

---

# 프로그래밍 언어의 역사#2

---

Concepts of Programming Languages

2024년 1학기

한양대학교 인공지능대학원  
임덕선

# 목차

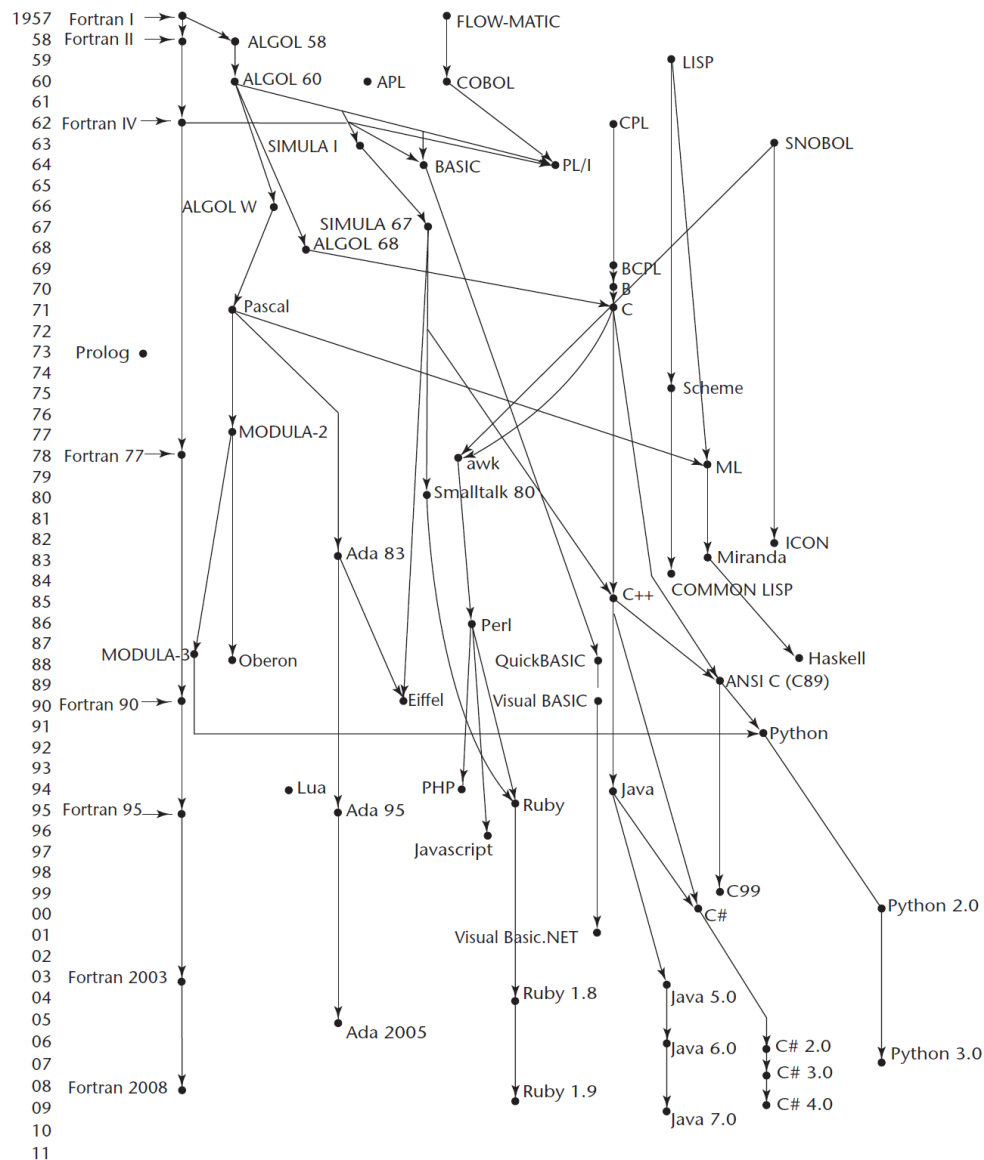
---

## □ 프로그래밍 언어의 계보

## □ 프로그래밍 언어 및 예제

- Fortran : Fortran IV, Fortran 95
- Algol60
- BASIC
- LISP
- COBOL
- Pascal
- C
- Ada
- Java
- C#

# 프로그래밍 언어의 계보



### Figure 2.1

Hanyang University – Concept of Progr

## Genealogy of common high-level programming languages

# 프로그래밍 언어(Fortran)

- FORTRAN은 수식(Formula) 변환기(Translation)의 약자
- 포트란은 기후, 우주항공, 유체 및 구조해석, 군사과학, 금융계산 등 거의 모든 산업 분야의 초대형 과학 계산 문제의 프로그래밍에 필수적인 언어
- 전문적인 과학 계산을 풀기 위해 프로그래밍 하는데 있어서 간단 명료하여 프로그래머는 신속하게 프로그래밍할 수 있음



