Lecture 1
COP 3014 Fall 2019

August 27, 2019
Instructor: Sharanya Jayaraman

Teaching Faculty, PhD Candidate in Computer Science
Research Interests: High Performance Computing, Numerical Methods, Computer Architecture
Other Interests: Movies, Food, Spongebob
Programming I - Course Information

- Teaching Assistant: Timothy Barao

- PhD Student

- Research Interests: Machine Learning, Deep Learning, Deeper Learning, Deepest Learning, Malware Analysis

- Other Interests: Movies, Food, Video Games, Scaring Elon Musk with AI Research
Programming I - Course Information

- Teaching Assistant: David Miller

- PhD student
- Research Interests: Manifold Learning, Computational Geometry, Topology
- Other Interests: Rock Climbing, PC Games, Mechanical Keyboards
Programming I - Course Information

- Teaching Assistant: Christopher Draper
- Graduate Student
- Research Interests: Random Number Generation
- Other Interests: Games, books, naps
Teaching Assistant: Xingang Fang

PhD Student, PhD in Chemistry
Research Interests: Machine Learning, Deep Learning, Big Data, Cheminformatics
Other Interests: Swimming, Cooking
Programming I - Course Information

- Teaching Assistant: Yu Miao

- Graduate Student

- Other Interests: food, cooking, swimming
Teaching Assistant (grader): Alexander No

Computer Science Senior

Research Interests: Natural Language Processing, Knowledge-based AI

Other Interests: memes, games
Programming I - Course Information

- Teaching Assistant (grader): Phat Thanh Tran

- Computer Science Senior
- Interests: Machine Learning, beatbox, work out, and Fire Emblem
- Fun facts: My fast food name is Eric
Programming I - Course Information

- Teaching Assistant (grader): Jeremiah Cummings

- Computer Science Senior
- Interests: Sailing, music
- Fun facts: Has walked all around fSU campus
Course Expectations

This is a hard class

- While this class is required for several majors as a capstone, it is also an introductory class for CS majors.

- You have signed up to learn programming in C++. At the end of the course, you should be a competent entry-level C++ programmer.

- However, this involves a lot of
Course Expectations

Effort!

▶ Programming is a skill. Almost anyone can be trained to do it well.

▶ You need to devote time outside class to practice. Practice is the only way to better yourself as a programmer.
Course Expectations

Reading

▶ Please read through the entire write-up a couple of times to understand the requirements before asking questions.

▶ Most of the assignments/problem statements will be long. Jumping the gun without reading the whole thing could be detrimental.
Course Expectations

Basic Arithmetic

- You will not be allowed calculators for the test. However, you will be expected to do some very basic math operations on your tests.

- You are being forewarned. Math is not scary.
Course Expectations

Start Early!

▶ You will be given a week to 10 days for homeworks. Please start early. You need that amount of time to complete them.
Course Expectations

Attendance

- The class is very incremental. So, skipping a few classes will get you into trouble. You are expected to attend class.

- While we understand that sometimes, circumstances result in missing a couple of classes, missing quite a few classes is not condoned.
Course Expectations

Retention

- The class is very incremental. Material introduced in one class will be applied through the rest of the course. Retaining material is important. There is no modularization of material.

- Please make sure you understand a concept before we move on to the next. We are ok with repeating material.
Ask for help!

▶ The instructor and the TA’s are available to help. Please do not hesitate to ask for help.

▶ We are willing to work with you to ensure you are learning the material. However, this requires that you start the assignments early.
Academic Honor Code

- Of late, we have been having a lot of issues with violations of the Academic Honor Code.
- Since this is your first programming class, the rules might be a bit ambiguous. We will try to clear up any confusion here.
- What is allowed:
  - Asking the instructor of TA for help
  - Discussing concepts in general, discussing the concepts used in an assignment.
  - Getting a tutor to help you with concepts and ideas.
- If you have a question about what counts as a violation to the honor code, please ask the instructor or the TA’s.
Academic Honor Code

This is a list (inexhaustive) of things that violate the Academic Honor Code

▶ Copying another person’s solution (changing variable names won’t help).
▶ Giving another person your solution.
▶ Telling another person how to solve the problem.
▶ Hiring a tutor to solve the problem for you.
▶ Asking friends/family to solve the problem for you.
▶ Copying solutions from the internet.
▶ Hiring people on the Internet (including websites like chegg) to solve the problem for you.
▶ Turning in older solutions if you’re repeating the class.
▶ Working with another student (collaboration).
Student Expectations

The most common student expectations from the survey
We have technology!
Main Components of a computer

- **CPU - Central Processing Unit**: The “brain” of the computer.
  - ISA - Instruction Set Architecture: the specific set of low-level instructions available to a CPU. Differs for various CPU types (Intel Pentium, Mac G4, etc).

- **ALU - Arithmetic & Logic Unit**: responsible for performing arithmetic calculations, as well as logical operations (comparisons for equality, inequality, for instance).

