COSE212: Programming Languages

Lecture 0 — Course Overview

Hakjoo Oh
2017 Fall

Basic Information

Instructor: Hakjoo Oh

- **Position:** Assistant professor in CS, Korea University
- **Expertise:** Programming Languages
- **Office:** 616c, Science Library
- **Email:** `hakjoo_oh@korea.ac.kr`
- **Office Hours:** 1:00pm–3:00pm Mondays and Wednesdays (by appointment)

TAs:

- Junho Lee (`wnsgh1906@korea.ac.kr`)
- Dowon Song (`sdw0316@gmail.com`)

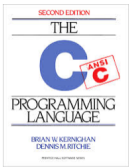
Course Website:

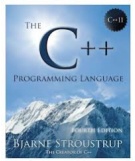
- `http://prl.korea.ac.kr/~pronto/home/courses/cose212/2017/`
- Course materials will be available here.

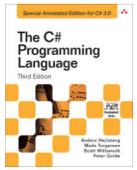
About This Course

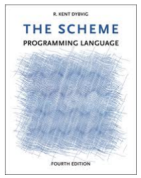
This course is *not* about

- to learn particular programming languages







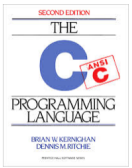


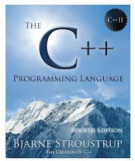

- to improve your "programming skills" (e.g., tools, libraries, etc)

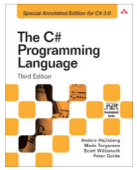
About This Course

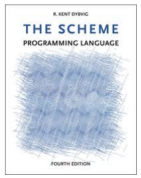
This course is *not* about

- to learn particular programming languages










- to improve your "programming skills" (e.g., tools, libraries, etc)

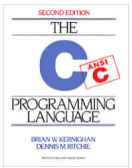
Instead, in this course you will learn

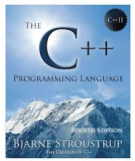
- fundamental principles of modern programming languages
- how programming systems are designed and implemented
- thinking formally and rigorously

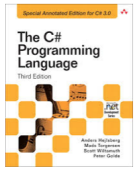
About This Course

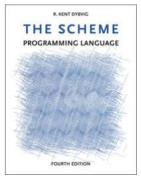
This course is *not* about

- to learn particular programming languages










- to improve your "programming skills" (e.g., tools, libraries, etc)

Instead, in this course you will learn

- fundamental principles of modern programming languages
- how programming systems are designed and implemented
- thinking formally and rigorously

To succeed in this course, you must

- have basic programming skills
- be familiar with at least two PLs (e.g., C, Java)
- have taken Theory of Computation, Discrete Math, etc
- be prepared to learn new things

Topics

- **Part 1 (Preliminaries):** inductive definition, basics of functional programming, recursive and higher-order programming
- **Part 2 (Basic concepts):** syntax, semantics, naming, binding, scoping, environment, interpreters, states, side-effects, store, reference, mutable variables, parameter passing
- **Part 3 (Advanced concepts):** type system, typing rules, type checking, soundness/completeness, automatic type inference, polymorphic type system, lambda calculus, program synthesis

Course Materials

- Essentials of Programming Languages (Third Edition) by Daniel P. Friedman and Mitchell Wand. MIT Press.

(Not required but recommended)

- Self-contained slides will be provided.

Grading

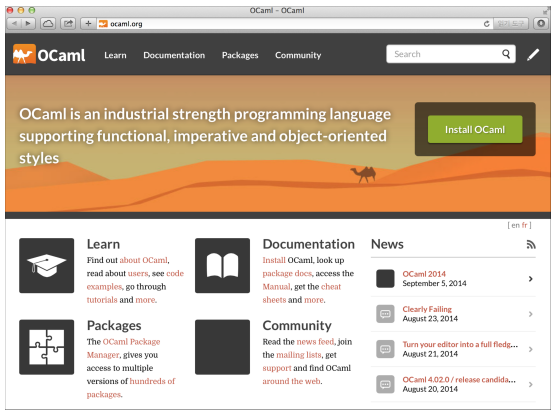
- Homework – 50%
  - 5–6 programming assignments
  - No late submissions will be accepted.
- Final exam – 45%
- Attendance – 5%

Assignment Policy / Academic Integrity

- **All assignments must be your own work.**
- Discussion with fellow students is encouraged and you can discuss how to approach the problem. However, your code must be your own.
  - Discussion must be limited to general discussion and must not involve details of how to write code.
  - You must write your code by yourself and must not look at someone else's code (including ones on the web).
  - Do not allow other students to copy your code.
  - Do not post your code on the public web.
- Cheating (violating above rules) gets you 0 for the *entire* HW score.
  - We use automatic technology for detecting clones

Programming in ML

- ML is a general-purpose programming language, reflecting the core research achievements in the field of programming languages.
  - higher-order functions
  - static typing and automatic type inference
  - parametric polymorphism
  - algebraic data types and pattern matching
  - automatic garbage collection
- ML inspired the design of modern programming languages.
  - C#, F#, Scala, Java, JavaScript, Haskell, Rust, etc
- We use OCaml, a French dialect of ML:



`http://ocaml.org`

Next Week (9/12, 9/14)

We will have a tutorial session about OCaml programming by TAs. Bring your notebook.

- Installation of the language system,
- How to write and run programs,
- How to submit assignments,
- Troubleshooting, etc.