John Backus, the inventor of FORTRAN



First page of the first FORTRAN manual with an autograph of John Backus

# IBM 704 and Fortran

## ❑ Fortran 0: 1954 - not implemented

## ❑ Fortran I: 1957

- Designed for the new IBM 704, which had index registers and floating point hardware
  - This led to the idea of compiled programming languages, because there was no place to hide the cost of interpretation (no floating-point software)
- Environment of development
  - Computers were small and unreliable
  - Applications were scientific
  - No programming methodology or tools
  - Machine efficiency was the most important concern

## ❑ Impact of environment on design of Fortran I

- No need for dynamic storage
- Need good array handling and counting loops
- No string handling, decimal arithmetic, or powerful input/output (for business software)

## ❑ First implemented version of Fortran

- Names could have up to six characters
- Post-test counting loop (**DO**)
- Formatted I/O
- User-defined subprograms
- Three-way selection statement (arithmetic **IF**)
- No data typing statements

## ❑ First implemented version of FORTRAN

- No separate compilation
- Compiler released in April 1957, after 18 worker-years of effort
- Programs larger than 400 lines rarely compiled correctly, mainly due to poor reliability of 704
- Code was very fast
- Quickly became widely used



## ❑ Distributed in 1958

- Independent compilation
- Fixed the bugs

## ❑ Evolved during 1960-62

- Explicit type declarations
- Logical selection statement
- Subprogram names could be parameters
- ANSI standard in 1966

## ❑ Became the new standard in 1978

- Character string handling
- Logical loop control statement
- **IF-THEN-ELSE** statement

## ❑ Most significant changes from Fortran 77

- Modules
- Dynamic arrays
- Pointers
- Recursion
- **CASE** statement
- Parameter type checking

# Latest versions of Fortran

---

- ❑ Fortran 95 – relatively minor additions, plus some deletions
- ❑ Fortran 2003 - ditto

# Fortran Evaluation

---

- ❑ **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**
- ❑ **Characterized as the *lingua franca* of the computing world**

# 프로그래밍 언어 예제(FORTRAN IV)

| 1 | ~    | 5  | 6  | 7   |
|---|------|----|----|---|
| C | Fort | r  | an | IV Example  |
|   |      |    |    | DIMENSION LIST(99)                                  |
|   |      |    |    | IRESULT = 0   |
|   |      |    |    | ISUM = 0  |
|   |      |    |    | READ 60, LEN  |
|   |      |    |    | IF ((LEN .GT. 0) .AND. (LEN .LT. 100)) GO TO 101    |
|   |      |    |    | GO TO 102   |
|   | 101  |    |    | DO 55 ICOUNT = 1, LEN, 1                            |
|   |      |    |    | READ 70, LIST(ICOUNT)                               |
|   |      |    |    | ISUM = ISUM + LIST(ICOUNT)                          |
|   | 55   |    |    | CONTINUE  |
| C | ---- |    |    | Computer the Average                                |
|   |      |    |    | AVERAGE = ISUM / LEN                                |
|   |      |    |    | DO 66 ICOUNT = 1, LEN, 1                            |
|   |      |    |    | IF (LIST(ICOUNT).GT. AVERAGE) IRESULT = IRESULT + 1 |
|   | 66   |    |    | CONTINUE  |
| C | ---- |    |    | Print the Result                                    |
|   |      |    |    | WRITE 80, IRESULT                                   |
|   |      |    |    | GO TO 103   |
|   | 102  |    |    | WRITE 90  |
|   | 103  |    |    | STOP  |
|   | 60   |    |    | FORMAT(0X, I2)                                      |
|   | 70   |    |    | FORMAT(2X, I3)                                      |
|   | 80   |    |    | FORMAT(5X, '*** Result is ***', 2X, I5)             |
|   | 90   |    |    | FORMAT(5X, 'Error - list length is not legal)       |
|   |      |    |    | END   |
| C | Data |    |    |   |
| 5 | 0    |    |    |   |
|   |      | 75 |    |   |
|   |      | 96 |    |   |
|   |      | 88 |    |   |
|   |      | 76 |    |   |
|   |      | :  |    |   |

