ENE4014: Programming Languages

Lecture 0 — Course Overview

Woosuk Lee
2024 Spring

Basic Information

Instructor: Woosuk Lee

- **Position:** Assistant professor, Hanyang University
- **Expertise:** Software Analysis, Programming Languages
- **Office:** Rm 403, 3rd Engineering Building
- **Email:** `woosuk@hanyang.ac.kr`
- **Office Hours:** 10:00am–12:00pm Wednesday (by appointment)

TA

- Jinsang Kim (`rkdfursnsqlc@gmail.com`)
- Wang Ao (`waa062804@gmail.com`)

Time & Place

- Monday 09:00 - 10:15, 10:30 - 11:45 @ Rm 106, 4th Eng Bldg.
- Wednesday 15:00 - 16:15, 16:30 - 17:45 @ Rm 507, Hakyeonsan Cluster Bldg.

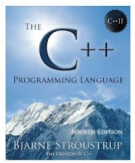
Course Website:

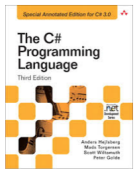
- `http://psl.hanyang.ac.kr/courses/ene4014_2024s/`
- Course materials will be available here.

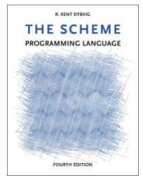
About This Course

This course is *not* about

- to learn particular programming languages








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

Instead, in this course you will learn

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

It would be helpful if you

- have basic programming skills
- are familiar with at least two PLs (e.g., C, Java)
- have taken Data Structures, Discrete Math, etc

Design and Implementation of Programming Languages

We will learn programming language concepts by designing and implementing our own programming language system.

- We will define a programming language. For example, "factorial" is written in our language as follows:

```
let x = read in
letrec fact(n) =
  if iszero n then 1
  else ((fact (n-1)) * n)
in (fact x)
```

- We will design and implement an interpreter for the language:

$$\text{Program} \longrightarrow \boxed{\text{Interpreter}} \longrightarrow \text{Result}$$

- We will design and implement a type checker for the language:

$$\text{Program} \longrightarrow \boxed{\text{Type Checker}} \longrightarrow \text{Safe/Unsafe}$$

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

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.
- Materials from related courses:
  - Programming Languages. Korea University: `http://prl.korea.ac.kr/~pronto/home/courses/cose212/2019/`
  - Programming Languages, Seoul National University: `http://ropas.snu.ac.kr/~kwang/4190.310/18/`

Grading

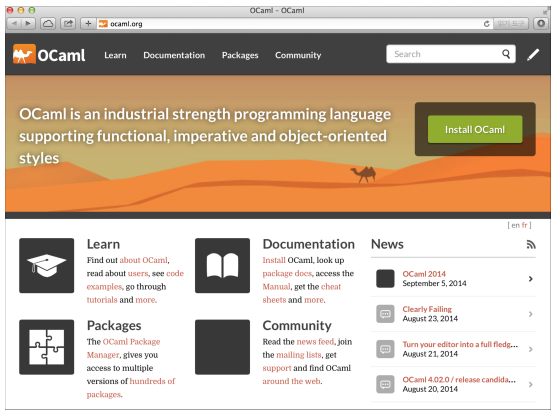
- Homework – 20%
  - 1 written assignment and 5 programming assignments
  - Late submissions will get penalty points (-20%)
- Mid exam – 35%
- Final exam – 35%
- Attendance – 10%

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`

Development Environment

- Installing OCaml: `http://www.ocaml.org/docs/install.html`
- Compiling and Executing OCaml Online
  - `https://try.ocamlpro.com` (with a simple tutorial)
  - `https://www.tutorialspoint.com/compile_ocaml_online.php`)
- IDE (Integrated Development Environment) for OCaml
  - VS Code
  - Eclipse + OcaIDE (Eclipse plugin)
  - You can find more details in the course website.