Table of Contents

In this Chapter:

Procedural programming, the subject of CS 124 preceding this class, is a style of programming where the focus is on the function. This includes how to subdivide a program into functions (in a process called modularization) and what goes on inside functions. Virtually any programming project can be completed with procedural programming tools. Object-Oriented programming, on the other hand, is a style of programming where the focus is on the class. A class is a construct containing a collection of functions as well as the data associated with them. Topics such as encapsulation (how to create well-designed classes), inheritance (the relationship between classes), and polymorphism (working with many versions of related classes) form the backbone of Object-Oriented programming. This class will teach Object-Oriented programming in C++ as well as software development methodologies that enable programmers to work with large projects and many individuals.

Goals

By the end of this semester, you will be able to:

- Generate a design document describing an approach to solve a given program definition
- Predict the output of Object-Oriented C++ code
- Identify syntax errors
- Identify code matching a given output example
- Identify code matching a given output description
- Identify the best design for a given problem
- Write Object-Oriented C++ code conforming to a problem definition
- List and define the terms and concepts of Object-Oriented design

These goals will be explored in the context of C++ using the Linux operating system.

Course layout

This course will be broken into four units:

1. **Using Classes**. This is a preparatory unit, ensuring we have the programming skills to learn ObjectOriented programming and to lay the foundation for the topics to come.
2. **Encapsulation**. Our first undertaking into the world of Object-Oriented programming, the encapsulation unit introduces us to classes. Here we will learn how to design a program with classes and create our first class.
3. **Inheritance & Polymorphism**. With knowledge of how to create and use classes, the next unit is concerned with leveraging the relationships between similar classes to minimize code duplication and to provide a new perspective on class design.
4. **Abstract Types**. After spending virtually the entire semester concerning ourselves with designing and creating new data types, the final unit is about defining functions and classes that operate independent of the data type passed to them.

How to use this textbook

This textbook is closely aligned with CS 165. All the topics, problems, and technology used in CS 165 are described in detail with these pages.

Sam and Sue

You may notice two characters present in various sections of this text. They embody two common archetypes representative of people you will probably encounter in industry.

Sue's Tips

Sue is a pragmatist. She cares only about getting the job done at a high quality level and with the minimum amount of effort. In other words, she cares little for the art of programming, focusing instead on the engineering of the craft. Sue's Tips tend to focus on the following topics:

- Pitfalls: How to avoid common programming pitfalls that cause bugs
- Effort: How to do the same work with less time and effort
- Robustness: How to make code more resistant to bugs
- Efficiency: How to make code execute with fewer resources

Sam's Corner

Sam is a technology nerd. He likes solving hard problems for their own sake, not necessarily because they even need to be solved. Sam enjoys getting to the heart of a problem and finding the most elegant solution. It is easy for Sam to get hung up on a problem for hours even though he found a working solution long ago. The following topics are commonly discussed in Sam's Corner:

- Details: The details of how various operations work, even though this knowledge is not necessary to get the job done
- Tidbits: Interesting tidbits explaining why things are the way they are

Neither Sue's Tips nor Sam's Corner are required knowledge for this course. However, you may discover your inner Sam or Sue and find yourself reading one of their columns.

Needing Help

Occasionally, each of us reaches a roadblock or feels like help is needed. This textbook offers two ways to bail yourself out of trouble.

If you find you are not able to understand the problems we do in class, work through them at home before class. This will give you time for reflection and help ask better questions in class.

If you find the programming problems to be too hard, take the time to type out all the examples by hand. Once you finish them, work through the challenge associated with each example. There is something

about typing code by hand that helps it seep into our brains. I hope this helps.