72

~ 80

# 프로그래밍 언어 예제(FORTRAN 95)

```
! Fortran 95 Example program
! Input:  An integer, List_Len, where List_Len is less
!         than 100, followed by List_Len-Integer values
! Output: The number of input values that are greater
!         than the average of all input values
Implicit none
Integer Dimension(99) :: Int_List
Integer :: List_Len, Counter, Sum, Average, Result
Result= 0
Sum = 0
Read *, List_Len
If ((List_Len > 0) .AND. (List_Len < 100)) Then
! Read input data into an array and compute its sum
  Do Counter = 1, List_Len
    Read *, Int_List(Counter)
    Sum = Sum + Int_List(Counter)

  End Do
! Compute the average
  Average = Sum / List_Len
! Count the values that are greater than the average
  Do Counter = 1, List_Len
    If (Int_List(Counter) > Average) Then
      Result = Result + 1
    End If
  End Do
! Print the result
  Print *, 'Number of values > Average is:', Result
Else
  Print *, 'Error - list length value is not legal'
End If
End Program Example
```



# 프로그래밍 언어(ALGOL)

- 알골(ALGOL)은 미국에서 만들어진 포트란에 대항하여 유럽 중심으로 개발된 프로그래밍 언어
- ALGO<sup>r</sup>ithmic Language를 줄여 붙여진 이름으로 알고리즘의 연구개발에 이용하기 위한 목적으로 만들어짐
- 1958년 취리히에서 열린 국제대회에서 제안된 것이 그 기원으로 여겨짐
- 파스칼, C언어 등 이후 언어의 발전에 큰 영향을 주었음

# The First Step Toward Sophistication: ALGOL 60

## ❑ Environment of development

- FORTRAN had (barely) arrived for IBM 70x
- Many other languages were being developed, all for specific machines
- No portable language; all were machine-dependent
- No universal language for communicating algorithms

## ❑ ALGOL 60 was the result of efforts to design a universal language

# Early Design Process

---

**❑ ACM and GAMM met for four days for design (May 27 to June 1, 1958)**

**❑ Goals of the language**

- Close to mathematical notation
- Good for describing algorithms
- Must be translatable to machine code

# 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”

## ❑ Modified ALGOL 58 at 6-day meeting in Paris

## ❑ New features

- Block structure (local scope)
- Two parameter passing methods
- Subprogram recursion
- Stack-dynamic arrays
- Still no I/O and no string handling

# 프로그래밍 언어 예제(ALGOL 60)

```
comment ALGOL 60 Example Program
Input:  An integer, listlen, where listlen is less than
        100, followed by listlen-integer values
Output: The number of input values that are greater than
        the average of all the input values ;

begin
  integer array intlist [1:99];
  integer listlen, counter, sum, average, result;
  sum := 0;
  result := 0;
  readint (listlen);
  if (listlen > 0) ^ (listlen < 100) then
    begin
comment Read input into an array and compute the average;
      for counter := 1 step 1 until listlen do
        begin
          readint (intlist[counter]);
          sum := sum + intlist[counter]
        end;
comment Compute the average;
      average := sum / listlen;
comment Count the input values that are > average;
      for counter := 1 step 1 until listlen do
        if intlist[counter] > average
          then result := result + 1;
comment Print result;
      printstring("The number of values > average is:");
      printint (result)
    end
  else
    printstring ("Error-input list length is not legal";
  end
```

# 프로그래밍 언어(BASIC)

□ BASIC은 Beginner's All-purpose Symbolic Instruction Code의 약자

□ New Hampshire의 Dartmouth 대학의 두 수학자 John Kemeny와 Thomas Kurtz에 의해 처음으로 설계됨

- 이들은 과학이나 수학분야가 아닌 학생들이 컴퓨터를 사용할 수 있는 목적으로 개발

□ BASIC은 시분할(time shared)기능을 제공

- 다중 도시 사용자들에 의한 터미널들을 통한 개인적인 접근이 가능하였음

# The Beginning of Timesharing: BASIC

---

## ❑ 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



# 프로그래밍 언어 예제(BASIC)

```
REM  BASIC Example Program
REM  Input:  An integer, listlen, where listlen is less
REM          than 100, followed by listlen-integer values
REM  Output: The number of input values that are greater
REM          than the average of all input values
      DIM intlist(99)
      result = 0
      sum = 0
      INPUT listlen
      IF listlen > 0 AND listlen < 100 THEN
REM  Read input into an array and compute the sum
        FOR counter = 1 TO listlen
          INPUT intlist(counter)
          sum = sum + intlist(counter)
        NEXT counter
REM  Compute the average
        average = sum / listlen
REM  Count the number of input values that are > average
        FOR counter = 1 TO listlen
          IF intlist(counter) > average
            THEN result = result + 1
        NEXT counter
REM  Print the result
        PRINT "The number of values that are > average is:";
          result
      ELSE
        PRINT "Error-input list length is not legal"
      END IF
END
```

