Course Syllabus
ECE2800J Programming and Elementary Data Structures
Fall 2025

Course Description:

This course is an introduction to programming and provides a foundation for data structures. The emphasis of this course will be techniques and principles to quickly write correct programs. This course is not simply a course about programming. Rather, it focuses on concepts and methods underpinning the C++ language. Specific topics of this course include: basics of Linux, procedure abstraction, abstract data type, inheritance, dynamic memory management, template, polymorphism, linked list, stack, and queue.

Instructor:

Weikang Qian
Email: qianwk@sjtu.edu.cn
Office: Room 430, Longbin Building
Office hour: Thursday 5:00 pm – 6:00 pm and Friday 5:00 pm – 6:00 pm, or by appointment

Teaching Assistants:

1. Wang, Yingjie
   Email: jenny_wyj@sjtu.edu.cn
2. Xue, Tianle
   Email: 1640652412@sjtu.edu.cn
3. Yang, Jingwen
   Email: yjw1520@sjtu.edu.cn

Textbook (Recommended but not required):

1. *C++ Primer, 4th Edition*, by Stanley Lippman, Josee Lajoie, and Barbara Moo, Addison Wesley Publishing, 2005.
2. *Problem Solving with C++, 8th Edition*, by Walter Savitch, Addison Wesley Publishing, 2011.
3. *Data Structures and Algorithm Analysis*, by Clifford Shaffer. Online available: http://people.cs.vt.edu/~shaffer/Book/C++3e20120605.pdf

Class Webpage:

Log into Canvas at https://oc.sjtu.edu.cn/courses/85906. Lecture slides, assignments, and grades will be posted on the class webpage. Check also Piazza for discussions.

Course Prerequisites:

Prior programming experience using C and C++ (preferably in ENGR1010J or ENGR1510J).

Grading Policy:

There will be about 7 coding exercises, 5 programming projects, one midterm exam, and one final exam. The grading distribution is:

- Random pick for answering question: 2%
- In-class quiz and random attendance record check: 3%
- Coding exercises: 10%
- Projects: 40%
- Midterm Exam: 20%
- Final Exam: 25%

For details of "random pick for answering question" and "In-class quiz and random attendance record check", please check Pages 8 and 9 of 01-introduction.pptx on Canvas.

Any questions about the grading of the coding exercises, projects, or exams must be brought to the attention of your TAs and the instructor **within one week** after the item is returned.

Programming Assignments

The programming assignments here refer to both coding exercises and projects.

1. Programming Environment

We require you to develop your programs on Linux operating systems using compiler g++. We will grade your programs in the Linux environment: they must compile and run correctly on this operating system.

2. Assignment Grading

Each programming assignment (*i.e.*, coding exercises and projects) will usually be evaluated along three dimensions: behavioral correctness, adherence to course principles, and general coding style.

A program is "correct" if it behaves as specified in the project/exercise description. Project/exercise descriptions in ECE2800J should clearly spell out all required behaviors, though sometimes we do make mistakes. If you believe something in the specification is inconsistent or not sufficiently detailed, please let us know.

For any given specification, there are many possible programs that behave correctly. Some of those programs will make use of the principles we discuss in this course, while others will not. To receive full credit, you must write your programs in accordance with these principles.

It is our belief that programs are meant to be read by other programmers, not just executed. Codes that are generally unreadable will not receive full credit. You should choose a good programming style and follow this style consistently throughout your code.

Moreover, an important skill the projects/exercises teach is how to show that your code exhibits exactly the behaviors required by the specification. So, while we will sometimes give you a few simple test cases and their results to try out, we will not give you all of the test cases we plan to use to evaluate correctness. Instead, you are required to come up with as many test cases as you can.

Finally, please note that we grade the **last submission** of you, not the one with the highest score.

3. Due Date

Each programming assignment will be given a due date. Your work must be turned in by 11:59 pm on the due date to be accepted for full credit.

