CSE 165/ENGR 140 Intro to Object Orient Programming

Lecture 1 - Introduction to Objects

CSE 165/ENGR 140: Spring 2022

- Principles that you learn in this class will be applied throughout your career
- This class is fundamental to becoming a good software engineer
- My ultimate goal is to help each of you:
  - Become a solid software engineer
  - Get a good job after you graduate
  - Become a better you

About me: Ammon Hepworth, PhD

- Grew up in San Diego, lived in UT, CT, TX, Hong Kong
- Married with 2 kids (wife from Merced)
- Developed software since 2007
- Former CEO of Jurybox Technologies
- BS, MS and PhD from Brigham Young University

TA Intro

- Hoa Nguyen
- Ghazal Zand

About you (Zoom Poll)

- Where are you from?
- What's your major?
- What Programming languages do you know?
  - Java, C, C++, Python, C#, Visual Basic, JavaScript, HTML/CSS

Contact Info

- Lecturer
  - Ammon Hepworth
  - Email: ahepworth@ucmerced.edu
  - Office: SE2 278
  - Office Hours: Tuesdays at 10:30 – 11:30am
- Teaching Assistants
  - Hoa Nguyen, hnguyen257@ucmerced.edu
  - Ghazal Zand, gzand@ucmerced.edu

Course Overview

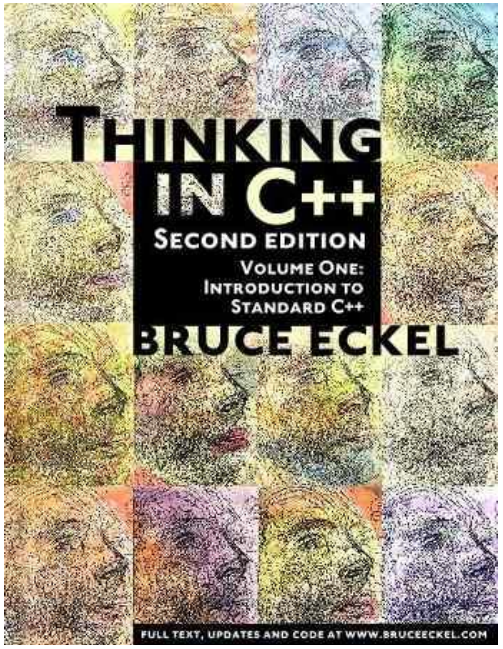
- CatCourses
  - Check regularly for announcements.
- 2 Lectures and 1 Lab per week
- Mid-term exam in class (March 29, tentative)
- Final exam on last day of class (May 5)
- Project presentation during final exam slot (May 10)

Course Objectives

- Create programs in Linux
- Learn C and C++
- Develop good programming habits
- Understand the concept of object-oriented programming
- Labs:
  - Giving each other help in finding bugs and in understanding the assignment is perfectly acceptable.
  - You may allow other students to see small portions of your code on-screen as an example, but you may not allow them to copy directly (or give them copies of your code)
  - We will be using C++; you can use any operating system

Course Material

- Text Book:
  - Bruce Eckel - *Thinking in C++: Introduction to Standard C++*, 2nd Edition, Volume 1, 2000, Prentice Hall
  - https://www.micc.unifi.it/bertini/download/programmazione/TICPP-2nd-ed-Vol-one-printed.pdf
- Online resources:
  - http://www.cplusplus.com/doc/tutorial
  - PDFs after each lecture in CatCourses


THINKING IN C++
SECOND EDITION
VOLUME ONE: INTRODUCTION TO STANDARD C++
BRUCE ECKEL
FULL TEXT, UPDATES AND CODE AT WWW.BRUCEECKEL.COM

Prerequisites

- CSE 031, CSE 100 and MATH 024
- Math: logarithms, series, Boolean logic, matrices, calculus …
- Coding: basic programming experience (Java, C, C++)
- Curiosity: observe how the world is run by computers, and what problems we face.

Grading

Lab assignments:	25%
Participation:	5%
Quizzes:	5%
Mid-term:	20%
Final exam (comprehensive):	25%
Project:	20%

Policies

- Do:
  - Ask applicable questions on Zoom chat
  - Raise hand in Zoom to get attention
  - Participate
  - Have fun
- Do not:
  - Copy someone else’s code
  - Give your code away
  - Outsource your assignments
  - Cheat

Don’t be a cheater!

