AI INSTITUTE #AIISC
UNIVERSITY OF SOUTH CAROLINA

UNIVERSITY OF SOUTH CAROLINA

CSCE 240: Advanced Programming Techniques
Lecture 29: Wrap-up and Conclude

PROF. BIPLAV SRIVASTAVA, AI INSTITUTE

2ND MAY 2023

Carolinian Creed: "I will practice personal and academic integrity."

Credits: Some material reused with permission of Dr. Jeremy Lewis. Others used as cited with thanks.

Organization of Lectures 29

- Introductory section
- Main section
  - Common project presentation
  - Course Recap
    - Course goals
    - Highlights
      - Lectures
      - Homework assignments and peer-evaluation
      - Programming assignment
  - Future: Programming, Research, AI
- Concluding section
  - Ask me anything

Introduction Section

Last Class

- PA #5 and Quiz-2 marks posted
- Grades to be posted by Tuesday, May 4
  - PA #5 remaining

Main Section

Common Course Project

- **Github**: https://github.com/JamelChouarfia/CSCE240FinalProject

Presentation: 20 mins

Course Wrap-Up

Learning Objectives

- Develop language-independent understanding of programming concepts by being exposed to multiple languages (C++, Java, Python)
- Independently design and implement programs in multiple language of choices (C++, Java or Python based on choice) in a Unix environment
- Demonstrate mastery of pointers, iterators, memory management including object creation and destruction, and parameter passing in C++
- Demonstrate mastery of object-oriented programming concepts including: inheritance, polymorphism, operator overloading, template functions and classes, and the use of STL containers.
- Develop object-oriented models using UML
- Able to work in programming teams with code review and walk throughs
- Solve practical problems that matter

Lectures: Topics Covered and In-Scope

Class #	Date	Description	Comments
1	Jan 10 (Tu)	Introduction	
2	Jan 12 (Th)	Introduction – Pointers, Iteration	
3	Jan 17 (Tu)	Input/ Output	
4	Jan 19 (Th)	I/O, Exceptions	HW 1 due
5	Jan 24 (Tu)	Memory management, User defined types	Prog 1 - start
6	Jan 26 (Th)	Object Oriented (OO) intro	HW 2 due
7	Jan 31 (Tu)	OO concepts, UML Notations	
8	Feb 2 (Th)	Code org (C++)	Prog 1 - end
9	Feb 7 (Tu)	OO – inheritance	Prog 2 - start
10	Feb 9 (Th)	Regex, OO - polymorphism	HW 3 due
11	Feb 14 (Tu)	In class test	Quiz 1 – In class
12	Feb 16 (Th)	Review: inheritance, Polymorphism	
13	Feb 21 (Tu)	Exceptions	Prog 2 - end
14	Feb 23 (Th)	OO – Constructor, Destructor	Prog 3 - start
15	Feb 28 (Tu)	OO – operators, access control	HW 4 due
16	Mar 2 (Th)	C++ standard library	Prog 3 - end Semester - Midpoint
17	Mar 14 (Tu)	Testing strategies	Prog 4 - start
18	Mar 16 (Th)	Advanced: Pointers	HW 5 due
19	Mar 21 (Tu)	Advanced: Pointers, I/O	
20	Mar 23 (Th)	Advanced: Operator overloading	Prog 4 – end (March 26, 2023)
21	Mar 28 (Tu)	Advanced: Memory Management	Prog 5 – start
22	Mar 30 (Th)	Advanced: Code efficiency	
23	Apr 4 (Tu)	Advanced: Templates	
24	Apr 6 (Th)	AI / ML and Programming	Prog 5 – end
25	Apr 11 (Tu)	Project code summary – student presentation for reuse Review material for Quiz 2	HW 6 due Prog 6 – assembling start
26	Apr 13 (Th)	In class test	Quiz 2 – In class
27	Apr 18 (Tu)	Project presentation	Prog 6 - due
28	Apr 20 (Th)	Project presentation	Last day of class
	Apr 25 (Tu)		Reading Day
29	May 2 (Tu)	9am – Exam or Final Overview	Examination

