CS 150 Syllabus (Hybrid), Summer 2023

Hello! Welcome to the Orange Coast College CS 150 information page for the **two** Summer 2023 sections of CS 150 which is taught in a **hybrid** format. CS 150 is a **second semester** programming class for Computer Science and Engineering majors who intend to transfer to a 4-year university.

To enroll in CS 150 you must first complete CS 170 (Java Programming) or CS 131 (Python Programming) here at OCC, *or* apply for a prerequisite waiver. You'll find more information on the prerequisite below.

Where and When | Content and Prerequisites | How to Succeed | Textbook and Software | Policies

Who, Where, and When

CS 150 is taught in a compressed - hybrid format this Summer. You will attend an In-Person lecture and testing period once a week. There will be seven mandatory, in-person meetings. (See the schedule for your section.) You must attend all of them. The remainder of the material you will complete on your own schedule, (providing you meet the weekly deadlines), using the resources provided.

- CRN# 11837 (Gilbert) meets in-person on Monday from 1:00 PM to 5:00 PM in MBCC 126
- CRN# 12130 (Gilbert) meets in-person on Tuesday from 1:00 PM to 5:00 PM in MBCC 126

Each in-person class meeting period consists of lecture, hands-on demonstration, programming exams and written exams. You must attend the in-person portion of the class; it is not optional. If you have a scheduling conflict, please do not enroll in the class.

Although you can arrange your at-home learning to meet your own schedule, this is not an "open-entrance, open-exit" course. Each day's reading, exercises and homework, must be completed by the beginning the next week.

Instructor Contact Information

- Email : sgilbert@occ.cccd.edu
- Office Hours: see Canvas

What is CS 150 All about?

Here's the official course description from the OCC catalog:

> *A first course in the ANSI/ISO Standard C++ programming language. Topics include data types, strings, operators, expressions, control flow, storage classes, input/output, functions, pointers, arrays, preprocessor, file I/O, standard library routines, function overloading, object construction and inheritance, object-oriented design and recursion.*
>
> *May be taken for a grade or on a credit/no-credit basis. Three and one-half hours lecture, one and one-half hours non-lecture.*

Prerequisites

At OCC **prerequisites are enforced** for Computer Science classes. You can be cleared in the registration system for enforced Prerequisites and Co-Requisites in one of three ways:

- You must complete (with a "C" or better) one introductory computer programming class we offer here at OCC: **CS 170** (Java Programming) or **CS 131** (Python Programming). You cannot take CS 150 concurrently with the prerequisite class.
- You may successfully complete an **equivalent** course with a "C" or better at another regionally accredited college or university, and submit transcripts to verify successful completion. You may submit unofficial transcripts for prerequisite clearance, but must submit official transcripts if you wish to transfer the course units to OCC. If you believe that you have completed the OCC course prerequisite at another college, complete the online OCC Prequsite Clearance Form and bring your transcripts to Student Records (Enrollment Center,1st Floor, Watson Hall) well in advance of your registration appointment.

- If you have learned the material in some other way, such as through self-study, and feel you are prepared to succeed in CS 150, you may submit a Pre/Corequisite Challenge Form directly to the Business and Computing Division office, in accordance with the Matriculation guidelines established by the State of California.

What's CS 150 *Really* About?

Computer Science 150 is a course in computer programming that uses the ANSI/ISO C++ programming language to teach *structured and object-based programming techniques.*

There are several reasons why you might want to take CS 150:

- If you intend to major in Computer Science, the C++ programming is widely used as the language for operating systems and embedded programming courses. Some universities use C++ as the primary teaching language for all of their courses.
- If you want to learn the advanced object-oriented and generic programming features of the C++ language, you should first learn the basic syntax of C++. CS 150 gives you a solid foundation in C++ syntax, control structures, and fundamental data types.
- If you want to write high-performance programs for Windows or Unix. It's good to know C++, even if you eventually program in Python or C#. Both Unix and Windows have the C/C++ language at its core.
- You want to go into game programming. High-performance games are almost always written in C++, and, if you transfer into a 4-year gaming program, like the CGS program at UCI, you'll need to know C++.

A computer programming course really has two parts. The first part is simply learning the syntax or grammar that a particular language uses. You need to learn the specific keywords and control structures used by each language, as well as the common idioms used to express generic computer algorithms.

