제 2 장

프로그래밍 언어의 발전사

제 2 장 프로그래밍 언어의 발전사

❖ 목적 :

- Zuse's Plankalkül
- Minimal Hardware Programming: Pseudocodes
- The IBM 704 and Fortran
- Functional Programming: LISP
- The First Step Toward Sophistication: ALGOL 60
- Computerizing Business Records: COBOL
- The Beginnings of Timesharing: BASIC
- Everything for Everybody: PL/I
- Two Early Dynamic Languages: APL and SNOBOL
- The Beginnings of Data Abstraction: SIMULA 67
- Orthogonal Design: ALGOL 68
- Some Early Descendants of the ALGOLs
- Programming Based on Logic: Prolog
- History's Largest Design Effort: Ada

제 2 장 프로그래밍 언어의 발전사

❖목적 :

- Object-Oriented Programming: Smalltalk
- Combining Imperative ad Object-Oriented Features: C++
- An Imperative-Based Object-Oriented Language: Java
- Scripting Languages
- A C-Based Language for the New Millennium: C#
- Markup/Programming Hybrid Languages

제 2 장 프로그래밍 언어의 발전사

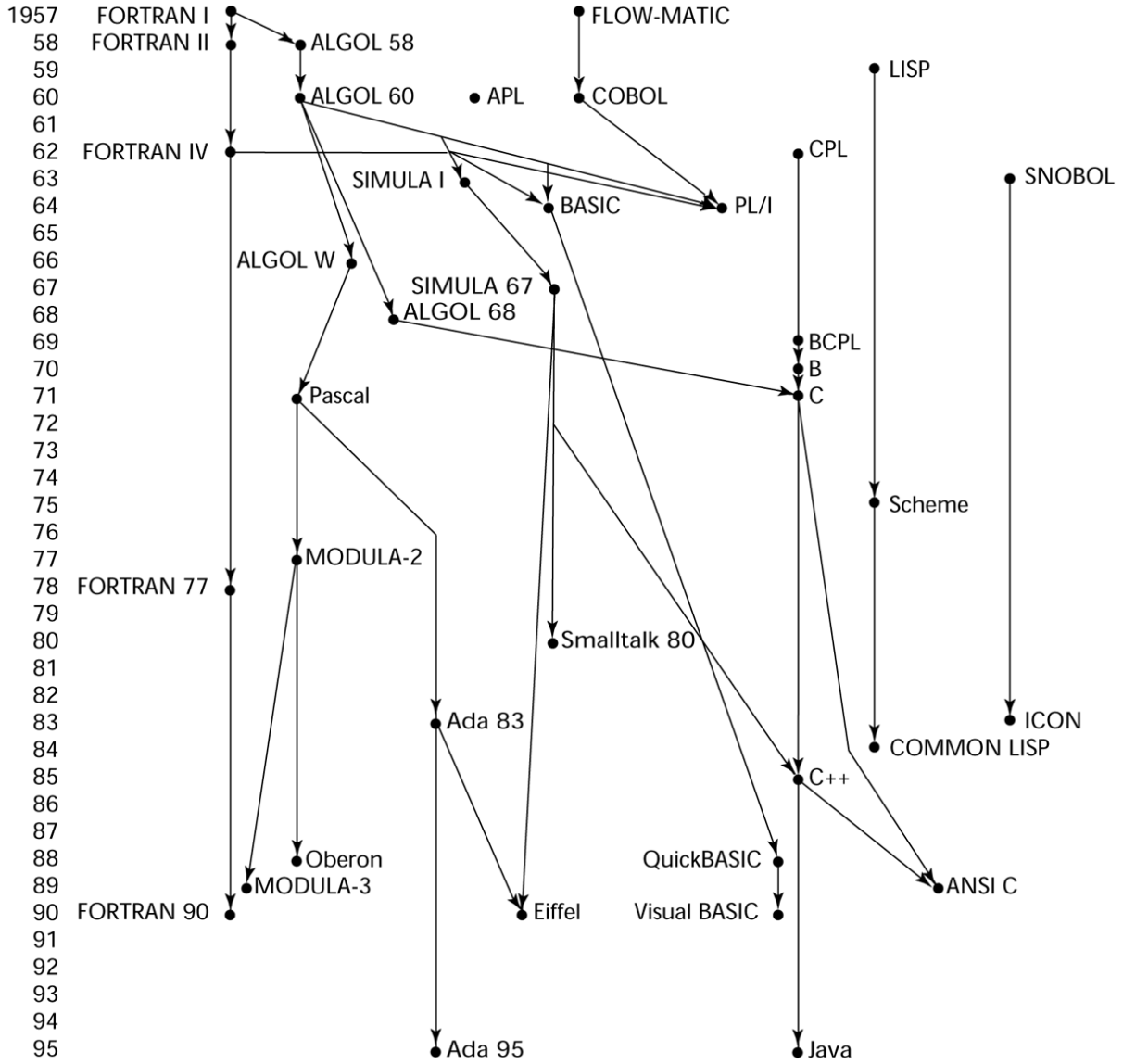
❖ 프로그래밍 언어의 역사

- 1940년대 말 폰 노이만(von Neumann)에 의해 세계 최초로 프로그램 내장 방식의 컴퓨터가 등장
- 일반적으로 말하는 프로그래밍 언어는 프로그램 내장 방식 컴퓨터에 대한 계산 과정을 기술할 수 있을 때부터 시작
- 메모리에 프로그램을 저장하고 프로그램 명령어들을 차례대로 실행
- 현재 대부분의 컴퓨터들이 이 방식을 적용

- **년도별 프로그래밍 언어의 특징과 언어의 종류**

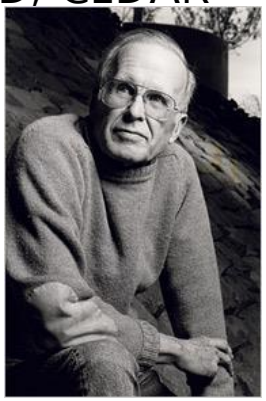
시기	특징	주요 프로그래밍 언어
1950년대 이전	디지털 컴퓨터 이전의 언어	Plankalkül
1950년대	최초의 프로그래밍 언어	어셈블리어, Autocode, FORTRAN, APL, LISP
1960년대	프로그래밍 언어의 범람	ALGOL 60, COBOL, PL/I, BASIC, Logo
1970년대	단순성, 추상화, 프로그래밍 언어에 대한 연구	Pascal, C, Smalltalk, Prolog, Scheme, CLU
