© Copyright by 2018 Doo-Soon Park

제 6 장

데이터 타입

©Copyright by 2018 Doo-Soon Park

제 6 장 데이터 타입

❖ 목적 :

- ➢ 데이터 타입의 개념
- ➢ 데이터 타입의 특성
  - ✓기본 데이터 타입(o)
  - ✓문자열 타입(x)
  - ✓사용자-정의 순서 타입(x)
  - ✓배열 타입(o)
  - ✓연상 배열X)
  - ✓레코드 타입(o)
  - ✓공용체 타입(x)
  - ✓집합 타입(x)
  - ✓포인터 타입(x)

©Copyright by 2018 Doo-Soon Park

6.1 서론

- **자료 형(data type) 정의 :**
  - 변수가 가질 수 있는 값의 범위와 그 값들에 대해서 수행 가능한 연산들의 집합
- **한 언어에서 자료 형은?**
  - 1)자료형의 specification(logical organization)
    - ✓ 1)attribute -
    - ✓ 2)value -
    - ✓ 3)operation - data object를 다루기 위한 것(select, change, access)
  - 2)자료형의 implementation(storage structure)
    - ✓ 1)storage representation - 프로그램이 실행되는 동안에 computer storage에 그 자료형의 data object를 표현하기 위한 것

©Copyright by 2018 Doo-Soon Park

6.1 서론

- **자료의 속성을 결정하는데 도움을 주는 것** - 선언문, type checking, type conversion, assignment
- **선언문**
  - **정의** - 실행하는 동안에 사용될 자료의 속성들(data type, 크기, 이름, 생성 시기, 소멸 시기, 참조하기 위한 첨자 등)을 language translator에게 알려주는 문장
    - [예] float a, b;
    - ALGOL integer array a[2:10]
  - **종류** -
    - implicit(default) - FORTRAN변수
    - explicit – 선언하는 변수 int a, b;
  - **static 과 dynamic type checking** –
    - FORTRAN, ALGOL, PL/I, COBOL, PASCAL ⇒ 선언문 허용
    - LISP, APL, SNOBOL 4 ⇒ 선언문 허용하지 않음
    - ALGOL, PL/I ⇒ flexibility때문에 자료에 관한 선언을 완화
      - ALGOL - 배열의 차원수와 TYPE선언
      - 배열의 크기와 첨자 변수의 범위는 선언하지 않아도 됨
      - int a[1:n];
      - PL/I - flexibility와 efficiency를 잘 조화

©Copyright by 2018 Doo-Soon Park

6.1 서론

- **type checking and type conversion**
  - 정의 - 하나의 프로그램에 의해서 실행되어지는 각 operation이 적당한 자료형의 적당한 argument를 받는지를 checking
  - 종류 -
    - ✓ static type checking - program이 번역되는 동안에 checking
      - ♣ 장점 - 실행 시간이 빠르다.
      - ♣ 단점 - flexibility부족, storage 낭비
      - ♣ FORTRAN, ALGOL, PL/I, COBOL, PASCAL
    - ✓ dynamic type checking - run-time에 checking
      - ♣ object의 data type을 지시하는 type tag를 갖고 수행
      - ♣ 장점 - flexibility, storage 낭비 막음, programming을 단순화
      - ♣ 단점 - 실행 시간이 느림, debugging이 어려움, 각각의 data object을 유지시키기 위한 extra storage필요
      - ♣ LISP, APL, SNOBOL 4 , PROLOG
- type conversion and coercion - type checking시 mismatch가 발생시 처리 방안
  - 종류 - explicit type conversion - programmer(cast 연산 –
    - ♣ Int x, y; double z; z=(double)x/y;
  - implicit type conversion(coercion) - system에서 자동적으로 type conversion
  - PASCAL, ADA - 거의 허용 안함
  - C, PL/I, COBOL - 거의 허용

