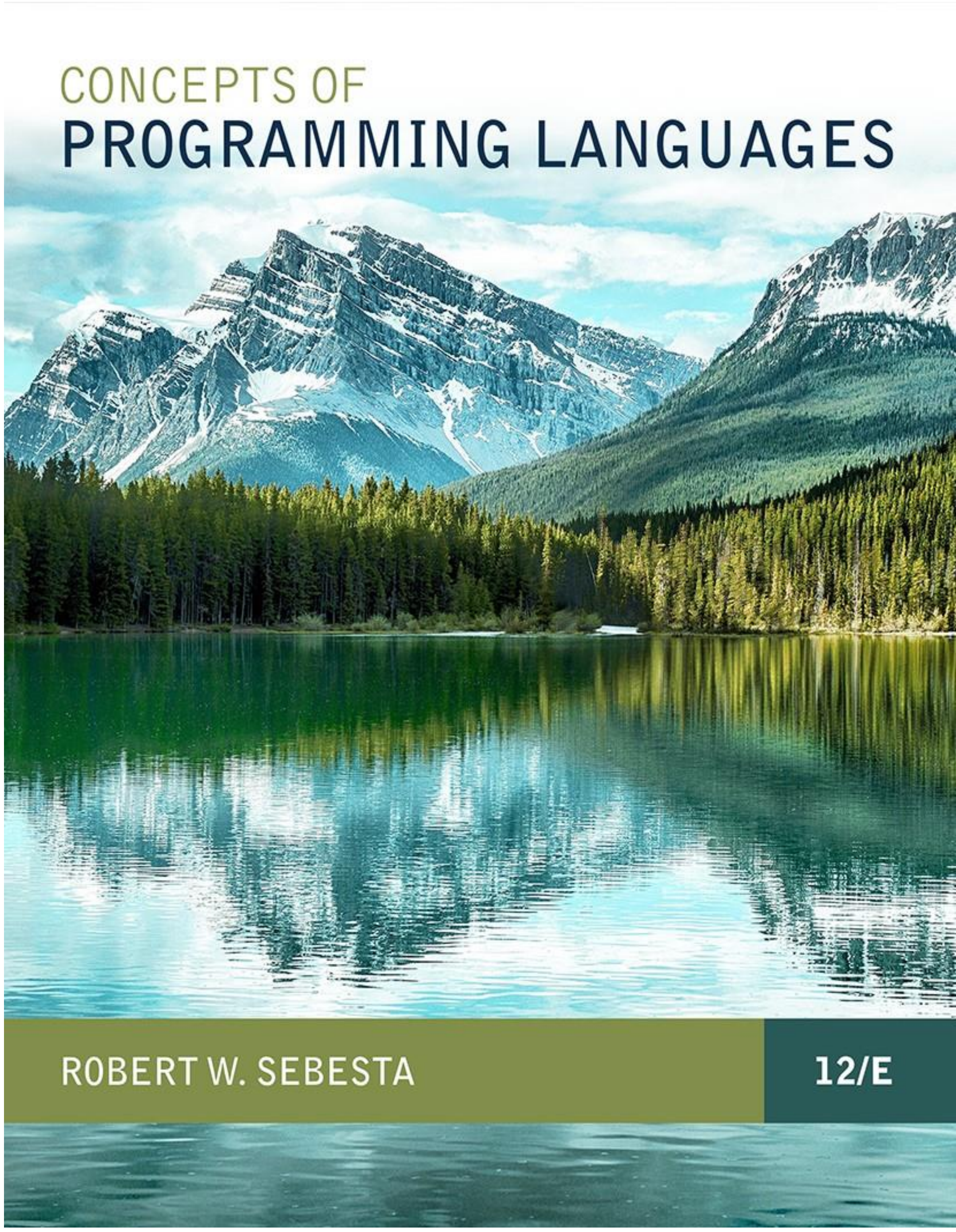