1980년대	새로운 방향과 객체지향의 부상	Modula 2, Ada, C++, Perl, Python
1990년대	통합, 인터넷, 라이브러리, 스크립트	Visual Basic, Java, JavaScript, PHP

일반 프로그래밍 언어의 계보



2.3 Fortran

- I, II, III, IV, 77, 86, 90, 95, 2003, 2008, HPF, FORTRAN D, CEDAR FORTRAN, KSR
  - ✓ **과학 계산 목적으로 IBM에서 설계**
  - ✓ IBM 704
  - ✓ **John Backus** (1924 – 2007)
    - ♣ IBM에 있을 때 FORTRAN 개발자, 그 후에 MIT 교수
    - ♣ Backus–Naur form (BNF) - 형식언어문법(formal language syntax)을 정의하기 위한 개념
    - ♣ 1977년 Turing Award 수상 - function-level programming 개념인 "Can Programming Be Liberated from the von Neumann Style?"



  - ✓ FORTRAN IV – 문자열 처리, 논리 루프 제어문, IF ~THEN ~ELSE
  - ✓ FORTRAN 90 – 동적 배열, 레코드, 포인터, 다중 선택문, 모듈 => 그래서 재귀적 호출 가능
  - ✓ FORTRAN 95 – Forall 구조 추가(병렬 처리)
  - ✓ Fortran 2003 – 객체지향프로그래밍
  - ✓ Fortran 2008 - 블록

2.4 LISP

- LISP(LISt Processor) -
  - ✓ 1960년 미국 MIT의 **존 매카시** (J. McCarthy) 가 개발
  - ✓ 리스트 처리용 프로그래밍 언어. 복잡한 수식 단위를 간략하게 표현한 언어로서 과학 기술 계산과 인공 지능용 언어임.
  - ✓ 함수형 언어
- 특징 :
  - ✓ Atom과 리스트
    - ♣ 초기의 리스프에 자료형은 Atom과 리스트로 구분되었다. Atom은 상수나 심볼을 뜻하고, 리스트는 이러한 Atom 또는 다른 리스트를 원소로 포함하며 유한한 길이를 가진다.
  - ✓ 문법
    - ♣ 기본적으로 전위 표기법(Prefix Notation)을 사용한다. 즉, 통상 프로그래밍 언어에서 1 + 2 를 수행하는 코드를 작성하면 (+ 1 2)
    - ♣ (/ (float 14)(float 4)) ➔ 3.5
    - ♣ (CAR '(FAST COMPUTERS ARE NICE)) ➔ FAST
    - ♣ (CDR '(FAST COMPUTERS ARE NICE)) ➔ COMPUTERS ARE NICE
- Common Lisp
  - ✓ 다목적, 멀티 패러다임 언어로 절차형, 함수형, OOP를 모두 지원하는 다이내믹 프로그래밍 언어의 표준안이다. 즉 이 자체로는 그냥 언어 사양일 뿐이다.