©Copyright by 2018 Doo-Soon Park

6.1 서론

- **assignment statement(배정문, 치환문)-**
  - 정의 - 하나의 data object에 하나의 값을 binding시키기 위한 basic operation
  - 종류 -
    - explicit assignment - A:=B, A=B, MOVE B TO A, A <-- B, (SETQ A B)
    - implicit assignment - SNOBOL 4의 INPUT, PASCAL의 WRITE(file의 buffer변수에 assign)
  - l(location, left-hand side)-value와 r(right hand side)-value -
    - A:=B의 의미
    - 예외 - BLISS는 l-value만 존재, R-value는 unary operator .를 앞에 붙여 표현
    - ALGOL 68 - R-value만 존재, l-value는 변수 선언시 type 앞에 ref를 붙여서 선언 ,
      - ref int
- 배정 연산에 관한 사항 -
  - dereferencing - 참조를 행하여 그와 관련된 값을 반환
    - 일반적으로 묵시적으로 행함
    - BLISS언어에서 l-value만이 존재하므로 dereferencing을 하기 위해서 변수 앞에 .(dot)를 붙여주어야 함.
  - argument가 평가되는 순서 -
    - APL에서 a[b<-- 1] <-- b의 경우 a[1] <-- 1을 넣느냐 혹은 전에 규정된 b값을 배정하느냐

©Copyright by 2018 Doo-Soon Park

6.2 기본 데이터 타입

- **Elementary data type(numeric data type, enumerations, Booleans, characters) structured data type(array, record, character string)**
- numeric data types(정수, 부동소수점, 십진수)
  - 정수 -
    - ✓ specification - ordered subset, maxint, short, int, long
    - ✓ implementation - integer에 대한 storage 표현
  - 부동소수점 -
    - ✓ specification - ordered sequence
    - ✓ implementation – 그림 6.1
      - ♣ 소수와 지수의 두 부분으로 표현 – float(32비트), double(64비트)
  - 십진수 – 사무 시스템 응용 분야 지원
    - ♣ 고정된 개수의 십진수 저장, 소수점은 고정된 위치
    - ♣ BCD 코드(10진수 하나에 4비트 할당)
- 논리형 -
  - specification : true나 false값을 갖는 data object로 구성
  - physical organization : single bit
- 문자 – single character를 갖는 data object로 수치 데이터로 저장
  - ASCII 코드

©Copyright by 2018 Doo-Soon Park

6.5 배열 타입

- **Structured data type - 다른 data object들의 aggregate로 vector, array, record, character string, list, stack, queue, tree**
- 설계 고려 사항
  - 첨자의 올바른 타입
  - 첨자 범위 검사 유무
  - 첨자 범위의 바인딩 시기
  - 배열 할당 시기
  - 허용되는 첨자의 개수
  - 배열의 초기화
  - 슬라이스 (slice)연산 제공
- **logical organization :**
  - 1)component의 수 - fixed size - 실행시간 고정(array, records)
  - variable size -실행 시간에 원소의 개수가 변하는
  - 것(문자열, stack, queue, list, set, file)
  - 2)각 component의 type – homogeneous(array)
  - heterogeneous(record)
  - 3)component를 selection하기 위해 사용되는 이름 - 첨자 변수, field name(quantifier, qualifier)
  - 4)component의 최대수 - 상하한값, 원소의 위치는 첨자에 의해서 판별, FORTRAN 3차원까지 허용, 하한값 1(대부분의 언어는 제한없음)

©Copyright by 2018 Doo-Soon Park

6.5 배열 타입

- operation :
  - ✓ 1)component의 selection - random selection, sequential selection
  - ✓ 2)whole-data structure operation – c 허용하지 않음, ada는 허용
  - ✓ 3)component의 insertion/deletion

❖ storage 의 표현방법 -

- sequential : descriptor와 component를 storage의 single storage block에 저장, fixed-size data structure 사용
- linked list : link라는 pointer를 이용하여 descriptor와 component를 storage의 여러 block에 저장, variable-size structure에 사용

❖ storage management -

- 1)access path - life time의 출발에 있어서 그 data object를 찾아갈 수 있는 access path가 생김
  - ✓ access 할 수 있는 방법 - name, point
- 2)storage management의 두가지 문제점 -
  - ✓ garbage - 하나의 data object에 access하는 모든 path가 없어졌지만 그 data object이 계속해서 존재하는 경우
  - ✓ dangling reference - 한 data object 의 life time이 끝났는데도 access path가 계속해서 존재하는 경우, 즉 pointer변수가 더 이상 의미 없는 것을 가리키고 있는 현상