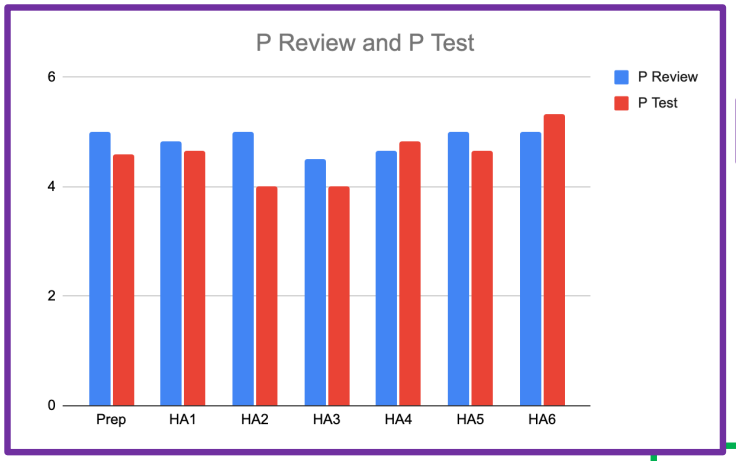
Lecture Logistics

- Material on github
- Code in C++
  - Java (and Python too whenever feasible)
- Homeworks (6) in C++; peer evaluated
- Prog. Assignments (6) in C++/Java [/Python an option]
- Quizzes (2) – in-class and pseudo-code
- Other practices encouraged
  - McNair Fellowship

Homework Assignments and Peer-Evaluation

- HA1 to HW5, all in C++ (HA6 optional)
- Maximum improvement achieved
  - Peer review (4.83 -> 5; 3.5% ↑)
    Peer test (4.6 -> 5.33; 15.8% ↑)
  - Small class (6), Honors majority
- Caveats
  - The students evaluated themselves although we gave the rubric on how to score.
  - The HWs were on different topics.
  - The number of students participating in each quiz were not constant.

Still, the results are encouraging!

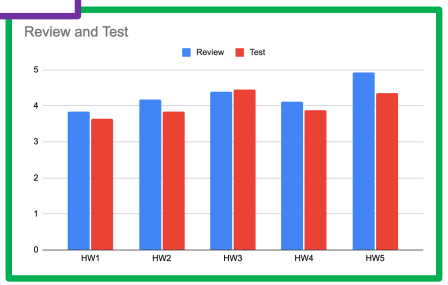


2023 Spring

2022 Spring

At least 1 point improvement
with min participation: 32
(max class strength: 50+)

- Peer review (3.83 -> 4.92; 28.3% ↑)
- Peer test (3.64 -> 4.35; 19.4% ↑)



Many Interesting Insights

- Initiative to learn – from project (2023)
  - Spelling correction library
  - Optimization of Q-A pairs
  - Dynamic fetching of content, HTML
  - Swing-based UI
- Choice of languages

C++	1
Java	5

- Initiative to learn – from project (2022)
  - Synonyms of terms, to detect intents better
  - Comparison at level of letters, to handle noisy text
  - Handling additional languages – Spanish
  - New UML diagramming tool – Mermaid - https://mermaid-js.github.io/mermaid/#/
  - Grouping concepts to answer higher concepts (knowledge graph)
  - Trying multiple programming languages for different project assignments
- Choice of languages

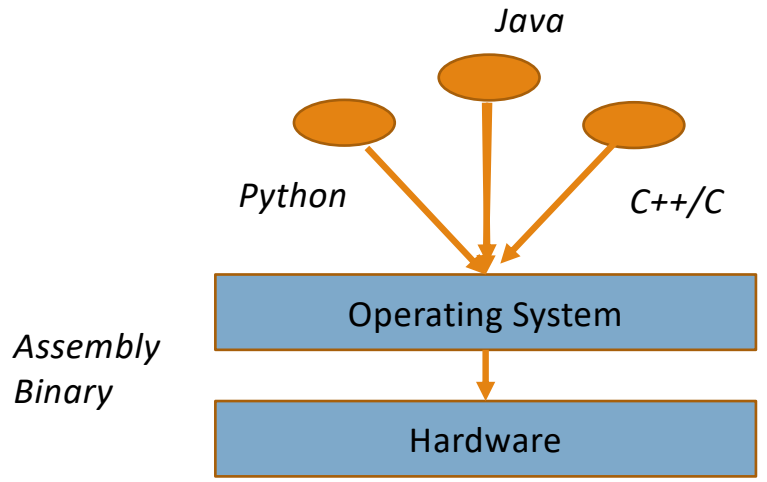
C++	12
Java	26
Python	6

Future: Programming, Research, AI

Programming Techniques

- Languages inevitably change over time
- Code practices remain
  - Adopt a language as mother tongue
    - Understand concepts in-depth
    - Experiment and settle on a coding style
  - Programming: variable initialization, understanding types, … usage of libraries
  - Memory: using just the right amount
  - Algorithms: focusing on efficiency
  - Documentation
  - Debugging methods
  - Testing, …

Programming languages are really for ***communicating among developers*** for building systems on OS/Hardware collaboratively