- **Main Memory (RAM - Random Access Memory)**:
  - storage close to CPU
  - Faster to access than hard disk
  - stores executing programs and data being currently worked on

- **Secondary Memory**
  - SSD, hard disk, DVD, etc.
Main Components of a computer

- **Input devices**
  - mouse, keyboard, scanner, network card, etc.

- **Output devices**
  - screen/console, printer, network card, etc.

- **Operating System**
  - Examples: Mac OS, Windows 10, Linux
  - Controls computer operations
  - Manages allocation of resources for currently running applications
Memory Concepts

- **bit**: a binary digit
  - Stores the value 0 or 1
  - Smallest unit of storage in a computer

- **byte**: 8 bits
  - Smallest addressable unit of storage in a computer
  - Storage units (variables) in a program are 1 or more bytes
  - Each byte in memory has an address (a number that identifies the location)
Programming, and Programming Languages

Program - a set of instructions for a computer to execute

Evolution of Programming languages

- **Machine Language**
  - Based on machine’s core instruction set
  - Needed by computer, hard for humans to read (1’s and 0’s)
  - Example: 1101101010110001101010

CSE 142 ROCKS HARDCORE. IT’S A GOOD THING WE DON’T USE PUNCH CARDS.
Programming, and Programming Languages

- Assembly Language
  - translation of machine instructions to symbols, slightly easier for humans to read
  - Example: ADD $R1, $R2, $R3
Programming, and Programming Languages

- High-level procedural languages
  - Abstraction of concepts into more human-readable terms
  - Closer to "natural language" (i.e. what we speak)
  - Easy to write and design, but must be translated for computer
  - Examples include C, Pascal, Fortran

- Object-oriented languages
  - Abstraction taken farther than procedural languages
  - Objects model real-world objects, not only storing data (attributes), but having inherent behaviors (operations, functions)
  - Easier to design and write good, portable, maintainable code
  - Examples include Smalltalk, C++, Java
Code Translation

Bridging the gap between high-level code and machine code

- Interpreted languages – source code is directly run on an interpreter, a program that runs the code statements
- Compiled Languages
  - A compiler program translates source code (what the programmer writes) to machine language (object code)
  - A linker program puts various object code files together into an executable program (or other target type, like a DLL)
- C and C++ are compiled languages
Software Development

Involves more than just writing code

1. Requirements Analysis
2. Design
3. Development
4. Testing
5. Maintenance

Software Development Life Cycle
Software Development

- Analysis and problem definition
- Design - includes design of program or system structure, algorithms, user-interfaces, and more
- Implementation (coding)
- Testing - can be done during design, during implementation, and after implementation
- Maintenance - usually the major cost of a software system. Not part of "development", but definitely part of the software life cycle
Programming is about Problem Solving

I can't see my forehead.
Programming is about Problem Solving

- Algorithm - a finite sequence of steps to perform a specific task
  - To solve a problem, you have to come up with the necessary step-by-step process before you can code it
  - This is often the trickiest part of programming
- Some useful tools and techniques for formulating an algorithm
  - Top-down Refinement: Decomposing a task into smaller and simpler steps, then breaking down each step into smaller steps, etc
  - Pseudocode: Writing algorithms informally in a mixture of natural language and general types of code statements
  - Flowcharting: If you can visualize it, it’s often easier to follow and understand!
Programming is about Problem Solving

- Testing - algorithms must also be tested!
  - Does it do what is required?
  - Does it handle all possible situations?

- Syntax vs. Semantics
  - Syntax – the grammar of a language.
    A syntax error: "I is a programmer."
  - Semantics – the meaning of language constructs
    Correct syntax, but a semantic error: "The headphones ate the tree."
Basic Creation and Execution of a C++ program

- Create **source code** with a text editor, store to disk.
  - Source code is just a plain text file, usually given a filename extension to identify the programming language (like .c for C, or .cpp for C++)

- **Preprocessor** – Part of compiler process, performs any pre-processing tasks on source code.

- **Compilation** – syntax checking, creation of **object code**.
  - Object code is the machine code translation of the source code.

- **Linking** – Final stage of the creation of an executable program. Linking of object code files together with any necessary libraries (also already compiled).

- **Execution of program**
  - Program loaded into memory, usually RAM
  - CPU executes code instructions
The recommended software is JetBrains CLion

- You can find it at https://www.jetbrains.com/clion.
- Get a student license for JetBrains products from https://www.jetbrains.com/community/education/?fromMenu
- You will get an email from JetBrains. Follow along with the instructions and you will be asked to create a JetBrains account. Upon doing that, you will get a key.
- You can then Download and Install CLion. Please follow the HowToCLion document on the Software page of the course website for instructions on installing and using the software.

- You can also use XCode, Visual Studio, etc. However, if you do so, please keep in mind that the TA’s will use CLion to grade.
- You can also create an account on the CS department programming servers, and use a text editor and the g++ compiler to run your code.