

CSC 211: Object Oriented Programming

Introduction

Michael Conti

Department of Computer Science and Statistics
University of Rhode Island

Fall 2020



Original design and development by Dr. Marco Alvarez

CSC 211?

- Introduction to Programming
 - ✓ focus on **problem solving**
- Introduction to Object Oriented Programming
 - ✓ classes, objects, inheritance
- Review of elementary CS techniques, algorithms and data structures
 - ✓ e.g. recursion, sorting, linked lists

Language of choice: **C/C++**.
Prior programming experience is not strictly necessary.

Prerequisites: **CSC 106** or major in Computer Engineering

3

Team

- Instructors

- ✓ Michael Conti

- Undergraduate Courses

- ✓ CSC 101, **CSC 211**, CSF 432, CSF 534

- Graduate Courses

- ✓ CSF 534

2

Tentative Schedule

Tentative Course Outline:

The weekly schedule is subject to change per instructor discretion.

Week	Topics	Resources
Week 1	Lecture - Introduction to 211, Computer Systems, Programming Languages Lab - Hello 211, IDE Setup, Basic Shell Commands Reading - Savitch, Chapter 1	Lecture Slides
Week 2	Lecture - Problems/Algorithms/Programs, History of C++, The Compiler Lecture - C++ Basics, Input/Output, Data Types, Expressions Lab - Algorithms, Problem Design, Pseudo-code Exercises Reading - Savitch, Chapter 2	Lecture Slides Lecture Slides Lab
Week 3	Lecture - Number Systems, Further look into DataTypes Lecture - Expressions, Selection Statements Assignment - Assignment 0 Lab - Programming Exercises (branching) Reading- Savitch, Chapter 3	Lecture Slides Lecture Slides Lab
Week 4	Lecture - Introduction to Loops (for) Lecture - Loops (while, do while) and Nested Loops (examples) Assignment - Assignment 1 Lab - Programming Exercises (loops and nested loops) Reading- Savitch, Chapter 3	Lecture Slides Lecture Slides Lab

4

Tentative Schedule

Week 5	Lecture - Functions Lecture - Scope of Variables, Parameter passing, Call Stack Lab - Using the Debugger, Programming Exercises (functions) Reading - Savitch, Chapter 4 Reading - Savitch, Chapter 5	Lecture Slides Lecture Slides Lab
Week 6	Lecture - Arrays, Arrays and Functions Exam - Midterm Exam (weeks 1 to 5) Lab - Strings (C style strings and string objects) Assignment - Assignment 2 Reading - Savitch, Chapter 7 Reading - Savitch, Chapter 8	Lecture Slides Lecture Slides Lab
Week 7	Spring Break	None
Week 8	Lecture - Basic Sorting Lab - Basic sorting algorithms	Lecture Slides Lecture Slides Lab
Week 9	Lecture - Multidimensional Arrays Lecture - Pointers Lab - Programming Exercises (pointers) Reading - Savitch, Chapter 9	Lecture Slides Lecture Slides Lab
Week 10	Assignment - Assignment 3 Lecture - Recursion and Examples Lecture - Recursion (cont.) and Examples Lab - Programming Exercises (tracing recursion, drawing recursion trees)	Lecture Slides Lecture Slides Lab

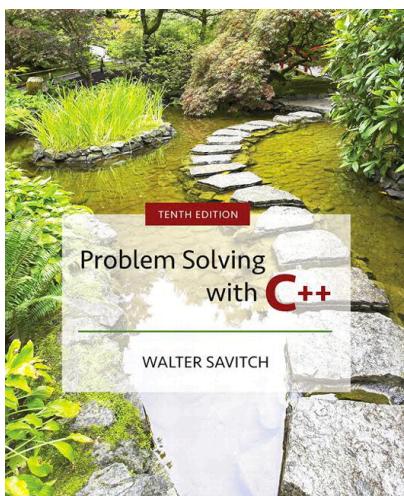
5

Tentative Schedule

Week 11	Lecture - Binary Search Lecture - Advanced Recursion (Backtracking), Structs Lab - Advanced Recursive Problems	Lecture Slides Lecture Slides Lab
Week 12	Assignment - Assignment 4 Lecture - Classes, Data Members and Methods (Encapsulation) Exam - Midterm Exam (weeks 6 to 10) Lab - Implementing Classes (source/headers), Arrays and Objects	Lecture Slides Lecture Slides Lab
Week 13	Lecture - Constructors Lecture - Dynamic Memory Allocation, Destructors Lab - Developing a string Class (overloaded operators and copy constructors) Reading - Savitch, Chapter 14	Lecture Slides Lecture Slides Lab
Week 14	Lecture - Class Inheritance Lecture - Singly Linked Lists Lab - STL Containers, read/write from files, and CLAs Reading - Savitch, Chapter 15	Lecture Slides Lecture Slides Lab
Week 15	Exam - Final Exam (cumulative with focus on weeks 11 to 15)	None

6

Required textbook



No need to buy
**MyLab
Programming**



C/C++?