# 프로그래밍 언어(LISP)

## □ LISP라는 이름은 LISt Processing의 줄임말

- 연결 리스트는 리스프의 주요 자료 구조 중 하나로서, 리스프 코드는 그 자체로 하나의 리스트

## □ 함수형 프로그래밍 언어로 독특하게 괄호 사용하는 문법이 특징

## □ 리스프는 본래 실용적인 목적 아래 컴퓨터 프로그램을 활용하여 수학 표기법을 나타내기 위한 목적으로 만들어짐

## □ 함수 호출의 경우, 함수 이름 혹은 연산자가 첫번째로 위치하여 피연산자가 이어 위치하게 됨

- 예) 함수  $f$ 가  $a, b, c$ 라는 세 개의 피연산자를 가진 경우는  $(f\ a\ b\ c)$ 와 같이 표기함

# Functional Programming: LISP

## ❑ LISP Processing language

- Designed at MIT by McCarthy

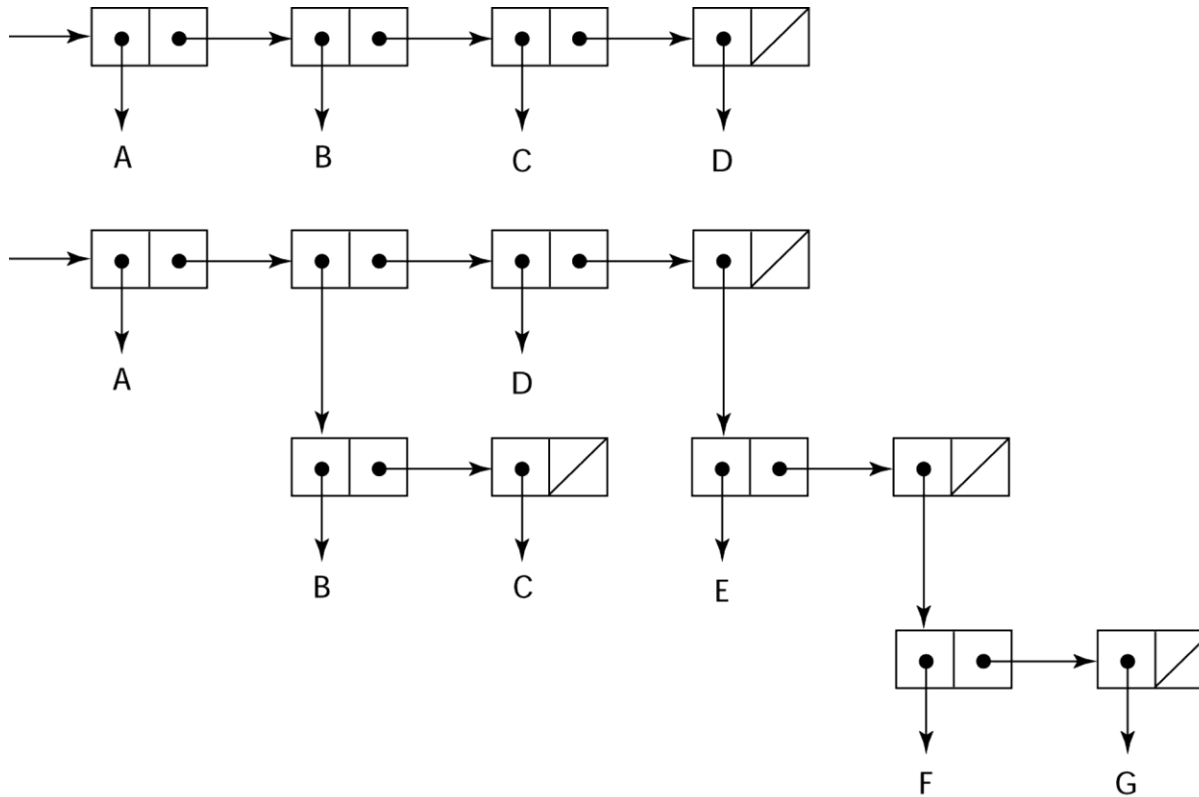
## ❑ AI research needed a language to

- 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*

# Functional Programming: LISP



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

# Functional Programming: LISP

---

## ❑ 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, Miranda, and Haskell are related languages

# Functional Programming: LISP

---

## ❑ 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, Miranda, and Haskell are related languages

# 프로그래밍 언어(LISP)

```
; LISP Example function
; The following code defines a LISP predicate function
; that takes two lists as arguments and returns True
; if the two lists are equal, and NIL (false) otherwise
(DEFUN equal_lists (lis1 lis2)
  (COND
    ((ATOM lis1) (EQ lis1 lis2))
    ((ATOM lis2) NIL)
    ((equal_lists (CAR lis1) (CAR lis2))
     (equal_lists (CDR lis1) (CDR lis2)))
    (T NIL)
  )
)
```