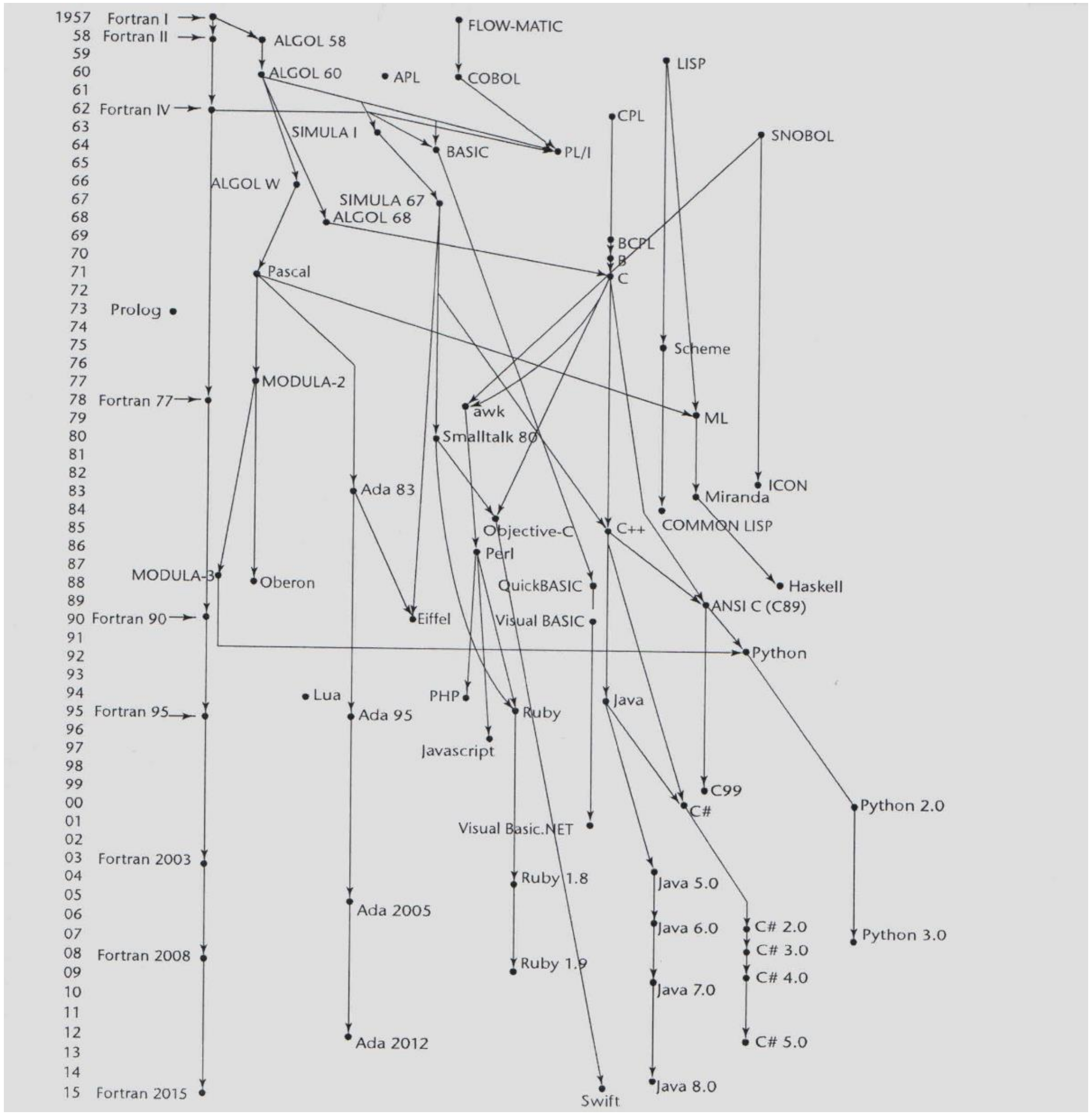
Chapter 2

