Programming Languages – Course Introduction

Jongwoo Lim

Course Objectives

- Introduction to fundamental concepts in programming languages.
  - Taxonomy and characteristics of modern programming languages.
  - How to describe a programming language:
    - What are the important language constructs?
    - How are they implemented in real systems?
  - Different paradigms of programming.
  - Other important issues such as concurrency and exception handling.
- Enhance your understanding on the programming languages currently used so that you can learn new languages quickly and easily.

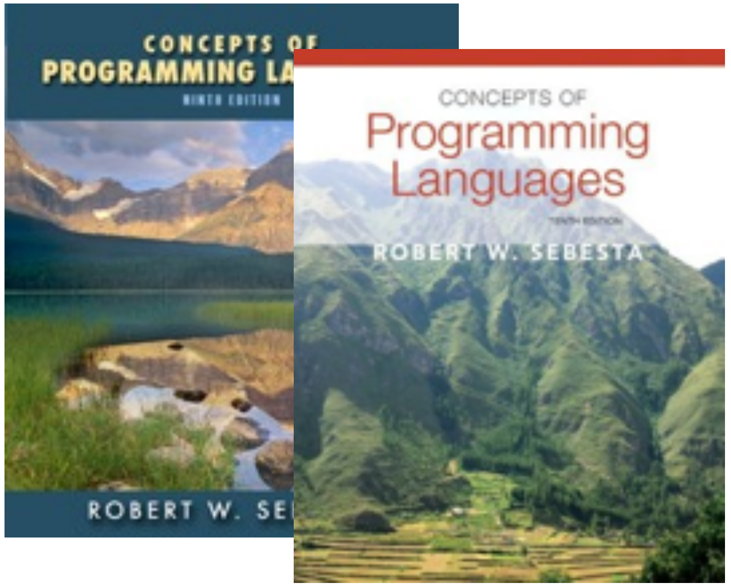
Course Objectives

- This course will help students to achieve the following objectives.
  - Understand the important features of various programming languages, and their similarities and differences.
  - Learn the formal methods of describing the syntax and semantics of programming languages.
  - Study various language constructs such as types, control structures, and subprograms.
  - Learn the characteristics and differences of imperative, object-oriented, functional, and logic programming languages.

Administration

- ENE414 Programming Languages
- Instructor: 임종우 (Jongwoo Lim, jlim@hanyang.ac.kr, IT/BT 505)
  - Office hour: Monday 14:00-15:00 @ IT/BT 505.
- Textbook:
  - Concepts of Programming Languages, Robert W. Sebesta
- Grading:
  - Mid-term: 30%
  - Final: 30%
  - Attendance/Participation: 10%
  - Homework: 30%

  → Final grade: A ≤ 30%, B ≤ 70%, ...