Research: Complexity of Code

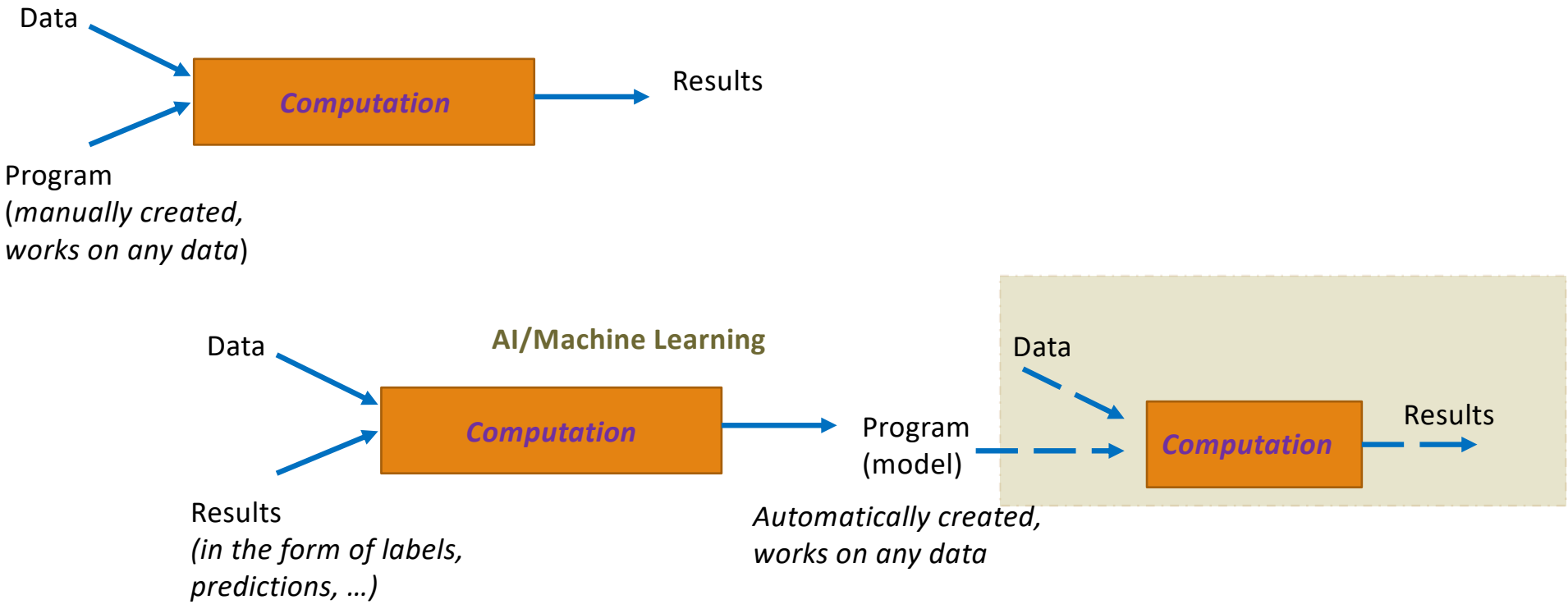
Source: From code complexity metrics to program comprehension, Dror Feitelson, CACM May 2023

- "***Developers spend 58-70% time understanding code, 5% editing it***"
- Hundreds of metrics to measure complexity – independent of language
  - Lines of code
  - Branching factor
  - …
- Shift towards understandability of code
  - Background and assumptions (about computers, problems, language constructs, performance, …) impact understanding of code

Research: Where is AI in IT?

- People have traditionally changed themselves to use IT
  - Examples: Typing, fixed menus
  - Focus on repeatability, user control
  - Disadvantage: usage barrier, entry barrier
- With AI: IT changing to enable people to use them naturally
  - Example: Natural language based interaction … chatbot
  - Focus on dynamicity, data-driven behavior
  - Disadvantage: hard to debug, audit and establish accountability

Traditional Programming v/s Machine Learning



Programming Trends

- Expect more languages to improve developer productivity
  - But good developers understand the underlying operating environment and have sound programming technique
- Expect more automatic code-generation
  - Example: OpenAI's Co-pilot: https://copilot.github.com/
  - Example: ChatGPT
- Automatic software generation is a long-established area
- AI in programming, and programming for AI will grow

Trends

- Stack Overflow: bans answers by ChatGPT –
  - Details: https://meta.stackoverflow.com/questions/421831/temporary-policy-chatgpt-is-banned
  - Overflow of wrong answers

Ask Me Anything