Programming with C++ as a Second Language

CSE/ICS 45C

Patricia Lee, PhD

Administrivia

Instructor:

Patricia Lee, PhD

Office Hours: By Appointment Only

Email: leep@uci.edu

Subject: 45C + <any text containing info about your msg>

Communication:

Please include your FULL NAME somewhere in the message

Use your UCINET email.

Teaching Assistant:

Syed Shahbaaz Safir

Email: fsyedsha@uci.edu

Reference Material

Textbook (Optional):

Absolute C++, 6th Edtion by Walter Savitch and Kenrick Mock,
Pearson 2016, ISBN 978-0-13-397078-4

Readings: Correspond to material in the lectures and may help you fill in the details missed in class. Feel free to use your own book, work without a book, or find supplemental material online or elsewhere.

Course Organization

- Information can be found on the EEE website (eee.uci.edu)
- Grading
  - Assignments - 50%
  - Exams - 50% (Quizzes - 30%, Final - 20%)
- Assignments: Due one week after assigned and due before 8am on Thursday. No late assignments will be accepted so make sure you give yourself ample time for submissions.
- Exams: Based on material in class or covered on projects.
- Course Schedule/Updates: Posted online and discussed in class and/or emailed
- Lectures: Attending lectures is required. If you miss class, it is your responsibility to get material as needed from your colleagues. The TA and I will not be repeating material covered during the lectures during office hours. I do not allow recording of any lectures.

Disabilities Service Center

- Any students who feel that they may need an accommodation based on the impact of a disability should contact the Disability Services Center online or by phone at (949) 824-7494 as soon as possible to better ensure that such accommodations, such as alternative test-taking environments or note-taking services, can be arranged for you in a timely way.
- http://www.disability.uci.edu/

Academic Honesty

- As ICS 45C or CSE 45C students, you are expected to know and follow the academic honesty policies of both the Bren School of ICS and the University as a whole. Please take a few minutes to read the policies, which can be found at this link: http://www.ics.uci.edu/ugrad/policies/#03

Course Background and Goals

- Assumptions:
  - Familiarity with at least one programming language (equivalent to 1st year computer science sequence, e.g. ICS 31, 32, and 33)
  - Successfully written programs of more than a trivial size (100's of lines)
  - Have an understanding of how to break larger problems into smaller ones
  - Use of language's features correctly
- Goals:
  - C++ Programming Language
  - Understanding similarities/differences and strengths/weaknesses with respect to other languages
  - Build new techniques

Types of Software Languages

- Machine Language: CPU instructions represented in binary
- Assembly Languages: CPU instructions with mnemonics
- High-Level Languages: Commonly used languages (C, C++, Java, Python) → must be translated into machine/assembly code

High-Level Language Types

- Compiled: Translate instructions once before running code
  - C, C++, Java (partially)
  - Translation occurs only once and saves time

- Interpreted: Translate instructions while code is executed
  - Perl, Python, Unix/Linux system shell scripts, BASICA (old BASIC language), Java (partially), “virtual machines” that allow architecture-specific information to be handled and the same source code to run on any platform, Java (partially) – executes in memory giving appearance of an interpreted language

Compiled Languages

- Advantages:
  - Speed performance
  - Can distribute stand-alone executables
- Disadvantages:
  - Parsing and execution occur in two distinct steps
  - Several different stages of files: source code (text instructions), object code (parsed source), executable (linked object code)

Interpreted Languages

- Advantages:
  - Ease of programming (type instructions in text file, interpreter runs it without a linker required)
- Disadvantages:
  - Poor Speed Performance
  - No executable generated (non-distributable program since interpreter must be present on a system to run the program)

Introduction

- Bjarne Stroustrup
- Improvement on language called C
- C++ is a high-level, compiled language
  - Written, **compiled**, **assembled**, **linked**, and **loaded** before it becomes an executable that is **run**
- Systems programming language (provides access to hardware while still being high-level enough to write application software)
- Compiled to machine code to make best use of resources (with performance)
- Compiled efficiently on machines
- Compatibility with C when possible