For projects, not coding exercises, we still allow you to submit your solution within 3 days after the due date, but there is a late penalty. The late penalty is listed in the following table. For example, if you submit your solution late by 23 hours and 40 minutes, your grade will be scaled by 80%. If you submit your solution late by 24 hours and 10 minutes, your grade will be scaled by 60%. No work will be accepted if it is more than 3 days late—you will receive a zero.

h Hours Late	Scaling Factor
$0 < h \leq 24$ hours	80 %
$24 < h \leq 48$ hours	60 %
$48 < h \leq 72$ hours	40 %

In very occasional cases, we accept deadline extension request. **Deadline extension requests will only be considered if you contact the course instructor (not TAs!) in person.** Such requests will normally only be considered if they are made before the assignment is due, and will only be granted for documented medical or personal emergencies that could not have been anticipated. If you cannot contact the instructor in advance due to the emergency, then contact him as soon as you possibly can. In all cases, you will be required to substantiate any extension request with written proof of the emergency. Extensions are not granted for reasons such as accidental erasure/loss of files and outside conflicting commitments. In order to prevent accidental erasure/loss, make sure that you make copies of your code frequently.

Exam

The exams will be closed-book ones. No electronic devices are allowed in the exams.

You are expected to take both exams at the scheduled times. If you miss an exam, and a medical or personal emergency is not involved, you will receive a zero for that exam. If you anticipate an exam in another course, you must notify the instructor at least one week before the exam date.

Academic Integrity:

1. All students are expected to attend all of the classes.
2. All programming assignments must be done by yourself. You may discuss the assignments in oral

with other students. However, you may not read/copy others' solution. In all cases in which we have reason to believe that cheating has occurred, we will submit relevant materials to the Honor Council for evaluation.

3. You cannot share coursework (including solution codes, test cases, exam solutions, etc.) with others either during or **after the semester**, including making it publicly available in any form (*e.g.*, a public GitHub repository). Note that you may not share test cases with others, as we consider your test cases as part of your solution.
4. You cannot use large language model (LLM)-based service/software, *e.g.*, GPT.
5. Exams will be given under the GC's Honor Code and will require individual effort.
6. You are required to take reasonable precautions to ensure that your own work remains confidential.
7. Teaching and learning materials, such as lecture slides, assignment descriptions, your solutions, quizzes, videos, exam problems, *etc*. are copyrighted and cannot be passed on to others without the permission of the course instructor.
8. Some other points mentioned from Pages 18 to 22 of 01-introduction.pptx on Canvas.

Teaching Schedule (Subject to Change)

Lecture	Date	Teaching Activities (Topics and Exams)
1	09/16	Introduction;
2	09/18	Linux;
3	09/19	Linux;
4	09/23	Developing and Compiling Programs on Linux;
5	09/25	Developing and Compiling Programs on Linux;
6	09/30	Review of C++ Basics;
7	10/09	Const Qualifier;
8	10/14	Procedural Abstraction; Recursion;
9	10/16	Function Pointers; Function Call Mechanism;
10	10/17	Function Call Mechanism; Enum Types; Program Arguments;
11	10/21	I/O Streams;
12	10/23	Testing;
13	10/28	Exceptions;
14	10/30	Abstract Data Types (ADT);
15	10/31	ADT Efficiency;
16	11/04	Midterm Exam;

17	11/06	Subtypes and Inheritance; Virtual Function;
18	11/11	Virtual Function; Interface;
19	11/13	Invariant; Dynamic Memory Allocation;
20	11/14	Dynamic Arrays; Default Arguments;
21	11/18	Destructor; Deep Copy;
22	11/20	Deep Copy; Dynamic Resizing;
23	11/25	Linked Lists;
24	11/27	Linked Lists; Templates;
25	11/28	Container of Pointers;
26	12/02	Polymorphic Containers; Operator Overloading;
27	12/04	Operator Overloading; Linear Lists; Stacks;
28	12/09	Queues; Standard Template Library (STL) Sequential Container;
29	12/11	STL Sequential Containers; STL Associate Containers;
30	12/12	STL Associate Containers;
	TBD	Final Exam;