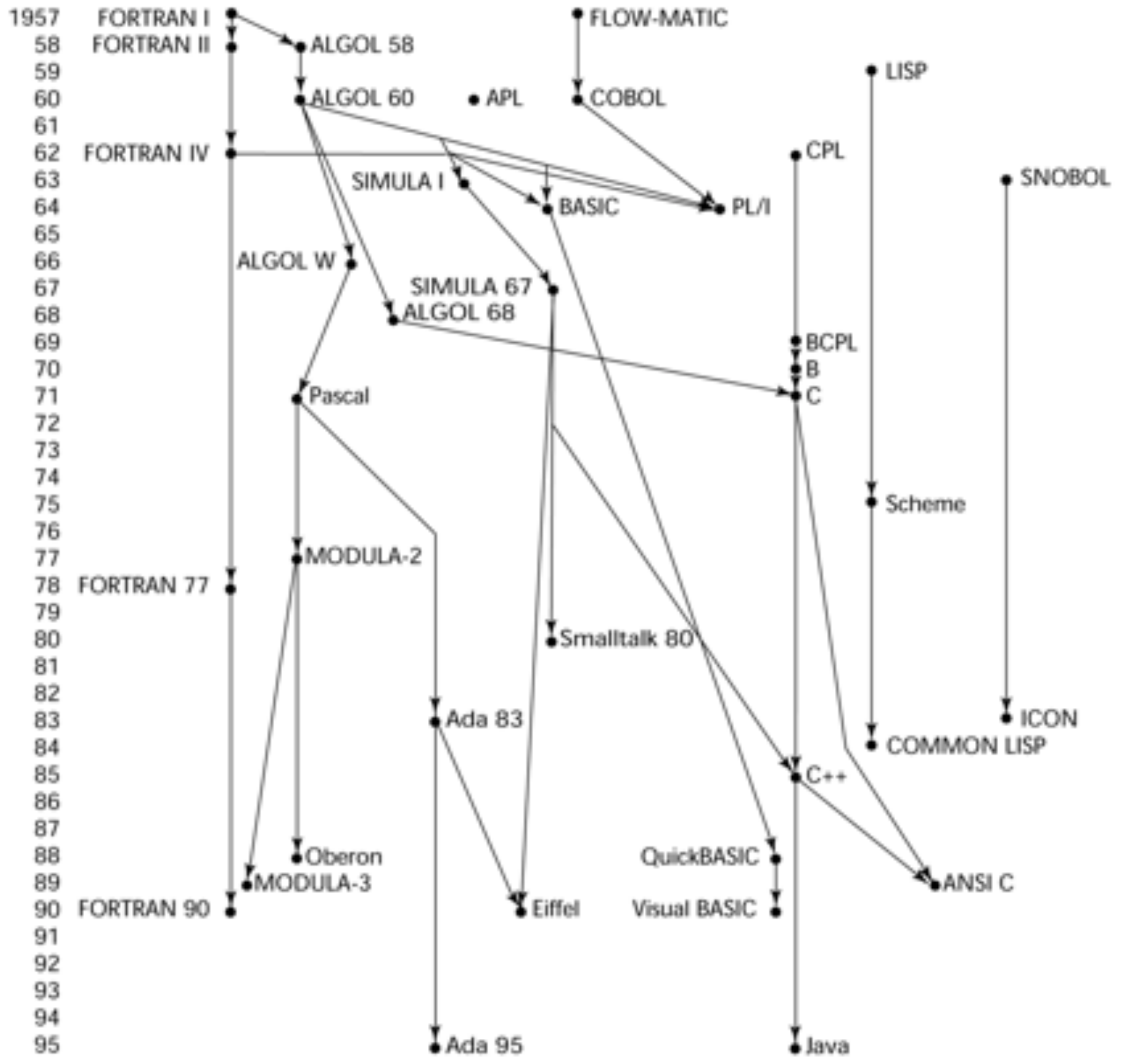
Weekly Schedule

week	chapter	description
1	1	Course introduction, Preliminaries.
2	3	Describing syntax and semantics
3	4	Lexical and syntax analysis.
4	5	Names, bindings, and scopes.
5	6	Data types.
6	7, 8	Expressions and assignment statements, Control structures.
7	9, 10	Subprograms, Implementing subprograms.
8		Mid-term exam
9	11	Abstract data types and encapsulation constructs.
10	12	Support for object-oriented programming.
11	15	Functional programming languages.
12		Functional programming languages.
13		Functional programming languages.
14	13	Concurrency.
15	14	Exception handling and event handling
16		Final exam.

Course Policy

- Homework
  - 3 programming assignments in C++, Python, and an ML.
  - You must be comfortable in writing classes in C++ / Python, and building programs on a Linux system.
  - SSH to csedev.hanyang.ac.kr (port no. 8022)
  - Late submission : -5% per every two hours.
- Attendance
  - You can be absent up to 2 times with no reason / 3 lates = 1 absence.
  - Absent in >1/3 classes = F.
- Academic integrity
  - Any violation of academic integrity = F in this class.
    - Cheating in exams, copying a homework, etc.