2.4 LISP

- **존 매카시**(John McCarthy, 1927년-2011년)
- 는 미국의 전산학자이자 인지과학자이다.
- 인공지능(AI; Artificial Intelligence)이라는
- 단어는 1955년 존 매카시가 발표한 "지능이
- 있는 기계를 만들기 위한 과학과 공학"
- 이라는 논문에 처음으로 등장.
- 1958년 MIT에서 인공지능의 기본 언어인 리스프 프로그래밍 언어를 설계 및 구현
- 인공지능에 대한 연구 업적을 인정받아 1971년 튜링상을 수상.
  - ✓

2.5 Algol 60

- ALGOL(ALGOrithmic Language)
  - ✓ Backus와 Naur를 포함한 위원회
- 새로운 언어에 대한 목적 :
  - ✓ 언어의 구문은 가능한 표준 수학적 표기에 가까워야 함. => 그러면 판독 가능(readability)
  - ✓ 출판물에 포함된 알고리즘들의 서술을 위해서 그 언어를 사용하는 것이 가능해야 함.
  - ✓ 새로운 언어로 작성된 프로그램은 기계어로 기계적으로 번역되어야 함.
- Algol 60에 대한 새로운 개발 사항 :
  - ✓ 블록 구조 개념 도입 – 자료의 영역을 지역화
  - ✓ 값 호출(call by value)과 이름 호출(call by name) 허용(9장 설명)
  - ✓ 재귀법 허용
  - ✓ 스택-동적 배열 허용(6장에서 설명)
- Algol 언어의 후손들 –
  - ✓ PL/I, SIMULA 67, ALGOL 68, C, PASCAL, ADA, C++, JAVA, C#

2.6 COBOL

- COBOL(COmmon Business-Oriented Language) –
  - ✓ 1959년에 CODASYL(코다실 - 데이터 시스템 언어 회의 (Conference/Committee on Data Systems Languages)의 준말로, 수많은 컴퓨터에서 사용할 수 있는 표준 프로그래밍 언어 개발을 안내한다. 목적은 더 효율적인 데이터 시스템 분석, 디자인, 구현을 제고해 나가는 데 있다.) 이 설계하였음.
  - ✓ 1968년 표준화되어 그 뒤로 4차례 개정되었다. 확장에는 구조적, 객체 지향프로그래밍의 지원을 포함한다.

- 특징 :
  - ✓ 사무용으로 설계된, 영어와 같은 컴퓨터 프로그래밍 언어이다. 절차적 , 명령형 언어
  - ✓ 코볼은 영어와 비슷한 문법을 갖고 있으며, 자체 문서화 및 높은 가독성을 염두에 두고 설계 (배정문을 MOVE x TO y).
  - ✓ 코볼 코드는 4개의 Division(IDENTIFICATION, ENVIRONMENT, DATA, PROCEDURE)으로 나뉘며 이 안에 엄격한 계층적 섹션, 문단, 문장들을 포함한다.

2.7 BASIC

- BASIC(Beginner's All-purpose Symbolic Instruction Code) –
  - ✓ Dartmouse 대학의 John Kemeny와 Thomas Kurtz의해 설계

- 목적 :
  - ✓ 비과학 전공학생들이 배우고 사용하기가 쉬워야 한다.
  - ✓ 즐겁고 친숙해야 한다.
  - ✓ 숙제를 위해서 빠른 작업 처리 시간을 제공해야 한다.
  - ✓ 자유롭고 개인적인 접근을 허용