프로그래밍언어의
발전사



ISBN 0-321-49362-1

주요 프로그래밍언어의 계보



Fortran

- Fortran 0: 1954 – not implemented
- Fortran I: 1957 – First implemented version
- Fortran II: Distributed in 1958
- Fortran IV: 1960-62; ANSI standard in 1966
- Fortran 77: the new standard in 1978
- Fortran 90: Dynamic arrays, Pointers, Recursion, **`CASE`**
  Fortran 95: relatively minor additions, plus some deletions
- Fortran 2003: support for OOP, procedure pointers,
  interoperability with C
- Fortran 2008: blocks for local scopes, co-arrays,
  `Do Concurrent`
- Fortran 2018

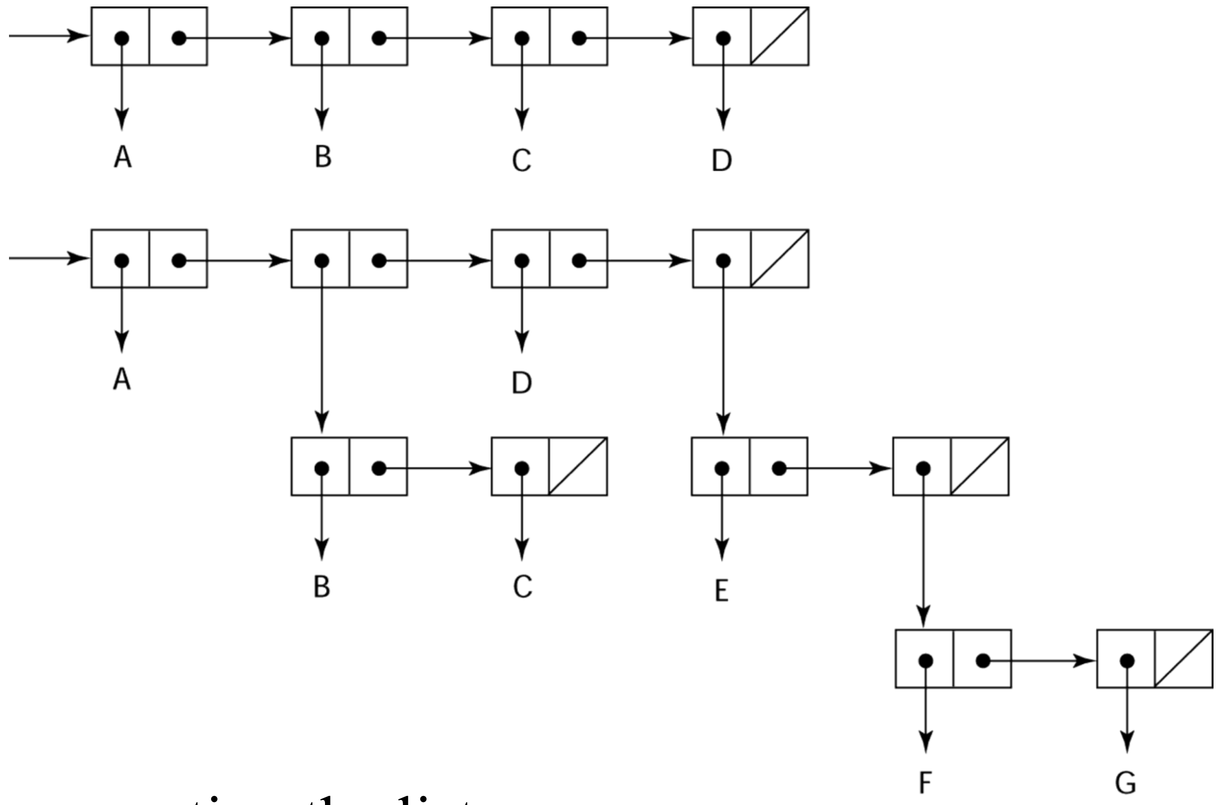
Fortran 평가

- Highly optimizing compilers (all versions before 90)
  - Types and storage of all variables are fixed before run time
- Dramatically changed forever the way computers are used

Functional Programming: Lisp

- LISt Processing language
  - Designed at MIT by McCarthy
- AI research needed a language to
  - Process data in lists (rather than arrays)
  - Symbolic computation (rather than numeric)
- Only two data types: atoms and lists
- Syntax is based on *lambda calculus*

Representation of Two Lisp Lists



Representing the lists `(A B C D)`
and `(A (B C) D (E (F G)))`

Lisp 평가

- Pioneered functional programming
  - No need for variables or assignment
  - Control via recursion and conditional expressions
- Still the dominant language for AI
- Common Lisp and Scheme are contemporary dialects of Lisp
- ML, Haskell, and F# are also functional programming languages, but use very different syntax

정교화를 향한 첫 단계: ALGOL 60

- ALGOL 58:
  - Concept of type was formalized
  - Names could be any length
  - Arrays could have any number of subscripts
  - Parameters were separated by mode (in & out)
  - Subscripts were placed in brackets
  - Compound statements (`begin ... end`)
  - Semicolon as a statement separator
  - Assignment operator was :=
  - `if` had an `else-if` clause
  - No I/O – "would make it machine dependent"
- ALGOL 60 was the result of efforts to design a universal language