Recommended Tools

- although you are free to use **any IDE on any platform**, we will grade all assignments using **g++ on a Linux machine**
- CS50 IDE is strongly **recommended**!
- alternatives:
 - vim, g++, gdb (running on Linux)
 - VSCode

7

8

CS50 IDE

The screenshot shows the CS50 IDE interface. In the top panel, there's a file tree with 'hello.c' selected. Below it, the code editor contains the following C code:

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     printf("hello\n");
6 }
```

At the bottom of the screen, a terminal window shows the command `~/ $ make hello`.

9

Grading (subject to change)

Assignments

- ✓ ~5 programming assignments (30%)
- ✓ lab attendance (5%)

Exams

- ✓ 2 exams (35%)
- ✓ 1 final exam (30%)



All exams are based on textbook chapters and lecture materials

10

Programming Assignments

• Discussions and collaboration are allowed, however you **must write your own code**

• All programming assignments will be **automatically** graded on [Gradescope](#)

✓ no less than 7 days to complete

✓ late submissions are **NOT** accepted

Plagiarism?

✓ just **don't do it**

✓ if you get caught (chances are very high), your name(s) will be immediately reported for further sanctions

Plagiarism

File 1	File 2	Lines Matched
spring-19/submit 15826983/functions.cpp (89%)	spring-19/submit 15857164/functions.cpp (94%)	167
spring-19/submit 15858590/functions.cpp (47%)	spring-19/submit 15860745/functions.cpp (88%)	161
spring-19/submit 15845050/functions.cpp (52%)	spring-19/submit 15859763/functions.cpp (64%)	151
spring-19/submit 15845050/functions.cpp (50%)	spring-19/submit 15854554/functions.cpp (42%)	122
spring-18/submit 6696725/functions.cc (48%)	spring-18/submit 6706969/functions.cc (66%)	91
