *programming projects*. You can use all of your slip days on one assignment or you may use them over multiple assignments. Slip days are calculated from the minute the assignment is due until you turn it in and are rounded *up* to the nearest integer value. That means that if you turn an assignment in 24 hours and 1 minute late, you will use up *two* slip days. The slip day clock runs over weekends and holidays. If a lab is due on Friday and you turn it in on Monday, you will have used three slip days, not one. Slip days cannot be shared, traded, bought, or sold, but can occasionally be earned by participation in relevant campus activities I select.

Grading Scale:

		100-91:	A	90:	A-
89:	B+	88-81:	B	80:	B-
79:	C+	78-71:	C	70:	C-
69:	D+	68-65:	D		
64 or lower: F		(There is no grade of D- in this course. Anything below a 65% is failing.)			

Attendance: In general, I expect you to attend class unless you are sick or engaged in an approved extracurricular activity. Please do NOT come to class if you are sick. Instead, contact me within 12 hours of the absence to report your illness and make arrangements for completing any missed work. You should also check the course web site for announcements, new assignments, and other important updates. It is your responsibility to make up any missed work and get notes on any material you have missed.

In general, I will not allow students to "stream in" when they are sick or must otherwise miss class, unless they are officially quarantined by the university. Setting up a stream requires me to change how I prepare the lecture and it takes an enormous amount of time to modify my materials to be suitable for that format.

However, if you are officially quarantined by the university, I will try to find a way to stream my lecture. To request this accommodation, you MUST contact me at least 48 hours in advance to make arrangements.

I will rely primarily on your honor for enforcement of the attendance policy. However, I will keep a record of your attendance as required by Longwood policy. In accordance with that policy, I may (at my discretion) penalize you for missing more than 10% of scheduled class time (about 5 class sessions) to unexcused absences. If you miss 25% or more of scheduled class meetings (about 14 sessions), you will automatically fail this course.

Honor Code Statement: I take the honor code very seriously. Any infractions of the honor code will be dealt with harshly. In addition to any penalties imposed by the Honor Board, any student convicted of an honor offense in this class will automatically receive a final course grade of **F**. You should consider all work in this class to be pledged work.

However, I view the honor code primarily as a tool – by establishing clear guidelines for which behavior is admissible in this class, the honor code allows you to properly take advantage of external resources without fear of violating the rules of the course.

Most academic dishonesty falls into two categories:

Plagiarism (Using someone's work or ideas without giving them proper credit)
Cheating (Obtaining an unfair advantage by violating course policy)

Since different professors and even different classes from the same professor may have different rules, it is vitally important that you carefully read this policy and know what is and is not permitted in this particular course.

Here are the rules I expect you to follow:

1. Exams and quizzes are to be completed entirely on your own. You SHOULD NOT use any resources (such as books, web sites, homework assignments) not explicitly permitted at the time of the exam. You should not discuss these assignments with anyone but the course instructor.

2. You MAY discuss homework problems and lab projects with other students subject to these restrictions**:**

A. Your work should be your own original solution to the assigned problem.

The work you submit should, in general, be either your own original work or modifications of material which I have provided. You **MAY** assist other students or get assistance with simple problems like syntax errors, but you **MAY NOT** copy large blocks of code from each other.

A good guideline of what "large" means is that changes that involve one or two lines of code are usually okay, but copying more than three complete statements is usually too much.

Also, only turn in work which YOU have typed or written.

You should never turn in anything which someone else has typed or written as if it were your own.

B. You MAY NOT copy code electronically from other students or online resources

This doesn't mean you can't look online for help with a project. It just means that you shouldn't copy/paste or download code and turn it in as your own. You must re-type any code you find. You should also not be using large blocks of code from the Internet (again, the three line limit is a good rule of thumb), unless otherwise instructed by the professor.

You may not share code with other students using flash drives, cell phones, e-mail, web sites, *floppy disks, CDs, or* ***any other*** *electronic storage or communication device. You* ***MAY*** *print out copies of your code for your personal use, but should not share code listings with other students.*

3. You must give proper attribution.

*Whenever you receive help or use an online resource, you should comment your code to give proper credit. A simple comment like "****/* based on** **http://codewarrior.com** ***/**" is fine. This comment should go directly above or directly after the place that you used the resource or received help to make it clear which parts of your program are not entirely original.*

4. You are responsible for securing your code.

Helping other students to cheat is also cheating. Furthermore, it is your responsibility to make sure that other students do not use your work to cheat. Be careful with who you allow to access your computer or account. Report any missing files, flash drives, or other devices that contain your work to me promptly.