©Copyright by 2018 Doo-Soon Park

6.5 배열 타입

- Vectors and arrays - vector와 array 의 구분
  - 1)정의 - data type이 homogeneous array이고 그의 component가 고정된 크기로 구성된 data structure
  - 2)logical organization :
    - 1)data type(1차원 array 또는 vector)
    - 2)각 component의 type(integer, real등)
    - 3)각 component의 개수(fixed size)
    - 4)component를 selection하기 위해서 사용되는 이름(첨자)
    - 5)component의 최대수(첨자의 상하한 값)
    - ☞첨자 값으로 상수만 허용할 것인지, 수식까지 허용할 것인지 (FORTRAN, PASCAL은 상수로 국한, ALGOL 60은 정수형 수식 허용
    - ☞실행시간 동안에 배열의 크기 변하는 것(가변 배열) - ADA, PL/I, ALGOL 60, SIMULA)
  - 배역의 특정 원소는 배열 이름과 첨자(인덱스)로 구성된 구성된 선택자에 의해서 참조, 즉 배열 이름(인덱스) -> 원소
  - 배열 첨자 표현 구문
    - 괄호:
      - Fortran, PL/1, Ada

        SUM = SUM + B(I)
    - 대괄호
      - Pascal, C, C++, Java

©Copyright by 2018 Doo-Soon Park

6.5 배열 타입

❖ 배열 연산

➢ 배열 단위로 이루어지는 연산

언어의 예

✓ APL: A + B

A × B

✓ Ada: 배열 배정, 일차원 배열에 대한 접합 연산

✓ FORTRAN 90: 배정, 산술, 논리 연산자 제공, 행렬 곱셈, 행렬 전치, 벡터 내적에 대한 라이브러리 함수 제공

✓ C-기반 언어들은 배열 연산 제공하지 않음

©Copyright by 2018 Doo-Soon Park

6.5 배열 타입

❖ 슬라이스

- ➢ 배열의 슬라이스는 배열의 부분 구조이다.

언어의 예

- ✓ FORTRAN 90: INTEGER MAT (1:4, 1:4)
  MAT(1 : 4, 1)
  MAT(2, 1 : 4)
- ✓ Ada: 단지 일차원 배열에 대해서만,
  LIST(4 .. 10)

©Copyright by 2018 Doo-Soon Park

6.5 배열 타입

❖ 배열의 구현

- physical organization : 어떻게 구현하느냐가 중요(sequential block 혹은linked representation)
  - ✓ row-major order : C, PASCAL, PL/I , Cobol
  - ✓ column-major order : FORTRAN
  - ✓ edge-vector에 의한 방법(slice방법) - 각 array의 substructure가 array가 되는 방법

©Copyright by 2018 Doo-Soon Park

6.5 배열 타입

❖ 배열의 구현

- 배열 원소의 접근에 대한 코드를 컴파일러가 생성

  주소(list[k]) = 주소(list[하한]) + (k-하한) * 원소 크기

- 서술자(descriptor)
  - ✓ 접근 함수 구성, 인덱스 범위 검사에 필요한 정보 포함
  - ✓ 배열, 원소 형, 인덱스 형, 인덱스 하한, 인덱스 상한, 주소
  - ✓ 컴파일 시간/실행시간 서술자

배열
원소 타입
인덱스 타입
인덱스 하한
인덱스 상한
주소

일차원 배열의 컴파일 시간 서술자

©Copyright by 2018 Doo-Soon Park

6.5 배열 타입

❖ 배열의 구현

➢ 다차원 배열의 저장

- ✓ 열-우선 순서(column major order): 3 6 1 5 2 3 7 5 8
- ✓ 행-우선 순서(row major order): 3 5 7 6 2 5 1 3 8

3	5	7
6	2	5
1	3	8

➢ 다차원 배열 주소:

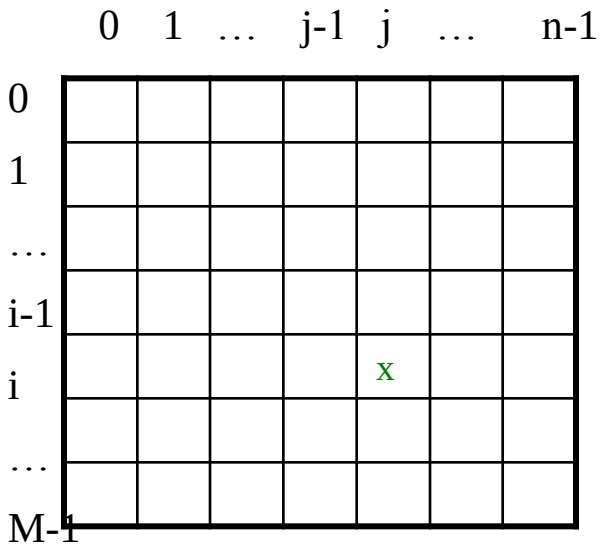
a[i, j] = 주소(a[0, 0]) +

((((i번째 행보다 위에 위치한 행의 개수)*

(행의 크기)) +

(j번째 열보다 왼쪽에 위치한 원소의 개수))*

원소_크기)