- 특징 :
  - ✓ 시분할(Time Sharing) multiprogramming

2.8 PL/I

- PL/I(Programming Language One)
  - ✓ IBM과 SHARE(IBM 과학 사용자 그룹)
  - ✓ IBM에서 자사의 메인프레임에서 사용하기 위해서 개발
- 과학 기술 계산과 사무 처리에 좋은 언어
- 특징 :
  - ✓ Multi-tasking 지원
  - ✓ 예외 처리 가능
  - ✓ 부프로그램들이 재귀적으로 사용되는 것이 허용
  - ✓ 포인터가 추가

2.9 APL과 SNOBOL

- APL :
  - ✓ APL(A Programming Language)은 IBM의 Kenneth E. Iverson에 의해 설계
    - ♣ 강력하고 수학적인 개념 사용, 일반적인 프로그래밍 언어의 규칙을 위배
- SNOBOL
  - ✓ **SNOBOL**(StriNg Oriented and symBOlic Language)은 AT&T 벨 연구소에서 David J. Farber, Ralph Griswold, Ivan P. Polonsky가 개발한 컴퓨터 프로그래밍 언어 시리즈이며, SNOBOL4까지 개발
  - ✓ 텍스트 처리를 위해서 특별하게 고안

2.10 SIMULA 67

- SIMULA(SIMUlation LAnguage)와 SIMULA(SIMple Universal Language) 의 두 곳에서 유래했다.
  - ✓ 노르웨이인 Kristen Nygaard와 Ole-Johan Dahl은 알골을 시뮬레이션 목적으로 확장하여 개발한 언어이다.
- 특징 :
  - ✓ 블록 구조와 제어문을 Algol 60으로 부터 가져와서 Algol 60 확장 언어
  - ✓ 코루틴(coroutine) – 호출자와 피호출 부프로그램이 대부분 명령형 언어에서 갖는 주종(master/slave)의 관계가 아닌 서로간에 동등한 관계를 갖는 프로그램.
  - ✓ 코루틴을 지원하기 위해 클래스 구조 개발 – 이는 자료 추상화 개념
  - ✓ => 객체지향 프로그래밍(object-oriented programming)

2.12 Pascal

- 파스칼(Pascal)은 1980년대와 1990년대 초반에 걸쳐 널리 사용된 프로그래밍 언어로, 당대의 가장 인기있는 교육용 언어 중 하나였으며, 현재까지도 폭넓은 분야에서 사용되고 있다.
  - ✓ 파스칼은 1969년에 스위스 ETH 취리히의 컴퓨터 과학자 니클라우스 비르트가 개발하였다.
  - ✓ 파스칼이라는 이름은 프랑스의 수학자이자 철학자 블레즈 파스칼의 이름을 딴 것이다. 포인터를 사용한 구조적 프로그래밍을 그 특징으로 한다.
  - ✓ 알골 60의 영향을 받은 까닭에, 같은 시기에 마찬가지 영향을 받아 제작된 C와 여러가지 면에서 유사한 점을 갖는다.
  - ✓ 초기의 파스칼에 비해 많은 부분이 추가, 개선되고 다른 언어의 장점들을 따와 상용 파스칼 컴파일러인 델파이는 C++과 거의 기능 차이가 없다.
  - ✓ 82페이지 프로그램

2.13 Prolog

- 프롤로그(Prolog)
  - ✓ 논리형 프로그래밍 언어이다.
  - ✓ 이름은 '논리 프로그래밍'을 의미하는 프랑스어: *programmation en logique*에서 온 것이다.
  - ✓ 1973년 프랑스 마르세유대학교의 알랭 콜메르(Alan Colmerauer)가 개발한 언어로서, 논리식을 토대로 하여 오브젝트와 오브젝트 간의 관계에 관한 문제를 해결하기 위해 사용한다.
  - ✓ 추론 기구를 간결하게 표현할 수 있기 때문에 인공지능이나 계산 언어학 분야, 특히 프롤로그가 만들어진 목적이었던 자연언어 처리 분야에서 많이 사용된다.