The second part, however, is really more interesting. At its heart, a course in computer programming is a course in *thinking* and *problem solving*. It is also a lot of fun. Fred Brooks, the father of modern software engineering, lists five reasons why programming is fun:

1. Because of the sheer joy of making things. "As the child delights in his mud pie, so the adult enjoys building things, especially things of his own design."
2. Because of the pleasure of making things that are useful to other people.
3. Because of the fascination of building complex, puzzle-like objects. "The programmed computer has all the fascination of the pinball machine or the jukebox mechanism, carried to the ultimate."
4. Because of the joy of always learning new things. Programming is ever new.
5. Because of the delight of working in such a tractable medium. As Brooks put it:

> *The programmer, like the poet, works only slightly removed from pure thought-stuff. He builds his castles in the air, from air, creating by exertion of the imagination...Yet the program construct, unlike the poet's words, is real in the sense that it moves and works, producing visible outputs separate from the construct itself. It prints results, draws pictures, produces sounds, moves arms. The magic of myth and legend has come true in our time. One types the correct incantation on a keyboard, and a display screen comes to life, showing things that never were nor could be.*
>
> *Programming then is fun because it gratifies creative longings built deep within us and delights sensibilities we have in common with all men.*
>
> *--Frederick P. Brooks, Jr, The Mythical Man Month, 2nd Ed.*

Goals and Outcomes

The ANSI/ISO C++ programming language is a ***multi-paradigm*** language. That means it can be used to write computer programs using several different *methodologies or techniques*. In this course, C++ Programming I, you will:

- Begin by writing *imperative*, *procedural* or *structured* programs.
- You'll also learn how to write *object-based* and *object-oriented* programs.
- You'll learn about the syntax and semantics of the C++ language, as well as many of the common

"idioms" used by C and C++ programmers, especially those that are different from those used in other programming languages.
- You will learn how to use pointers and how to employ recursion.
- Finally, you will also be introduced to the C++ Standard Template Library (STL) through the **`string`** and **`vector`** classes.

In CS 250, (C++ Programming II, the course that follows this course), you will study more advanced object-oriented and generic programming methods, as well as more of the C++ Standard Template Library. You'll also learn to use the **functional** LISP programming dialect called **Scheme** (or **Racket**).

To help you master these new programming techniques, you will be writing a lot of programs. Each week you will read six short written lessons, and then solve several programming exercises for each lesson. In addition, each week you will complete several programing homework assignments, and take a reading quiz to assess your understanding.

It is important that you ***actually do*** the homework and reading exercises. In addition I encourage you to write small programs on your own to explore the language. You **cannot** passively absorb the material, and expect to pass the class; you must "learn by doing".

The purpose of these programming assignments—and the goals of the course—are that you:

- Understand and can apply the basic concepts of structured programming.
- Can use modern C++ programming tools to produce computer programs written in the C++ programming language.
- Understand and can apply the syntax and semantics of the C++ programming language.
- Gain experience in writing programs, thereby developing skills in program design, coding, and debugging, as well as developing critical thinking and reasoning skills.

Outcomes

Upon successful completion of the course, you will be able to:

- Apply structured and object-oriented programming techniques to solve various problems using the C++ programming language.
- Produce programs correctly using each of the different flow-of-control constructs.
- Produce programs which show an understanding of how to pass data between different modules without resorting to global variables.
- Demonstrate skills in using different forms of arrays and structures to solve problems.
- Design and implement programs that use sequential and random file concepts.
- Differentiate between the different data storage classes used in C++ programs.
- Apply the concept of structures to solve various problems.
- Compare and contrast the different fundamental data structures used in Computer Science.

The Student Learning Outcomes or SLOs for CS 150 describe things you are expected to be able to **do**. After completing this class you'll be able to:

1. Use C++ to solve various problems that incorporate the use of correct data types, arrays, vectors, loops, branching, operators, functions, recursion, dynamic memory allocation and sequential file I/O concepts.
2. Use C++ to solve various problems that use object-oriented methodology by defining classes and using inheritance and polymorphism.

How can you succeed in CS 150?

Everybody learns in different ways and at different speeds. CS 150 is a 4-unit class, designed to take 216 hours to complete, (if you want a B/C) grade. This works out to about **31 hours a week** during the compressed summer session. You'll spend 4 of those hours each week in class with your instructor, which leave you with about 27 hours on your own. If you're a good student or you're satisfied with a lower grade, you may get by with less. If you have difficulty with the material, or if you want to receive an A in the course, you'll probably have to spend more time.