ALGOL 60 개요

- New features
  - Block structure (local scope)
  - Two parameter passing methods
  - Subprogram recursion
  - Stack-dynamic arrays

  - Still no I/O and no string handling

ALGOL 60 평가

- Successes
  - It was the standard way to publish algorithms for over 20 years
  - All subsequent imperative languages are based on it
  - First machine-independent language
  - First language whose syntax was formally defined (BNF)
- Failure
  - Never widely used, especially in U.S.
  - Reasons
    - Lack of I/O and the character set made programs non-portable
    - Too flexible--hard to implement
    - Entrenchment of Fortran
    - Formal syntax description
    - Lack of support from IBM

사무기록의 전산화: COBOL

- First Design Meeting (Pentagon) – May 1959
- Design goals
  - Must look like simple English
  - Must be easy to use, even if that means it will be less powerful
  - Must broaden the base of computer users
  - Must not be biased by current compiler problems
- Design committee members were all from computer manufacturers and DoD branches
- Design Problems: arithmetic expressions? subscripts? Fights among manufacturers

COBOL 평가

- Contributions
  - First macro facility in a high-level language
  - Hierarchical data structures (records)
  - Nested selection statements
  - Long names (up to 30 characters), with hyphens
  - Separate data division
- First language required by DoD
  - would have failed without DoD
- Still the most widely used business applications language

시분할의 발단: Basic

- Designed by Kemeny & Kurtz at Dartmouth
- Design Goals:
  - Easy to learn and use for non-science students
  - Must be “pleasant and friendly”
  - Fast turnaround for homework
  - Free and private access
  - User time is more important than computer time
- Current popular dialect: Visual Basic
- First widely used language with time sharing

모든 사람을 위한 모든 것: PL/I

- Computing situation in 1964
  - Scientific computing: IBM 1620 and 7090, FORTRAN
  - Business computing: IBM 1401, 7080, COBOL
- Initial design concept
  - An extension of Fortran IV
- Initially called NPL (New Programming Language)
- Name changed to PL/I in 1965

PL/I 평가

- PL/I contributions
  - First unit-level concurrency
  - First exception handling
  - Switch-selectable recursion
  - First pointer data type
  - First array cross sections
- Concerns
  - Many new features were poorly designed
  - Too large and too complex

초기 동적 언어들: APL, SNOBOL

- 특징 : dynamic typing, dynamic storage allocation
- Variables are untyped: A variable acquires a type when it is assigned a value
- Storage is allocated to a variable when it is assigned a value

APL and SNOBOL

- APL(A Programming Language)
  - Designed as a hardware description language, 1960
    - Highly expressive (many operators, for both scalars and arrays of various dimensions)
    - Programs are very difficult to read
  - Still in use; minimal changes
- SNOBOL
  - Designed as a string manipulation language, 1964
  - Powerful operators for string pattern matching
  - Slower than alternative languages (and thus no longer used for writing editors)
  - Still used for certain text processing tasks

데이터 추상화의 발단: SIMULA 67

- Designed primarily for system simulation
- Based on ALGOL 60 and SIMULA I
- Primary Contributions
  - Coroutines – a kind of subprogram
  - Classes, objects, and inheritance

직교적 설계: ALGOL 68

- Is not a superset of ALGOL 60
- Source of several new ideas (even though the language itself never achieved widespread use)
- Design is based on the concept of orthogonality
  - A few basic concepts, plus a few combining mechanisms

ALGOL 68 평가

- Contributions
  - User-defined data structures
  - Reference types
  - Dynamic arrays (called flex arrays)
- Comments
  - Less usage than ALGOL 60
  - Had strong influence on subsequent languages, especially Pascal, C, and Ada

설계의 단순성: Pascal – 1971

- Developed by Wirth (a former member of the ALGOL 68 committee)
- Designed for teaching structured programming
- Small, simple, nothing really new
- Largest impact was on teaching programming
  - From mid-1970s until the late 1990s, it was the most widely used language for teaching programming