fall-18/submit 9926606/functions.cpp (72%)	spring-19/submit 15843429/functions.cpp (61%)	128
spring-18/submit 6697832/functions.cc (56%)	spring-18/submit 6706969/functions.cc (60%)	117
spring-18/submit 6696725/functions.cc (44%)	spring-18/submit 6701298/functions.cc (44%)	85
spring-19/submit 15849001/functions.cpp (66%)	spring-19/submit 15856189/functions.cpp (73%)	189
fall-18/submit 9867275/functions.cpp (89%)	spring-19/submit 15860062/functions.cpp (65%)	179
spring-19/submit 15861942/functions.cpp (87%)	spring-19/submit 15863011/functions.cpp (85%)	132
spring-19/submit 15856189/functions.cpp (68%)	spring-19/submit 15857164/functions.cpp (67%)	122
fall-18/submit 9933156/functions.cpp (73%)	spring-19/submit 15857316/functions.cpp (49%)	128
spring-19/submit 15826983/functions.cpp (58%)	spring-19/submit 15856189/functions.cpp (61%)	105
spring-18/submit 6712472/functions.cc (67%)	spring-18/submit 6712960/functions.cc (64%)	149
spring-18/submit 6701298/functions.cc (35%)	spring-18/submit 6706969/functions.cc (48%)	58
spring-18/submit 6696725/functions.cc (35%)	spring-18/submit 6697832/functions.cc (44%)	92
spring-19/submit 15861491/functions.cpp (65%)	spring-19/submit 15861942/functions.cpp (74%)	150
spring-19/submit 15856342/functions.cpp (42%)	spring-19/submit 15858250/functions.cpp (45%)	112
spring-19/submit 15862600/functions.cpp (66%)	spring-19/submit 15863011/functions.cpp (71%)	96
spring-19/submit 15861491/functions.cpp (64%)	spring-19/submit 15863011/functions.cpp (71%)	138
spring-19/submit 15861809/functions.cpp (59%)	spring-19/submit 15862609/functions.cpp (46%)	116
fall-18/submit 9936095/functions.cpp (61%)	spring-18/submit 6700982/functions.cc (42%)	67