# 프로그래밍 언어(COBOL)

- **COBOL**은 **Common Business Oriented Language**의 약자이며, 영업 및 사무 업무 중심의 언어
- **1950년대** 사무 처리 언어가 개발 업체마다 달라서 문제가 있었음. 이것을 인식한 미국 국방부에서 사무 처리 언어의 통일을 위해 개발한 일반 사무 처리 언어가 “**COBOL**”임
- 코볼보다 먼저 개발된 포트란(**FORTRAN**)은 주로 과학 기술 계산용인 반면 비슷한 시기에 탄생된 **COBOL**은 대량 데이터 처리를 위한 업무처리 및 관리 분야용으로 자리 잡게 됨
- 장점은 호환성이 좋고, 파일관리가 쉽고 영어형식으로 이해하기가 쉽고 프로그램 편집이 쉬움



# Computerizing Business Records: COBOL

---

## ❑ Environment of development

- UNIVAC was beginning to use FLOW-MATIC
- USAF was beginning to use AIMACO
- IBM was developing COMTRAN

# COBOL Historical Background

---

## ❑ Based on FLOW-MATIC

## ❑ FLOW-MATIC features

- Names up to 12 characters, with embedded hyphens
- English names for arithmetic operators (no arithmetic expressions)
- Data and code were completely separate
- The first word in every statement was a verb

# 프로그래밍 언어 예제(COBOL (1))

IDENTIFICATION DIVISION.

PROGRAM-ID. PRODUCE-REORDER-LISTING.

ENVIRONMENT DIVISION.

CONFIGURATION SECTION.

SOURCE-COMPUTER. DEC-VAX.

OBJECT-COMPUTER. DEC-VAX.

INPUT-OUTPUT SECTION.

FILE-CONTROL.

    SELECT BAL-FWD-FILE    ASSIGN TO READER.

    SELECT REORDER-LISTING  ASSIGN TO LOCAL-PRINTER.

# 프로그래밍 언어 예제(COBOL (2))

```
DATA DIVISION.
FILE SECTION.
FD  BAL-FWD-FILE
   LABEL RECORDS ARE STANDARD
   RECORD CONTAINS 80 CHARACTERS.
01  BAL-FWD-CARD.
    02 BAL-ITEM-NO          PICTURE IS 9(5).
    02 BAL-ITEM-DESC        PICTURE IS X(20)
    02 FILLER                PICTURE IS X(5).
    02 BAL-UNIT-PRICE        PICTURE IS 999V9
    02 BAL-REORDER-POINT     PICTURE IS 9(5).
    02 BAL-ON-HAND           PICTURE IS 9(5).
    02 BAL-ON-ORDER          PICTURE IS 9(5).
    02 FILLER                PICTURE IS X(30)
```

```
FD  REORDER-LISTING
   LABEL RECORDS ARE STANDARD
   RECORD CONTAINS 132 CHARACTERS.
01  REORDER-LINE.
    02 RL-ITEM-NO            PICTURE IS Z(5).
    02 FILLER                PICTURE IS X(5).
    02 RL-ITEM-DESC          PICTURE IS X(20).
    02 FILLER                PICTURE IS X(5).
    02 RL-UNIT-PRICE         PICTURE IS ZZZ.99.
    02 FILLER                PICTURE IS X(5).
    02 RL-AVAILABLE-STOCK    PICTURE IS Z(5).
    02 FILLER                PICTURE IS X(5).
    02 RL-REORDER-POINT      PICTURE IS Z(5).
    02 FILLER                PICTURE IS X(71).

WORKING-STORAGE SECTION.
01  SWITCHES.
    02 CARD-EOF-SWITCH       PICTURE IS X.
01  WORK-FIELDS.
    02 AVAILABLE-STOCK        PICTURE IS 9(5).
```