You'll find that the material in CS 150 is **very sequential**. You can't really succeed by staying up all night every Sunday. Instead, you need to budget your time. You'll need to spend at least **seven hours a day**, Tuesday through Friday, (or Wednesday through Saturday), to get through the course work. (On Monday, you'll attend class for four hours in person). Spreading out your time will really help you to absorb the material better and quicker. For most students, the best way to learn to program is to actually

write programs.

Reading and Lecture

You will watch 28 video lectures (1-2 hours) and complete a lecture exercise for each lecture. For every lecture, you will also read a chapter in the Course Reader, and complete a set of short, hands-on exercises. Each week's lectures, reading and reading exercises are due **before 1pm** on the following Monday or Tuesday (depending on your section).

During the in-person class meetings, your instructor will preview the topics covered in the next set of lessons, but much of the time will be given to exams.

Programming & Written Examinations

Programming is a **skill to be mastered**. To make sure that you've mastered the programming topics, you'll take **ten hands-on programming exams**. These are closed-book, closed-notes, programming exams where you will be asked to demonstrate your proficiency with different aspects of C++.

You will have 45 minutes for each exam, strictly timed. The programming exams will be given during the scheduled in-person class meetings.

There will also be **four written exams—three midterms and a final**. The written exams may have both multiple-choice questions as well as short-answer and fill-in type questions. The written exams will be given on Canvas.

- The written exams are closed-notes, closed book.
- You'll have one hour for each midterm and two hours for the final.
- The final exam is comprehensive.
- You **must** take the final to pass the class.
- There are no makeups for the written exams.

If your final exam percentage grade is greater than your average midterm exam percentage, then the final exam (percentage) score will replace your midterm score.

All exams will take place during the regular in-person class session.

What about programming assignments?

Learning to program is a lot like learning to play the tuba; you'll never be any good unless you practice. To help you do this, you'll spend **most** of your out of class time, and a portion of each class period, writing programs. Each week you'll complete several different programming problems from the material we've covered that week.

You'll complete these programming assignments as you read the textbook. Often the textbook will go over the solution, so you should be able to get 100% on the homework if you put in the time.

You must complete each assignment by its due date. You cannot makeup missed homework assignments after the fact. (You can work ahead, however.) You'll generally receive a with a test program, so you can "grade" yourself after you've finished the problem.

When you are finished with each homework assignment, you'll submit a "completion code" generated by the test program.

Class Discussion

One of the greatest learning resources at your disposal are the other students you'll meet in this class. Turning to StackOverflow (or Google) for help with your questions is not the best way to get them answered. The answer you receive (if any) is often wrong. Instead, take advantage of the fact that you have forty (or more) students who are facing exactly the same problems, and asking exactly the same questions you are asking.

To encourage you to help each other, use the online discussion area to ask **all of your technical questions**, instead of sending an email message. Let's keep email for "personal" correspondence—things like "I'll be missing class this Tuesday", instead of questions like "Why do I get this error message when I compile?" If you ask these types of questions on the discussion board, you'll help all the other students who are having the same problem, but didn't think to ask. (Your instructor will answer questions posted on the discussion board, after your fellow students have had an opportunity to respond. Your instructor also reads all of the messages on the discussion board.)

Class announcements and general tips will also be posted on the discussion board, as well as resources such as slides and starter files. Check the board regularly, (several times each week), for new announcements.

How will I be graded?

You may take this course for a grade, or on a P/NP basis. If you take the course for pass/no-pass, you must obtain a score equivalent to a "C" to get credit. (If you're transferring, though, make **sure** that the institution you're transferring **to** will accept P/NP; most institutions won't if it's a requirement for your major.) To receive a P/NP grade, you simply make a selection using MyOCC. (You no longer have to go to the Admissions and Records office.) However there is still a cutoff deadline to make your selection.

Cautionary Note: starting in 2019, UCI requires CS admits to get at least a B in every lower-division course in the major. If you are a CS major, and you intend to transfer to a UC, and you don't have at least a B at the drop deadline, you probably should take a W and try again later.