spring-19/submit 15861942/functions.cpp (68%)	spring-19/submit 15862600/functions.cpp (62%)	86
spring-19/submit 15849001/functions.cpp (49%)	spring-19/submit 15857164/functions.cpp (54%)	155
spring-19/submit 15857164/functions.cpp (53%)	spring-19/submit 15862216/functions.cpp (53%)	125
spring-19/submit 15862555/functions.cpp (57%)	spring-19/submit 15862609/functions.cpp (44%)	107
fall-18/submit 9856744/functions.cpp (61%)	spring-19/submit 15827542/functions.cpp (47%)	149

11

12

How to succeed in this class?

- › I do not spend time taking attendance ... but ...
 - ✓ students skipping lectures will (very) likely **fail** this class
- › **Organize** your time
 - ✓ lectures, labs, discussion sections, programming assignments, exams
- › **Participate** and think critically
 - ✓ ask questions (lectures, labs, discussion sections, office hours, Piazza)
- › Start assignments **early**
 - ✓ programming and debugging takes time (especially for all/nothing grading)
 - ✓ **avoid** copying/pasting or google'ing answers by all means

13

How to succeed in this class?

- › Skim related textbook section / chapter before coming to lecture
 - ✓ read thoroughly afterwards and solve problems/exercises
- › Do not expect assignment solutions from the TAs
- › Think before you code
 - ✓ write pseudocode on paper
- › Write code incrementally
 - ✓ write, compile, test frequently

14

Need help?

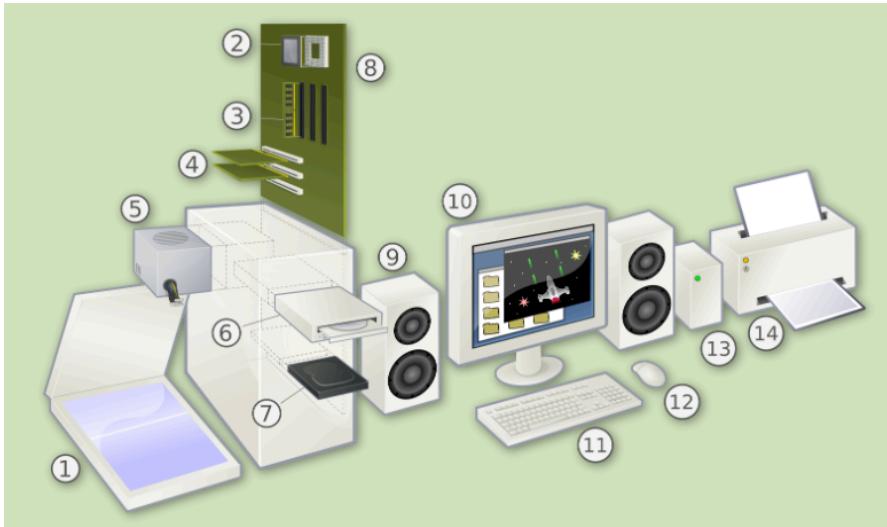
- › Come to **Office Hours**
- › Post questions on **Piazza**
 - ✓ answer questions, share information

piazza

15

Computer Systems

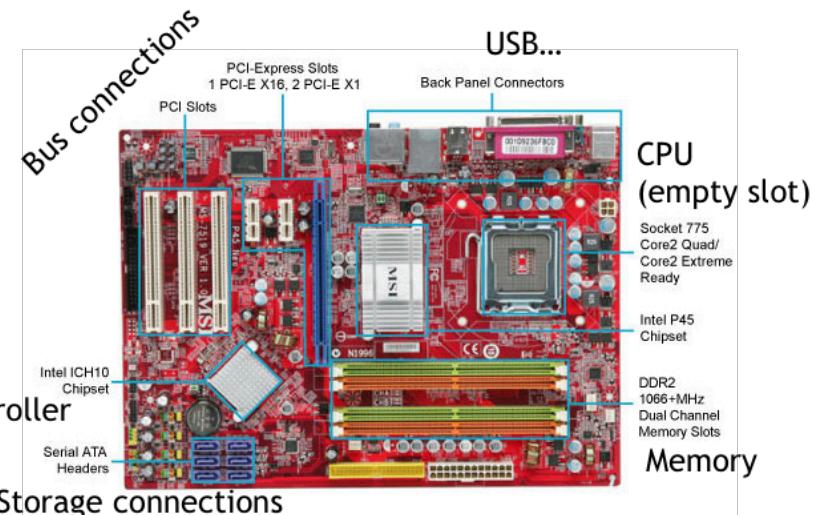
Computer Components



<https://bjc.edc.org/bjc-r/cur/programming/6-computers/1-abstraction/07-digital-components.html>

17

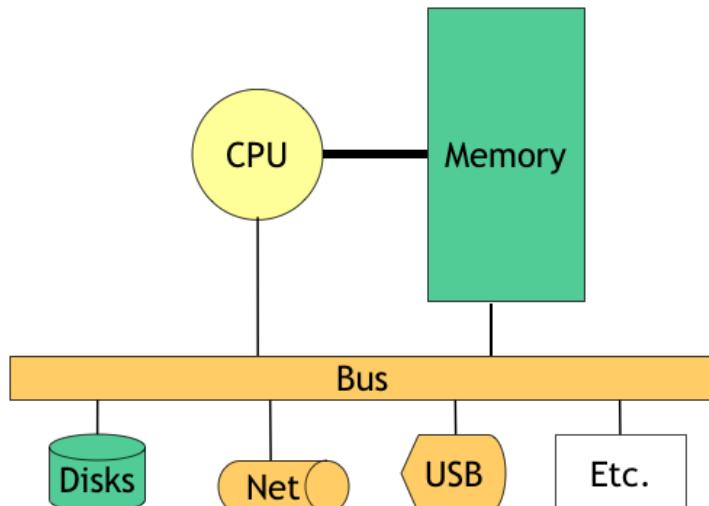