# 프로그래밍 언어 예제(COBOL (3))

```
PROCEDURE DIVISION.
000-PRODUCE-REORDER-LISTING.
    OPEN INPUT BAL-FWD-FILE.
    OPEN OUTPUT REORDER-LISTING.
    MOVE "N" TO CARD-EOF-SWITCH.
    PERFORM 100-PRODUCE-REORDER-LINE
        UNTIL CARD-EOF-SWITCH IS EQUAL TO "Y".
    CLOSE BAL-FWD-FILE.
    CLOSE REORDER-LISTING.
    STOP RUN.

100-PRODUCE-REORDER-LINE.
    PERFORM 110-READ-INVENTORY-RECORD.
    IF CARD-EOF-SWITCH IS NOT EQUAL TO "Y"]
        PERFORM 120-CALCULATE-AVAILABLE-STOCK
        IF AVAILABLE-STOCK IS LESS THAN BAL-REORDER-POINT
            PERFORM 130-PRINT-REORDER-LINE.

110-READ-INVENTORY-RECORD.
    READ BAL-FWD-FILE RECORD
    AT END
        MOVE "Y" TO CARD-EOF-SWITCH.

120-CALCULATE-AVAILABLE-STOCK.
    ADD BAL-ON-HAND BAL-ON-ORDER
    GIVING AVAILABLE-STOCK.

130-PRINT-REORDER-LINE.
    MOVE SPACE                TO REORDER-LINE.
```

# 프로그래밍 언어(PASCAL)

- 파스칼은 1969년에 스위스 ETH 취리히의 컴퓨터 과학자 니콜라우스 비르트가 개발
- 포인터를 사용한 구조적 프로그래밍이 특징
- 알골 60의 영향을 받은 까닭에 같은 시기에 마찬가지로 영향을 받아 제작된 C와 여러가지 면에서 유사한 점이 있음
- 현재는 초기의 파스칼에 비해 많은 부분이 추가, 개선되고 상용 파스칼 컴파일러인 델파이는 C++과 거의 기능 차이가 없음

# 프로그래밍 언어 예제(PASCAL(1))

```
{Pascal Example Program
Input:  An integer, listlen, where listlen is less than
        100, followed by listlen-integer values
Output: The number of input values that are greater than
        the average of all input values }
program pasex (input, output);
  type intlisttype = array [1..99] of integer;
  var
    intlist : intlisttype;
    listlen, counter, sum, average, result : integer;
begin
  result := 0;
  sum := 0;
  readln (listlen);
  if ((listlen > 0) and (listlen < 100)) then
    begin
    { Read input into an array and compute the sum }
    for counter := 1 to listlen do
      begin
        readln (intlist[counter]);
        sum := sum + intlist[counter]
      end;
```

## 프로그래밍 언어 예제(PASCAL(2))

```
{ Compute the average }
  average := sum / listlen;
{ Count the number of input values that are > average }
  for counter := 1 to listlen do
    if (intlist[counter] > average) then
      result := result + 1;
{ Print the result }
  writeln ('The number of values > average is:',
          result)
  end { of the then clause of if (( listlen > 0 ... }
else
  writeln ('Error-input list length is not legal')
end.
```



# 프로그래밍 언어 기타

## □ PL/I

- 과학, 공학 및 산업 응용 프로그램을 위해 설계된 명령형 프로그래밍 언어
- IBM에서 자사의 메인프레임에서 사용하기 위해 개발
- 데이터 처리, 수치 해석, 과학적 연산, 시스템 프로그래밍이며, 언어의 문법은 영어와 같이 취급되며 이들을 확인하고 조작하는 명령어 집합을 사용하여 복잡한 데이터 형식을 기술하는데 적절함

## □ APL

- APL(A Programming Language)은 고급 수학용 프로그래밍 언어
- 케네스 아이버슨에 의해 하버드 대학교에서 발명됨
- 금융 및 보험 애플리케이션, 시뮬레이션, 수학 응용 프로그램 등 다양한 응용에서 사용

## □ SNOBOL

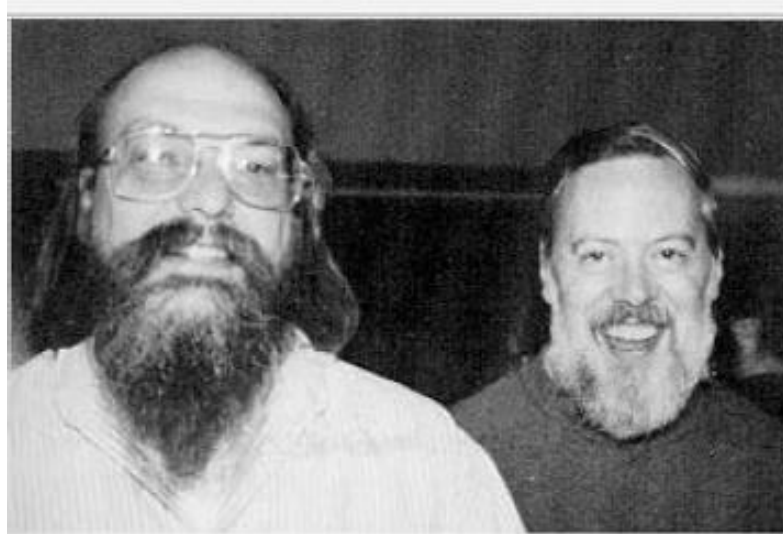
- 개발자가 다항식을 기호적으로 조작하기 위해 사용되는 도구로 제작
- IBM 7090용 어셈블리어로 작성되었으며, 단순한 문법, 오직 하나의 자료형, 문자열, 함수의 부재, 선언의 부재, 그리고 오류 제어가 거의 없음이 특징임