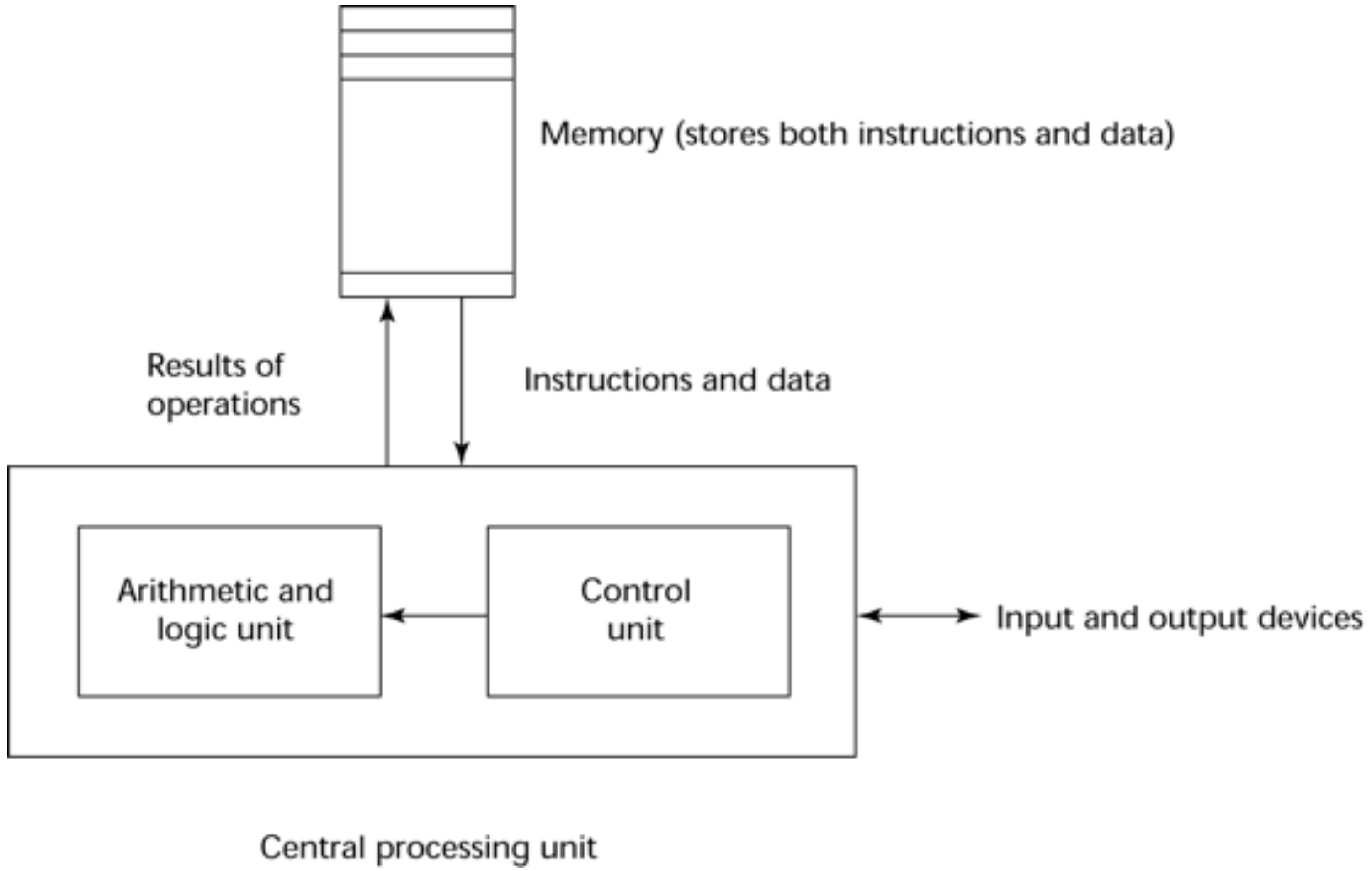
Genealogy of Common Languages



Influences on Language Design

- Computer Architecture
  - Languages are developed around the prevalent computer architecture, known as the von Neumann architecture.
- Programming Methodologies
  - New software development methodologies (e.g., object-oriented software development) led to new programming paradigms and by extension, new programming languages.

The von Neumann Architecture



The von Neumann Architecture

- Fetch-execute-cycle

  (on a von Neumann architecture computer)

```
initialize the program counter
repeat forever
  fetch the instruction pointed by the counter
  increment the counter
  decode the instruction
  execute the instruction
end repeat
```

Imperative Programming

- **Imperative programming** is a programming paradigm that describes computation in terms of statements that change a program state. [wikipedia]
  - In much the same way that imperative mood in natural languages expresses commands to take action, imperative programs define sequences of commands for the computer to perform.

Imperative Programming

```
#include <iostream>
// Fibonacci numbers, imperative style [from wikipedia]

int fibonacci(int iterations) {
  int first = 0, second = 1;  // seed values
  for (int i = 0; i < iterations; ++i) {
    int sum = first + second;
    first = second;
    second = sum;
  }
  return first;
}

int main() {
  std::cout << fibonacci(10) << “\n";
  return 0;
}
```

Functional Programming

- **Functional programming** is a programming paradigm that treats 'computation as the evaluation of mathematical functions' and avoids state and mutable data.
- It emphasizes the application of functions, in contrast to the imperative programming style, which emphasizes changes in state. [wikipedia]

Functional Programming

```
-- Fibonacci numbers, functional style (Haskell) from
-- wikipedia.
-- Describe an infinite list based on the recurrence
-- relation for Fibonacci numbers.
fibRecurrence first second =
    first : fibRecurrence second (first + second)

-- Describe fibonacci list as fibRecurrence with
-- initial values 0 and 1.
fibonacci = fibRecurrence 0 1

-- Describe action to print the 10th element of
-- the fibonacci list.
main = print (fibonacci !! 10)
```

Object-oriented Programming

- **Object-oriented programming (OOP)** is a programming paradigm using "objects“ to design applications and computer programs. [wikipedia]
  - Objects = data structures consisting of data fields and methods together with their interactions.
  - Features:
    - data abstraction, encapsulation, messaging, modularity, polymorphism, and inheritance.