Inside a typical computer/laptop



from: CSE 351, University of Washington

18

Von Neumann model

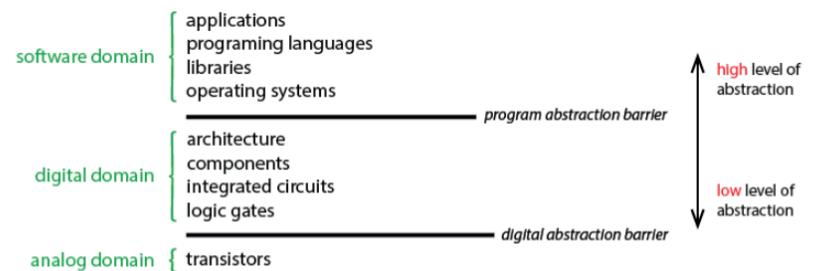


from: CSE 351, University of Washington

19

Abstraction Layers

"the process of removing physical, spatial, or temporal details or attributes in the study of objects or systems in order to focus attention on details of higher importance" [wikipedia]



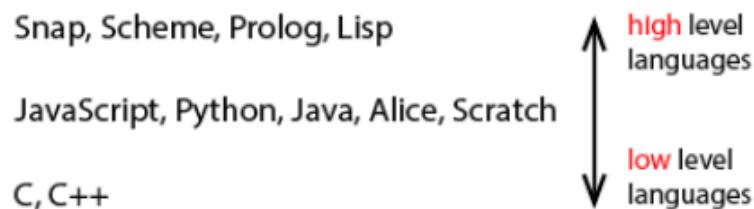
Different people might draw this diagram slightly differently, so don't try to memorize all the levels. The key abstraction levels to remember are software, digital computer hardware, and underlying analog circuit components.

<https://bjc.edc.org/bjc-r/cur/programming/6-computers/1-abstraction/01-abstraction.html>

20

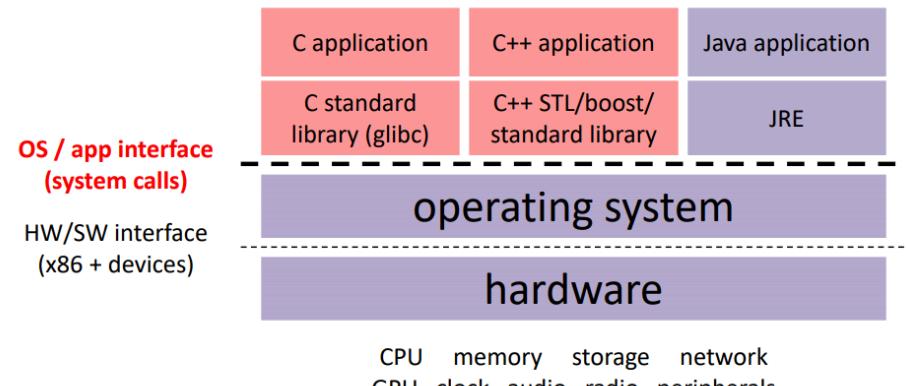
High-Level and Low-Level Languages

A *high-level language* (like Snap! or Scheme) includes many built-in abstractions that make it easier to focus on the problem you want to solve rather than on how computer hardware works. A *low-level language* (like C) has fewer abstractions, requiring you to know a lot about your computer's architecture to write a program.



<https://bjc.edc.org/bjc-r/cur/programming/6-computers/1-abstraction/03-software-languages.html>

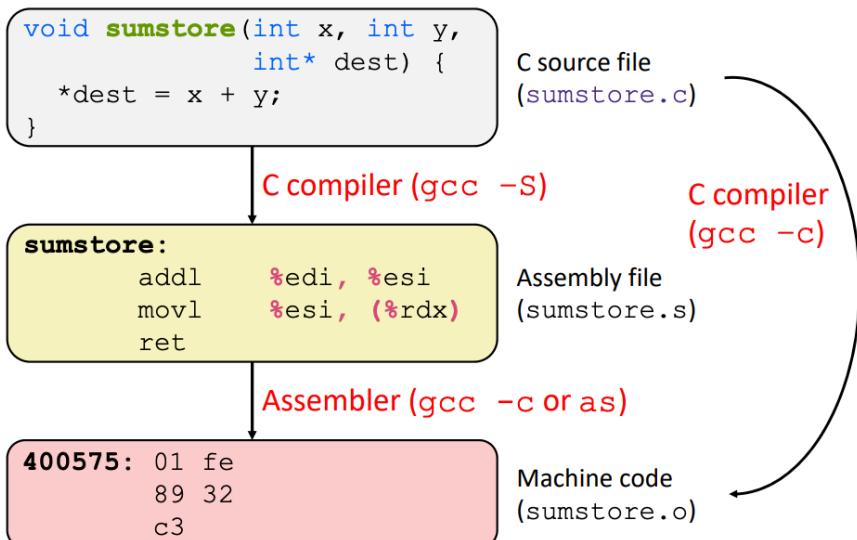
Programming applications



from: CSE 333, University of Washington

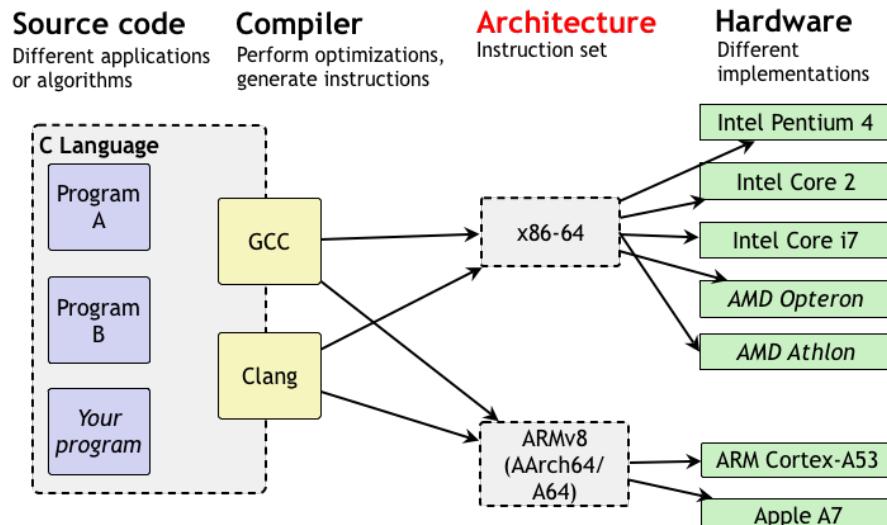
22

Compiling C code



from: CSE 333, University of Washington

Multiple targets



from: CSE 351, University of Washington

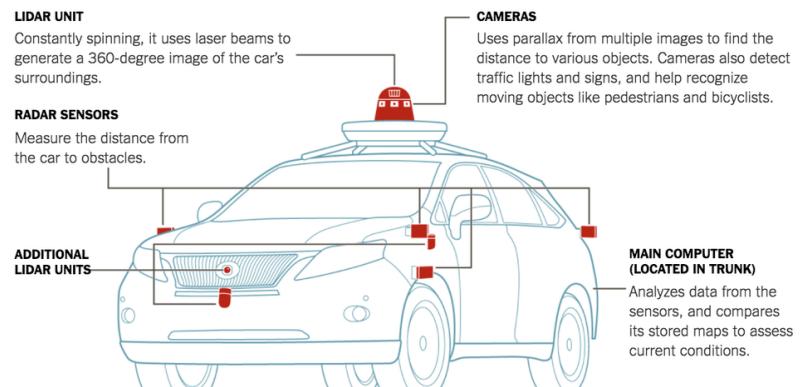
24

Devices everywhere



25

Devices everywhere



<https://www.nytimes.com/2018/03/19/technology/how-driverless-cars-work.html>

26