2.14 Ada

- 에이다(Ada)
  - ✓ 미국 국방성(Department of Defence: DoD)을 위해서 개발.
  - ✓ 에이다는 컴퓨터 프로그래밍을 발명하는 데 공헌한 에이다 러브레이스 (1815년-1852년)의 이름을 딴 것이다

- 특징
  - ✓ 구조화되고, 정적인 형태를 가지고, 명령적이며, 객체 지향적인 고급 컴퓨터 프로그래밍 언어이다.
  - ✓ 에이다는 C나 C++의 몇 가지 작업이 같지만, 에이다는 매우 강력한 유형 시스템의 언어이다.
  - ✓ 90페이지 프로그램

2.15 Smalltalk

- **스몰토크**(Smalltalk)
  - ✓ 동적 형을 지원하는 객체 지향 프로그래밍 언어(= class + 상속 + 다형성)이다.
  - ✓ 스몰토크는 제록스 파크(PARC)에서 앨런 케이, 댄 잉갤스, 테드 캘러, 에이들 골드버그가 만들었다.
  - ✓ 최초로 GUI를 제공하는 언어.
- 특징
  - ✓ 순수 객체지향 언어이다. C++나 자바와는 달리 원시 자료형이 없고, 모든 정수, 부동소수, 문자열, 블록을 포함한 모든 것이 객체며 클래스가 있다.

- **앨런 케이**(Alan Curtis Kay, 1940년 5월 17일 ~)는
- 미국의 전산학자이다. 객체 지향 프로그래밍과 사용자
- 인터페이스 디자인 분야의 선구자.
- 1970년에 제록스사의 팔로 알토 연구소(PARC)에 입사
- 하였으며 스몰토크 프로그래밍 언어를 이용한 네트워크 워크스테이션의 초기 모델을 개발하는 핵심 연구원으로 활동했다. 여기서 개발된 디자인은 이후 애플사의 매킨토시 컴퓨터 개발에 큰 영향을 주었다. 그는 객체 지향 프로그래밍(OOP, Object Oriented Programming)의 아이디어를 만들어 낸 창시자 중 한 명이다. 그는 현대적인 윈도 기반의 그래픽 사용자 인터페이스(GUI)를 설계한 장본인
- 2003년 객체지향 프로그래밍을 개척한 공로로 ACM 튜링상을 수상

2.18 스크립트 언어

- **Scripting languages(스크립트 언어)** - 스크립트라고 불리는 명령들의 리스트를 한 개의 실행될 파일로 작성
  - ✓ **sh(shell의 축약)** - 파일 관리나 간단한 파일 필터링 등과 같은 유틸리티 함수를 수행하는 시스템
  - ✓ **Awk** – 벨 연구소의 Al Aho, Brian Kernighan, Peger Weinberger가 개발
  - ✓ **Perl** - Larry Wall이 개발, sh와 awk의 결합 형태로 시작하여 언어로 발전, CGI 프로그래밍에 이상적,
    - ♣ 변수는 묵시적으로 선언 - $로 시작하면 스칼라 변수, @로 시작하면 배열
    - ♣ p105
  - ✓ **JavaScript** - Netscape, 웹 서버와 브라우저에서 사용
    - ♣ 여기에 변수, 제어 구조, 함수 등이 추가되어 언어로 발전
  - ✓ **PHP** - Web 응용을 위해서 특별하게 설계된 HTML에 포함된 서버측 스크립트 언어
  - ✓ **파이썬(Python)** –
    - ♣ 1991년 프로그래머인 귀도 반 로섬(Guido van Rossum)이 발표한 고급 프로그래밍 언어로, 플랫폼 독립적이며 인터프리터식, 객체 지향적, 동적 타이핑(dynamically typed) 대화형 언어이다.
    - ♣ 파이썬은 비영리의 파이썬 소프트웨어 재단이 관리하는 개방형, 공동체 기반 개발 모델을 가지고 있다. C언어로 구현된 C파이썬 구현이 사실상의 표준이다.

2.18 스크립트 언어

- 파이썬은 무료이며 누구나 다운받아 사용 가능 하다.
- 파이썬은 초보자부터 전문가까지 사용자층을 보유하고 있다. 동적 타이핑(dynamic typing) 범용 프로그래밍 언어로, 펄 및 루비와 자주 비교된다. 다양한 플랫폼에서 쓸 수 있고, 라이브러리(모듈)가 풍부하여, 대학을 비롯한 여러 교육 기관, 연구 기관 및 산업계에서 이용이 증가하고 있다.
- 파이썬은 기본적으로 해석기(인터프리터) 위에서 실행될 것을 염두에 두고 설계되었다.
- 주요 특징
  - 동적 타이핑(dynamic typing). (실행 시간에 자료형을 검사한다.)
- 해석 프로그램의 종류
  - C파이썬: C로 작성된 인터프리터.
  - 스택리스 파이썬: C 스택을 사용하지 않는 인터프리터.
  - 자이썬: 자바 가상 머신 용 인터프리터. 과거에는 제이파이썬(JPython)이라고 불렸다.