- Communicating information to another student during examination.
- Knowingly allowing another student to copy one’s work.
- Offering another person’s work as one’s own.
- **You are a better than that**

You can do hard things

- Programming is hard
  - Learning another language
  - Learning new concepts
  - Applying mathematics
- Ask a lot of questions
- Just keep trying

https://youtu.be/KdxEAt91D7k

Hints for success

- Attend lectures
- Attend labs
- Read the textbook
- Do & understand the labs YOURSELF
- Take notes while reading and in lectures
- Ask questions

History Lesson

- C developed by Dennis Ritchie at AT&T Bell Labs in the 1970s.
  - Used to maintain UNIX systems
  - Many commercial applications were written in C
- C++ developed by Bjarne Stroustrup at AT&T Bell Labs in the 1980s
  - Overcame several shortcomings of C
  - Incorporated object-oriented programming
  - C remains a subset of C++

History Lesson

- Where did C++ come from?
  - Derived from the C language
  - C was derived from the B language
  - B was derived from the BCPL (Basic Combined Programming Language)
- Why the ‘++’?
  - ++ is the post-increment operator
  - Therefore, C++ is C, ++

Object oriented programming (OOP)

- Everything is viewed as an object
- A program is a bunch of objects telling each other what to do by sending messages
- Each object has its own memory made up of other objects
- Every object has a type
- All objects of a particular type can receive the same messages

Object oriented software goals

- Robustness
  - How well can it handle errors?
- Adaptability
  - How portable is it on different hardware and operating systems?
- Reusability
  - How much code can be reused in other applications?

Object oriented concepts

- Encapsulation
  - The ability to package data with functions allows you to create a new data type
  - Example: members are encapsulated in a class/structure
- Implementation hiding (same as data/information hiding)
  - Access control
  - To prevent important data from being corrupted
- Interface
  - It establishes what requests you can make for a particular object
  - It is an abstraction of an object
  - It tells what an object does without telling the details (ex. header files).

Good Programming Practices

- Good programmers format programs so they are easy to read
- Good programmers typically:
  - Place opening brace ‘{‘ and closing brace ‘}’ on a line by themselves
  - Indent statements
  - Use only one statement per line
  - Use intuitive object names
    - e.g. int count, instead of int c

C++ Compiler

- C++ compilers accepts almost any pattern of line breaks and indentation
- However, this invites bad programming practices
- We don't want to learn bad programming habits, they are hard to unlearn

Very Simple Program

```
#include <iostream>
using namespace std;
int main()
{
        int classNumber = 165;
        cout << "Hello world!\n";
        cout << "Welcome to CSE ";
        cout << classNumber;
        cout << "!\n";
        return 0;
}
```

Output:

```
Hello world!
Welcome to CSE 165!
```

Things to notice about the example

- Variables are declared **before** they are used
  - Typically, variables are declared at the beginning of the program
- **Statements** (can be multi-line) end with a semi-colon
- The #include directive: **#include <iostream>**
  - Tells compiler where to find information about items used in the program
- **iostream** is a library containing definitions of **cin** and **cout**

C++ Syntax

- using namespace std;
  - Tells the compiler to look for methods and data types in the “std” **namespace**
  - A **namespace** allows us to have methods, classes, and data types with the same name that exist in separate “namespaces” (more detail about it later in the semester)
- In C++, our program begins with a main() method:
  - int main()
- Which returns an integer value at the end of the its execution (optional in many compilers):
  - return 0;

C++ Syntax Highlights

- By now you've probably noticed that C++ looks a lot like Java, though not identical by any means
- That means a lot of your old knowledge of simple logical structures (do, while, for, if, else, etc.) will transfer
- When the compiler fails, it will try to give you a meaningful error message
- However, sometimes they're hard to understand, so be patient

Writing C++ Code

- C++ source code can be written with a text editor
  - **gedit** is popular in Linux as well
  - **nano** is simple with less functionality
- Don’t need an IDE, but it can help
  - Visual Studio
  - NetBeans
  - Eclipse
- The compiler on your system converts the source code to object code
- The linker combines all the object code into an executable program

Reading assignment

- Reading assignment
  - Chapter 1 and 2 of textbook
- No lab this week