Grades will be assigned based on the following weights:

10 hands-on Programming Exams (50 points each)	500
3 Midterm Exams (50 points each)	150
Final Exam (Comprehensive)	200
Weekly Reading Exercises	100
Programming Homework Assignments	50
Lesson Exercise Documents	50

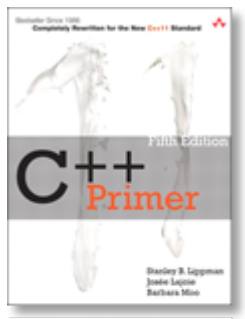
As you can see, this adds up to **105%**. Of course, if you saw the "Homer at the Bat" episode of the Simpsons, you know **that's impossible; by definition, the most anyone can earn is 100%**. So, you'll be graded using 100% as the maximum, leaving the extra 5% for "extra credit" or emergencies. Note, however, that this means that if you have 1020 points at the end of the semester, you will have 102% in the class. I do not add any *extra* points at the end.

Letter grades cutoff percentages: 90%->A, 78%->B, 62%->C 50%->D

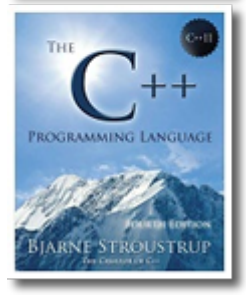
There will be no additional curve or extra-credit.

What textbook and software will I need?

The reading for this class will be the online lessons from the **CS 150 Course Reader** which to which will be **available for free** on Canvas. If you want a printed reference text, I recommend two of them.



Stanley Lippmann's **C++ Primer**, 5th Edition. This covers the features in the new C++11 standard. The bookstore will have a few copies of the book; you can also get it at Amazon for about $40.



The definitive reference is Bjarne Stroustrup's **C++ Programming Language**. Make sure you get the 4th Edition, which covers C++ 11. On Amazon (today, right before Christmas), it was also $40, but I paid $60 for mine earlier in the year. On Canvas, I have a link to an early draft of the first five chapters, which was published separately as *A Tour of C++*.

For those of you who feel the need for a little refresher, I've also posted a link to an alternative CS1 C++ textbook on the Canvas home page.

Software

This semester we will be teaching the course using the UNIX/Linux cloud-based servers provided by Harvard's CS50 infrastructure. We be using **GitHub Classroom** to store your source code. You will be able to log into your workspace from any computer and will not have to have any software installed locally.

You will set up and configure these accounts during the first in-person class session.

What are the "rules" for this course?

I don't accept late work on your homework. You must turn it in by the due date. You may turn in quizzes, reading exercises, and in-class exercises up to one day late for a penalty of 1% per hour. I will provide two "makeup" days (after the midterm and after the final) where you can take up to **four** of the programming exams you missed, (or **retake** those you did poorly on.)

If your percentage score on your written final exam is better than your combined midterm percentage, I'll "throw out" the midterm score, and use your final exam only for calculating the written exam portion of your final grade.

Attendance and Withdrawal

I will take attendance, and you are expected to attend every in-person class session. It is your responsibility to withdraw from any class. If you stop attending, yet fail to withdraw, you will receive a grade of 'F'.

Please pay special attention to the OCC drop deadlines. Every semester I have students who want to drop the class after the deadline to drop has already passed. Those students end up getting an 'F' in the class if they are unable to complete the coursework. **I do not give Incomplete grades.**

Academic Honesty

Programming by its nature relies on learning from the work of others. You are encouraged to examine each other's code and to discuss various approaches and coding styles. You won't really learn anything, though, if you don't "try it yourself".

In this regard, discussing the general method used to solve a problem is certainly encouraged. Taking another student's source code and modifying it is really counter-productive; you learn to program by working out the programming exercises and assignments. If you don't do the assignments yourself, you won't learn how to program, and you won't be able to pass the programming exams.

When it comes to testing time, I expect you to maintain a high level of academic honesty. Cheating on exams will result in course failure.

Disruptive Behavior

You are entitled to an environment that encourages learning, as are all your fellow students. You should not behave in a manner that negatively impacts other class members. In a classroom, such behavior includes hostile behavior such as yelling and screaming, as well as interrupting other students and attempting to inappropriately dominate the classroom. In our "online classroom", disruptive behavior includes "flaming" or harassing email, as well as posting offensive material on the class discussion board.

I expect all of you to be polite, respectful, and helpful to your fellow students; in short, I expect you to act like adults should act. If, in my judgment, your behavior negatively impacts the rest of the class, you may be subject to disciplinary action.

Disabilities

If you have a disability that may impede your ability to successfully complete this course, you should contact the Disabled Student's Center (432-5807 or 432-5604 TDD) not later than the first week of the course. Their staff will assist you in arranging accommodations that can help you meet course requirements.

Reservation of Rights

I reserve the right to change this syllabus, including, without limitation, these policies, without prior notice.