이식성을 갖는 시스템언어: C – 1972

- Designed for systems programming (at Bell Labs by Dennis Richie)
- Evolved primarily from BCPL and B, but also ALGOL 68
- Powerful set of operators, but poor type checking
- Initially spread through UNIX
- Though designed as a systems language, it has been used in many application areas

논리 기반 프로그래밍: Prolog

- Named from programming logic
- Based on formal logic
- Non-procedural
- Can be summarized as being an intelligent database system that uses an inferencing process to infer the truth of given queries
- Comparatively inefficient
- Few application areas

역사상 최대 설계 노력: Ada

- Huge design effort, involving hundreds of people, much money, and about eight years
- Sequence of requirements (1975-1978) in DoD
- Named Ada after Augusta Ada Byron, the first programmer
- Contributions: Packages, Exception handling, Generic program units, Concurrency
- First compilers were very difficult; the first really usable compiler came nearly five years after the language design was completed

Ada 95

- Ada 95 (began in 1988)
  - Support for OOP through type derivation
  - Better control mechanisms for shared data
  - New concurrency features
  - More flexible libraries
- Ada 2005
  - Interfaces and synchronizing interfaces
- Popularity suffered because the DoD no longer requires its use but also because of popularity of C++

객체지향프로그래밍: Smalltalk

- First full implementation of an object-oriented language (data abstraction, inheritance, and dynamic binding)
- Pioneered the graphical user interface design
- Promoted OOP

명령형과 객체지향 특징의 결합: C++

- Developed at Bell Labs by Stroustrup in 1980
- Evolved from C and SIMULA 67
- Facilities for object-oriented programming, taken partially from SIMULA 67
- A large and complex language, in part because it supports both procedural and OO programming
- Rapidly grew in popularity, along with OOP
- ANSI standard approved in November 1997
- Backward compatible(후방 호환가능)

관련 OOP Language

- Swift – a replacement for Objective-C
  - Released in 2014
  - Two categories of types, classes and struct, like C#
  - Used by Apple for systems programs
- Delphi – another related language
  - A hybrid language, like C++
  - Began as an object-oriented version of Pascal
  - Designed by Anders Hejlsberg, who also designed Turbo Pascal and C#

명령형 기반의 객체지향 언어: Java

- Developed at Sun in the early 1990s
  - C and C++ were not satisfactory for embedded electronic devices
- Based on C++
  - Significantly simplified (does not include **`struct`**, **`union`**, **`enum`**, pointer arithmetic, and half of the assignment coercions of C++)
  - Supports *only* OOP
  - Has references, but not pointers
  - Includes support for applets and a form of concurrency

Java 평가

- Eliminated many unsafe features of C++
- Supports concurrency
- Libraries for applets, GUIs, database access
- Portable: Java Virtual Machine concept, JIT compilers
- Widely used for Web programming
- Use increased faster than any previous language
- Most recent version, 8, released in 2014

Scripting Languages for the Web

- Perl
  - Designed by Larry Wall—first released in 1987
  - Variables are statically typed but implicitly declared
  - Three distinctive namespaces, denoted by the first character of a variable’s name
  - Powerful, but somewhat dangerous
  - Gained widespread use for CGI programming on the Web
  - Also used for a replacement for UNIX system administration language
- JavaScript
  - Began at Netscape, but later became a joint venture of Netscape and Sun Microsystems
  - A client-side HTML-embedded scripting language, often used to create dynamic HTML documents
  - Purely interpreted
  - Related to Java only through similar syntax
- PHP
  - PHP: Hypertext Preprocessor, designed by Rasmus Lerdorf
  - A server-side HTML-embedded scripting language, often used for form processing and database access through the Web
  - Purely interpreted

Scripting Languages for the Web

- Python
  - An OO interpreted scripting language
  - Type checked but dynamically typed
  - Used for form processing
  - Supports lists, tuples, and hashes
- Ruby
  - Designed in Japan by Yukihiro Matsumoto (a.k.a, “Matz”)
  - Began as a replacement for Perl and Python
  - A pure object-oriented scripting language
    - All data are objects
  - Most operators are implemented as methods, which can be redefined by user code
  - Purely interpreted

.NET Language: C#

- Part of the .NET development platform (2000)
- Based on C++ , Java, and Delphi
- Includes pointers, delegates, properties, enumeration types, a limited kind of dynamic typing, and anonymous types
- Is evolving rapidly