CPSC 427a: Object-Oriented Programming

Michael J. Fischer

Lecture 1
August 30, 2012

About This Course

Where to find information

All information about this course is posted on the course website:

http://zoo.cs.yale.edu/classes/cs427/2012a/

There you will find:

- Syllabus.
- The main textbook Exploring C++ by Alice Fischer.
- Lecture notes.
- Code samples.
- Homework assignments.

The syllabus contains important additional information.
Read it!

Course mechanics

You will need a Zoo course account. Get it now!
You can't submit work without it.

Submit your assignments on the Zoo using the script in `/c/cs427/bin/submit`. Remember to give the problem assignment number as the first argument to `submit`.

I recommend that you buy the book Herbert Schildt, *C++: The Complete Reference, 4th edition*. It serves as a basic text for C++ as well as a reference manual. It is available in electronic formats as well as in hardcopy.

Course Requirements: Homework assignments (∼50%), midterm exam (∼20%), final exam (∼30%).

Topics to be Covered

Major Areas

1. Foundations (basics of objects and classes)
2. Reusable software design (both language support and design technique)
3. Programming for reliability
4. Programming for efficiency
5. Software toolset and framework design

Course goals - practical

- Learn how to follow instructions, and how question them if you think they are wrong.
- Learn how to get a big job done one module at a time.
- Learn how to use a reference manual.
- Learn how to design for efficiency and reliability.
- Learn how to test, analyze, and debug code.
- Learn how to present your work in a professional manner.
- Become proficient at C++ programming, starting with a knowledge of C.
- Learn how to use UML (Unified Modeling Language) to document your work.

Course goals - conceptual

- What object-oriented programming is – and isn't.
- The basic principles and patterns of object oriented design.
- Learn how C++ differs in syntax and semantics from standard ISO C on the one hand and from other languages with support for OO-programming such as Python, Ruby, and Java.
- Learn about classes, objects, type hierarchies, templates, and their implementations in C++.
- The principles behind the exception handler and how to use it.
- Learn how to use class libraries such as the C++ standard template library (STL), GTKmm, boost, etc.

Kinds of Programming

Two views of programming

People program for different reasons.

Programming is ...

1. A means to solve computational problems;
2. The process of software construction.

Problem solving

Desired properties of programs for solving problems:

- Correct outputs from correct inputs
- Succinct expression of algorithm
- Simple development cycle

Beginning programming courses tend to focus on programs to solve small problems.

Software Construction

Desired properties of software constructed for widespread use:

- Correct outputs from correct inputs
- Robust in face of bad inputs; stable; resilient
- Economical in resource usage (time and space)
- Understandable and verifiable code
- Secure
- Easily repurposed
- Easily deployed
- Maintainable

This course will focus on constructing **large-scale** software.

Programming in the large

- Thousands of lines of code
- Written by many programmers
- Over a large span of time
- Deployed on a large number of computers
- With different architectures and operating systems
- Interacting with foreign code and devices

C++ Programming Standards

Four commandments for this course

From Chapter 1 of Exploring C++ and elsewhere:

1. Use C++ input and output, not C I/O, for all assigned work.
2. Don't use global variables. If you think you need one, ask for help. Your class design is probably defective.
3. Don't use setter functions.
4. Don't believe a lot of the rules of thumb you may have learned in a Java course or that you read on the internet.

Can is not the same as should!

From Chapter 1 of Exploring C++:

- C++ is a very powerful language, which, if used badly can produce projects that are badly designed, badly constructed, and impossible to debug or maintain.
- Your goal is to learn to use the language well, and with good style.
- Please read *and follow* the style guidelines in Section 1.2.
- Download the two tools files from the website.
- Read Section 1.3, about the tools library, and use this information to customize your own copy of the tools.

Rules for preparing your work

1. Every code file you submit must contain a comment at the top giving your name, the course number, and the assignment number.
2. If your work is based on someone else's work, you *must* cite them at the top of the file and describe what parts of the code are theirs.
3. If you have started from a file that you obtained from someone else and it contains authorship/copyright information, you must leave that information in place.
4. If you have any questions about the proper way to cite your sources, *ask*, don't just guess.

Rules for submitting your work

1. All submissions must be done from your Zoo account.
2. Test every line of code you write. It is your job to prove that your entire program works. If you submit a program without a test plan and test output, the TA will assume that it does not compile and will grade it accordingly.
3. Compile and test your program on the Zoo before submission.
4. Supply a `Makefile` with your code so that a grader can type `make` and your code will compile and be ready to run.
5. Supply a `README` file that contains instructions on how to run and test your code.