Language Evaluation Criteria

- **Readability**:
  the ease with which programs can be read and understood.
- **Writability**:
  the ease with which a language can be used to create programs.
- **Reliability**: conformance to specifications.
- **Cost**: the ultimate total cost.

Language Evaluation Criteria

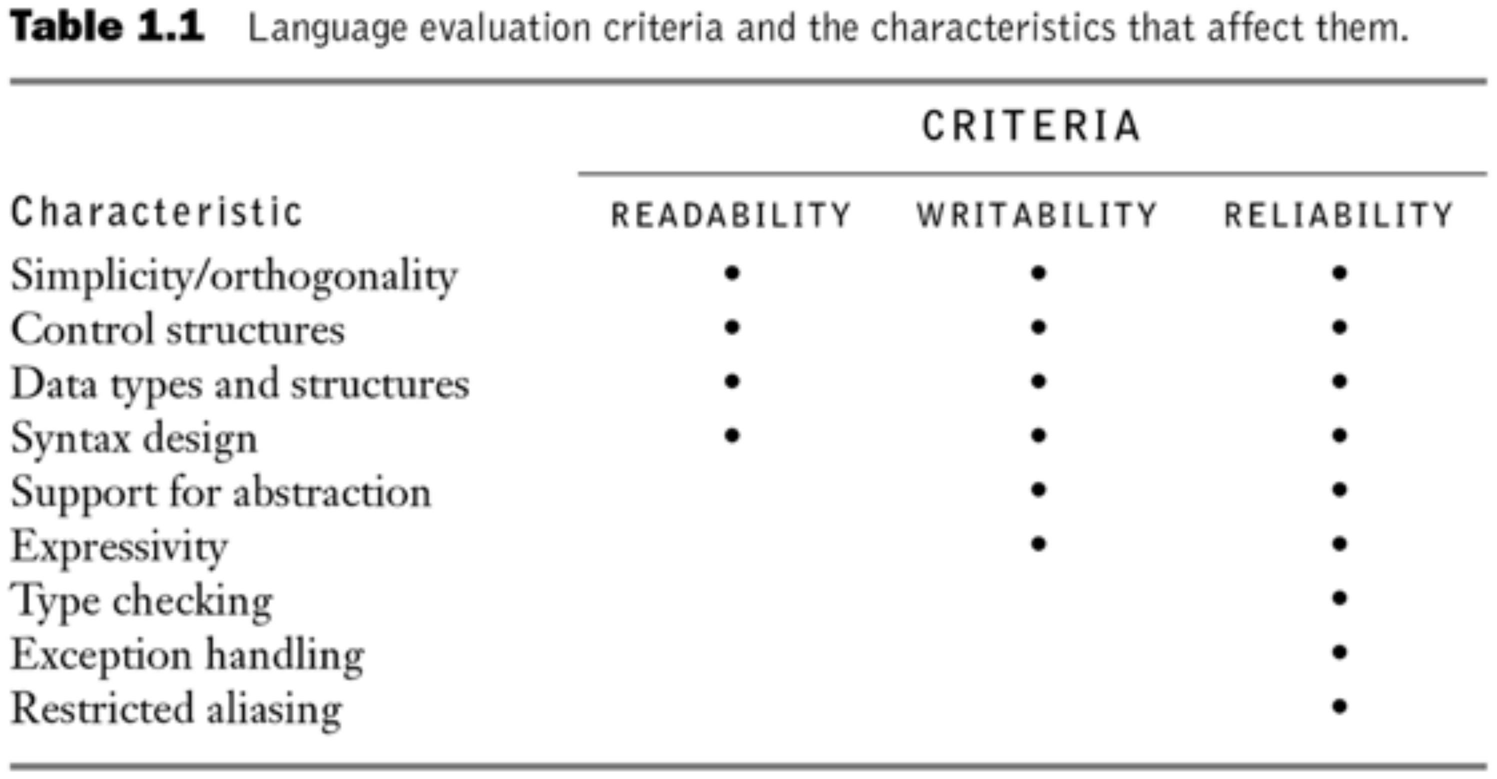


Table 1.1 Language evaluation criteria and the characteristics that affect them.

	CRITERIA		
Characteristic	READABILITY	WRITABILITY	RELIABILITY
Simplicity/orthogonality	•	•	•
Control structures	•	•	•
Data types and structures	•	•	•
Syntax design	•	•	•
Support for abstraction		•	•
Expressivity		•	•
Type checking			•
Exception handling			•
Restricted aliasing			•

Topics in Programming Languages

- Syntax and semantics.
- Lexical and syntax analysis.

- Names, bindings and scopes.
- Data types.
- Expressions and assignment statements.

- Statement-level control structures.
- Subprograms.