## □ SIMULA 67

- ALGOL60으로부터 블록구와 제어문을 가져옴
- 실행이 중단된 지점에서 실행을 재개하는 것을 허용함(coroutine)
- 이를 지원하기 위해 클래스 구조가 개발됨

# 프로그래밍 언어(c)

- C언어는 1972년 켄 톰슨과 데니스 리치가 벨 연구소에서 일할 당시 새로 개발된 유닉스 운영 체제에서 사용하기 위해 개발한 프로그래밍 언어
- 켄 톰슨은 BCPL 언어를 필요에 맞추어 개조해서 “B”언어(벨 연구소의 B를 따서) 명명했고, 데니스 리치가 이것을 개선하여 C언어가 탄생
- 유닉스 시스템의 프로그램들은 C로 작성되었고, 수많은 운영체제의 커널 또한 C로 만들어졌음



Ken Thompson (L) and Dennis Ritchie (R)

# 프로그래밍 언어 예제(C(1))

/\* C Example Program

Input: An integer, listlen, where listlen is less than 100, followed by listlen-integer values

Output: The number of input values that are greater than the average of all input values \*/

```
int main () {
    int intlist[99], listlen, counter, sum, average, result;
    sum = 0;
    result = 0;
    scanf("%d", &listlen);
    if ((listlen > 0) && (listlen < 100)) {
/* Read input into an array and compute the sum */
        for (counter = 0; counter < listlen; counter++) {
            scanf("%d", &intlist[counter]);
            sum += intlist[counter];
        }
    }
}
```

## 프로그래밍 언어 예제(C(2))

```
/* Compute the average */
    average = sum / listlen;
/* Count the input values that are > average */
    for (counter = 0; counter < listlen; counter++)
        if (intlist[counter] > average) result++;
/* Print result */
    printf("Number of values > average is:%d\n", result);
}
else
    printf("Error-input list length is not legal\n");
}
```

# 프로그래밍 언어(Ada)

- Ada는 구조화되고 정적인 형태를 가지고 명령적이며, 객체 지향적인 고급 컴퓨터 프로그래밍 언어
- 1977년에서 1983년까지 수백개의 프로그램 언어를 대신할 목적으로 고안된 언어로 매우 강력한 유형의 시스템 언어
- Ada는 컴퓨터 프로그래밍을 발명하는데 공헌한 에이다 러브레이스(1815-1852년)의 이름을 딴 것
  - Ada는 찰스 배비지가 고안한 해석 기관에서 처리될 알고리즘이 최초의 프로그램으로 인정되어 ‘세계 최초의 프로그래머’라고 불림

# 프로그래밍 언어 예제(Ada (1))

```
-- Ada Example Program
-- Input:  An integer, List_Len, where List_Len is less
--         than 100, followed by List_Len-integer values
-- Output: The number of input values that are greater
--         than the average of all input values
with Ada.Text_IO, Ada.Integer.Text_IO;
use Ada.Text_IO, Ada.Integer.Text_IO;

procedure Ada_Ex is
    type Int_List_Type is array (1..99) of Integer;
    Int_List : Int_List_Type;
    List_Len, Sum, Average, Result : Integer;
begin
    Result := 0;
    Sum := 0;
    Get (List_Len);
    if (List_Len > 0) and (List_Len < 100) then
-- Read input data into an array and compute the sum
        for Counter := 1 .. List_Len loop
            Get (Int_List(Counter));
            Sum := Sum + Int_List(Counter);
        end loop;
```

## 프로그래밍 언어 예제(Ada (2))

```
-- Compute the average
  Average := Sum / List_Len;
-- Count the number of values that are > average
  for Counter := 1 .. List_Len loop
    if Int_List(Counter) > Average then
      Result:= Result+ 1;
    end if;
  end loop;
-- Print result
  Put ("The number of values > average is:");
  Put (Result);
  New_Line;
else
  Put_Line ("Error-input list length is not legal");
end if;
end Ada Ex;
```