Campus Policies: This class complies with campus policies on wearing of face masks, intellectual property, disability accommodations, mental health, and reporting of crimes and sexual misconduct. For more information, see http://www.longwood.edu/academicaffairs/syllabus-statements/.

Food and Drink: You **MAY** bring non-alcoholic beverages, including soft drinks, to class. However, please **DO NOT** eat in class (it distracts me and the other students). Violations of this policy will be considered an unexcused absence. I grant exceptions to this rule for students who must otherwise forgo lunch or have medical needs that require them to eat in class. If you feel that you need such an exception, you must make arrangements with me in advance (that is, before bringing food to class).

Cell Phones and Laptops: Cell phones, music players, and laptops are to be turned off and put away during class, except as needed for the lab sessions, online streaming, or as directed by the professor. Violations of this policy will be considered an unexcused absence.

Tentative Course Schedule:

Jan. 10 – 12	Introduction: C++ Development in a UNIX environment Algorithms and Pseudocode, Programs, Program Structure, Comments Compiling Programs, Editing Programs with Vim **Read Chapter 1**
Jan. 11	Warmup Project: Writing C++ Programs in Linux using Vim **(Due Jan. 19th by 11:59:59pm)**
Jan. 15	**Martin Luther King Jr. Day, NO CLASS**
Jan. 17 – 19	Using SSDL Literal Types: Floating Point Numbers, Integers, Strings, and Characters Binary and Hexadecimal Named constants, Colors, Text, Fonts, Images, and Sound Functions: sqrt, pow **Read Chapter 2 and Skim Appendices D and E**
Jan. 18	**LAST DAY TO ADD/DROP COURSES**
Jan. 22 – 26	Variables and Constants, Arithmetic Operations, Order of Operations, Casting Mouse Functions Creating Functions, Software Requirements Signed Binary Numbers **Read Chapter 3 and Skim Appendix H**
Jan. 25	Lab 1: Using SDL **(Due Feb. 2nd by 11:59:59pm)**
Jan. 29 – Feb. 2	Conditional Statements, Compound Conditions (AND, OR, and NOT) Chained If Statements and Nested If Statements Strings and String Functions **Read Chapter 4**
Feb. 1	Lab 2: Mouse Functions and Conditional Statements **(Due Feb. 21st by 11:59:59pm)**

Feb. 5 – 9	Loops, Increment, Decrement Loop-and-a-half (break and continue) Formatted I/O Stream Input: Reading from the keyboard Debugging **Read Chapters 5 and 25**
Feb. 8	Lab: Catch up and Review
Feb. 12 – 16	Creating Functions Return values and void functions Software Development: Top-Down Design and Bottom-Up Design Requirements Analysis and Testing **Read Chapters 6 and 7**
Feb. 15	Lab 3: Console I/O **(Due Feb. 23 by 11:59:59pm)**
Feb. 19 – 23	Advanced Functions and Variable Scope Debugging with gdb Random numbers, Boolean Functions, Reference Parameters **Read Chapters 8 and 9 and Appendix G**
Feb. 22	Lab: Catch up and Review
Feb. 26 – Mar. 1	Arrays and Enumerations Passing arrays to functions Multidimensional arrays **Read Chapter 10**
Feb. 29th	Lab 4: Arrays and String Formatting **(Due Mar. 15 by 11:59:59pm)**
Mar. 4 – 8	**NO CLASS: Spring Break**
Mar. 11 – 15	C++ Built-In Templates: Vectors, Sets, and Maps
Mar. 14	Lab: Catch up and Review
Mar. 18 – 22	Structures and Objects, Animation Vectors of Objects **Read Chapters 11 and 12**
Mar. 21	Lab 5: Objects and Animation **(Due Apr. 5th by 11:59:59pm)**
Mar. 25 – 28	Reading from Files Writing to Files Software Testing **Read Chapter 13**
Mar. 27	**Deadline to withdraw with a W**
Mar. 27	Lab: Catch up and Review
Apr. 1 – 5	Classes, Constructors, Accessors, and Mutators Public and Private Member Variables **Read Chapter 15**
Apr. 4	Lab 6: Two Dimensional Arrays **(Due Apr. 12th by 11:59:59pm)**
Apr. 8 – 12	Objects of Vectors, Dynamic Arrays and Pointers Searching Lists **Read Chapter 14**
Apr. 9	**NO CLASS: Symposium Day**

Apr. 15 – 19	Sorting Lists
Apr. 17	**NO CLASS: Research Showcase**
Apr. 18	Lab: Catchup and Review
Apr. 22 – 26	Catchup and Review
Apr. 30	**Final Exam** (Tuesday, 11:30am – 2:00pm, Rotunda 354)