CSE 165/ENGR 140
Intro to Object Orient Program

Lecture 1 - Introduction to Objects
Lecturer

- Chi Yan “Daniel” Leung
- cleung3@ucmerced.edu
- Office Room: AOA 146
- Office Hours:
  - M 1:00-3:00pm
  - T/R 10:30-12:30pm
  - By appointment

TA

- Angelo Kyrilov
- akyrilov@ucmerced.edu
Course Overview

- UCMCROPS
  - Check regularly for announcements.
- 2 Lectures and 1 Lab per week
- 1 Mid-term exam (March 17, tentative)
- Final exam (May 7)
- Project presentation (May 12)
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++, specifically the GNU compiler tool-chain (g++ and gdb).
- All code will need to run on a Linux OS.
Course Material

- **Text Book:**
  - [http://www.mindview.net/Books/TICPP/ThinkingInCPP2e.html](http://www.mindview.net/Books/TICPP/ThinkingInCPP2e.html)

- **Online resources:**
  - PDFs after each lecture from CROPS
Prerequisites

- CSE20/21 combination or equivalent knowledge
- 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.
Hints for success

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

- Lab assignments: 30%
- Mid-term: 25%
- Final exam (comprehensive): 25%
- Project: 15%
- Quizzes: 5%
Grading

- Fixed Scale
  - A+ 95-100%
  - A 90-94%
  - A- 85-89%
  - B+ 80-84%
  - B 75-79%
  - B- 70-74%
  - C+ 65-69%
  - C 60-64%
  - C- 55-59%
  - F <55%
Policies

- Don’t copy someone else’s code
- Don’t give your code away
- Don’t outsource your assignments
- Don’t use electronic devices in exams
- Turn off speakers/cellphone during class
No Cheating!

- 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.
- I am serious!
About me

- Originally from Hong Kong
- B.S. degree at the University of Wisconsin, Madison
- M.S. degree at the California State University, Fresno
- PhD. at UCM two years ago
- Research interests: computer vision/image processing
About you

- When did you take CSE21?
- Computer at “home”?
  - Windows
  - Mac
  - Linux
- Programming languages?
  - Java, Python, HTML, Perl/CGI, C, C++...
- What’s your major?
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++
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 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
  - 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.

- And, most importantly
  - Good programmers get good grades!
C++ Compiler

- Compiler 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.
#include <iostream>

using namespace std;

int main()
{
    int class_num = 165;
    cout << "Hello, world!\n";
    cout << "Welcome to CSE \n";
    cout << class_num;
    cout << "!/n";
    return 0;
}

Output:
Hello, world!
Welcome to CSE 165!
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`
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”

In C++, our program begins with a main() method:

- int main()

Which must return an integer value at the end of the its execution:

- return 0;
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.

However, there are still some gotchas to look out for!

When the compiler fails, it will try to give you a meaningful error message.

However, that being said, sometimes they’re hard to understand, so be patient.
C++ vs. Java

- C++ is a compiled language, meaning that when you compile C++ code, it effectively becomes machine instructions.
- Java, on the other hand, is compiled into intermediate bytecode which is then executed by a virtual machine.
- C++ looks a lot like Java syntactically, but it is not identical!
- Java is fully object-oriented from the ground-up, meaning that Java "intuitively" knows what objects are (the Object class) and every class implicitly inherits from this Object class.
- C++, on the other hand, is really just C with "objects" grafted on as an after-thought.
Writing C++ Code

- C++ source code is written with a text editor, we don’t need a fancy IDE
- Example editors:
  - *gedit* is popular in Linux as well
  - *nano* is simple with less functionality
- The compiler on your system converts the source code to object code.
- The linker combines all the object code into an executable program.
Things to remember about C++

- C++ is not compiled to an intermediate bytecode, but rather into operating-system specific instructions.
- This makes C++ binaries not very portable as they are tied to the libraries and operating system internals they were built on and for.
- This does not make C++ as a language less portable, in fact C/C++ is quite portable unless you rely on operating-system specific code.
- Our source code is compiled into object-files which are then linked together to form an executable that can be run.
Reading assignment

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