# 프로그래밍 언어(JAVA)

- 자바(JAVA)는 썬마이크로 시스템즈의 제임스 고스링과 다른 연구원들이 개발한 객체 지향적 프로그래밍 언어
- 처음에는 가전제품 내에 탑재해 동작하는 프로그래를 위해 개발했지만, 현재 웹 어플리케이션 개발에 가장 많이 사용하는 언어 가운데 하나이며 모바일 기기요 SW개발에도 많이 사용됨
- 자바 개발자들은 유닉스 기반 배경이 있기 때문에 문법적인 특성은 C언어와 비슷함
- 자바를 다른 컴퓨터 언어와 구분짓는 가장 큰 특징은 컴파일된 코드가 플랫폼 독립적이라는 것임
  - 자바 프로그램을 실행하기 위해서는 JVM(자바 가상 머신)이 필요하다. 이 가상 머신은 어느 플랫폼에서나 동일한 형태로 실행시킴
  - CPU나 운영체제의 종류에 상관없이 JVM을 설치할 수 있는 환경에서 실행이 가능



# 프로그래밍 언어 예제(JAVA(1))

```
// Java Example Program
//  Input: An integer, listlen, where listlen is less
//          than 100, followed by length-integer values
//  Output: The number of input data that are greater than
//          the average of all input values
import java.io.*;
class IntSort {
public static void main(String args[]) throws IOException {
    DataInputStream in = new DataInputStream(System.in);
    int listlen,
        counter,
        sum = 0,
        average,
        result = 0;
    int[] intlist = new int[99];
    listlen = Integer.parseInt(in.readLine());
    if ((listlen > 0) && (listlen < 100)) {
/* Read input into an array and compute the sum */
        for (counter = 0; counter < listlen; counter++) {
            intlist[counter] =
                Integer.valueOf(in.readLine()).intValue();
            sum += intlist[counter];
        }
    }
}
```

## 프로그래밍 언어 예제(JAVA(2))

```
/* Compute the average */
    average = sum / listlen;
/* Count the input values that are > average */
    for (counter = 0; counter < listlen; counter++)
        if (intlist[counter] > average) result++;
/* Print result */
    System.out.println(
        "\nNumber of values > average is:" + result);
} /** end of then clause of if ((listlen > 0) ...
else System.out.println(
    "Error-input list length is not legal\n");
} /** end of method main
} /** end of class IntSort
```

# 프로그래밍 언어(C#)

- C#은 마이크로 소프트에서 개발한 객체 지향 프로그래밍 언어로 닷넷 프레임워크의 한 부분으로 만들었으며 C++와 자바의 문법과 비슷한 문법을 갖고 있음
- C#은 닷넷 프로그램이 동작하는 닷넷 플랫폼을 가장 직접적이고 반영하고, 또한 닷넷 플랫폼에 강하게 의존하는 프로그래밍 언어임
- C#의 기본 자료형은 닷넷의 객체 모델을 따르고 있고, 런타임 차원에서 쓰레기 수집(**garbage collection**)이 되면 또한, 클래스, 인터페이스, 위임, 예외와 같이 객체 지향 언어로서 가져야할 모든 요소들이 포함되어 있음

# 프로그래밍 언어 예제(C#(1))

```
// C# Example Program
// Input:  An integer, listlen, where listlen is less than
//          100, followed by listlen-integer values.
// Output: The number of input values that are greater
//          than the average of all input values.
using System;
public class Ch2example {
    static void Main() {
        int[] intlist;
        int listlen,
            counter,
            sum = 0,
            average,
            result = 0;
        intList = new int[99];
        listlen = Int32.Parse(Console.ReadLine());
        if ((listlen > 0) && (listlen < 100)) {
// Read input into an array and compute the sum
            for (counter = 0; counter < listlen; counter++) {
                intList[counter] =
                    Int32.Parse(Console.ReadLine());
                sum += intList[counter];
            } //- end of for (counter ...
```

## 프로그래밍 언어 예제(C#(2))

```
// Compute the average
    average = sum / listlen;
// Count the input values that are > average
    foreach (int num in intList)
        if (num > average) result++;
// Print result
    Console.WriteLine(
        "Number of values > average is:" + result);
} //- end of if ((listlen ...
else
    Console.WriteLine(
        "Error--input list length is not legal");
} //- end of method Main
} //- end of class Ch2example
```

---

*Thank you for your attention !!*

---

Q and A