Major Language Features

- Classes
- Templates
- Exceptions
- Inheritance with Polymorphism
- Large Standard Library of Classes

Other Features

- Primitive Data Types
- Expressions
- Statements
- Functions

Classes

- Analogous to a house blueprint
  - Can build actual objects (homes in particular locations) from this blueprint
- Defines state (attributes) and behavior (methods) for a set of similar objects
- Has 3 levels of accessibility/3 sections
  - interface (public)
    - Part accessible by owners of an object
    - Robust access to attributes and operations
  - implementation (private)
    - Internal part that makes the object work
  - inherited interface (protected)
    - Efficient or unchecked access to internals
    - Avoids giving derived classes direct access to implementation

Objects

- Instances of Classes
- Have a life-time
  - Created: allocation into memory, construction
  - Life: call member functions on them (bound to the object), call operators on them
  - Destruction: deallocation

3 Areas of Memory

- Static Data Area (retain values across function calls)
  - Static function locals
  - Static data members
  - Static “global” variables
- Stack (fast allocation and deallocation, re-used easily)
  - Function parameters
  - Function local variables
  - Anonymous expression temporaries and return values
- Free store (flexible, but programmer must be careful when deleting)
  - All objects allocated via new and new[]
  - Matching delete must be called by programmer
  - Potential for memory leaks or duplicate deletes

Templates

- Classes and functions may be templates
- Template parameters may be types and constants
- Allows definition of reusable classes and functions
- Defining methods externally can be painful (and may not work with older compilers)
- Write and test as non-template before making into a template

Exceptions

- Useful for making code more robust
- Typically for handling errors and boundary conditions
- Exceptions are thrown and must be caught
- Excepts are sent to appropriate catch by type matching
- Use only when necessary to make your unit interface robust
- Don’t silently correct an error, throw an exception instead
- Probably a good idea to have main catch any exception

Inheritance and Polymorphism

- Allows common interface to a variety of different implementations
- Makes systems pluggable and configurable
- Allows you to define frameworks which objects may be plugged into

Standard Template Library (STL)

- An extensible set of containers, iterators, and algorithms
- Uses templates which are instantiated by the compiler
- You only pay for the code you use
- Uses template specialization to optimize certain cases
- A worthwhile investment to learn
- Starts with: string, set, map, list, vector
- Avoid using C strings

Primitive Data Types

- bool
- char, wchar_t, short, int, long (and unsigned versions of each)
- float, double, long double
- pointers
- C strings
- C arrays
- Better to reserve primitives for class implementations

Expressions and Statements

- Nearly the same as C
- Plus you may overload operators for class objects

Functions

- Useful unit of code
- Keep functions small and understandable
- Give them a good name that describes their purpose
- Declare local variables close to their first use and always initialize them in the declaration
- Use reference variables for efficiency and clarity

Compilers/IDEs

- openlab.ics.uci.edu computers will be used to test your programs (g++ compiler on Linux OS with bash command processor)
  - Only platform where we will provide help
  - OpenLab Info: https://docs.google.com/document/d/1ixkx1elCOKUW-Kt7aB1EQ4Jr_dp6G_QDb6xNYckDA6k/edit
  - Reset Password: https://support.ics.uci.edu/ltb/
  - Recovering Files: https://www.ics.uci.edu/computing/services/snapshot.php
- VisualStudio (Microsoft on Windows)

Assignment 0

- Do not turn in
- Will be posted tonight

References

- Professors Ian Harris, Alex Thornton, Richert Wang, and Raymond Klefstad
- *Absolute C++*, 6th Edtion by Walter Savitch and Kenrick Mock
- *The Design and Evolution of C++* by Bjarne Stroustrup
- http://www.dsbscience.com/index.php