©Copyright by 2018 Doo-Soon Park

6.5 배열 타입

❖ 배열의 구현

- ➢ 다차원 배열 서술자:

다차원배열
원소 형
인덱스 형
차원수
인덱스1 상한
인덱스1 상한
…
주소

- ➢ 배열 저장 순서를 아는 것이 왜 중요한가?

©Copyright by 2016

6.5 배열 타입

배열

- 동질(homogeneous)의 자료 객체들을 묶고 여기에 그룹 이름을 부여하는 배열
- 프로그램을 실행하는 동안 특정한 원소를 상대 주소로 직접 접근할 수 있도록 허용
- C 언어에서 배열이 다음과 같이 선언될 때, 배열 score는 정수형 자료 100개를 메모리에 연속적으로 기억시킬 수 있는 기억 공간을 필요로 한다.
  - ✓ int score [100];
- 배열의 요소들은 연속적으로 저장되면 빠르게 접근할 수 있어 실행 시간이 단축된다. 다음과 같이 가정하고 위치를 계산해보자.
  - ✓ 각 배열 요소의 크기를 w라고 하면 배열 A의 i번째 요소는 다음 위치에서 시작한다.
  - ✓ base + (i - low + 1) × w … (9.1)
  - ✓ 여기서 low는 배열 첨자의 하한 값이며 1이라고 하자(C 언어의 경우 하한 값이 0이므로 주소 계산이 달라진다). base는 배열에 할당된 메모리의 상대 주소이다. 즉 base는 A[low]의 상대 주소이다. 또한 w는 각 요소의 크기이다. 식 (9.1)을 식 (9.2)로 변환해보자.
  - ✓ (i + 1) × w + (base - low × w) … (9.2)
  - ✓ 식 (9.2)는 컴파일 시간에 부분적으로 실행될 수 있다. 부분식 d = base - low × w는 배열의 선언이 나타나면 바로 계산될 수 있다. 즉 배열 A에 대한 기호표 항목 안에 d의 값이 계산 및 저장되면 A[i]의 상대 주소는 단순히 (i + 1) × w를 d에 더함으로써 계산된다.

©Copyright by 2016

6.5 배열 타입

- ✓같은 방법으로 다차원 배열을 다음과 같이 선언한다.
- ✓ int board[2][3];
- ✓board는 각각 원소가 연속된 메모리 영역을 할당하지만 board[1][2]가 자료 영역에서 몇 번째 셀에 해당되는지 쉽게 알 수 없다. 그러나 논리적인 배열의 구조는 메모리에서 1차원 배열로 정돈되기 때문에 손쉽게 상대적인 위치를 계산해낼 수 있는 반면에 다차원 배열은 언어에 따라 하한 값과 상한 값이 달라질 수 있다. C 언어에서는 하한 값으로 0이 고정되어 있으나 파스칼에서는 하한 값이 음수 또는 임의의 정수로 서술될 수 있다. 또한 포트란 언어는 하한 값이 1로 고정되어 있다.
- ✓2차원 배열을 메모리에 저장하기 위해 1차원 배열로 변환하는 방법은 행 우선(row-major), 열 우선(column-major), 슬라이스(slice) 등 여러 가지 방법이 있다. [그림 9-3]은 행 우선 및 열 우선 방법으로 저장된 2×3 배열 A의 내용을 보여준다. 슬라이스는 배열에서 어떤 부분의 구조이다.

©Copyright by 2016

6.5 배열 타입

그림 9-3 2차원 배열의 메모리 저장 형태

- 다차원 배열을 1차원 배열로 저장하는 방법
  - ✓ 언어마다 사용하는 방법이 다르므로 이런 특성을 반영한다면 매우 효율적인 프로그래밍이 가능할 것이다. C, 파스칼, PL/I 등은 행 우선 방법을 사용하고, 포트란은 열 우선 방법을 사용한다.
  - ✓ 행 우선 방법으로 저장된 2차원 배열의 경우 A[i, j]의 상대 주소는 식 (9.3)과 같다.
  - ✓ base + (((i - low1) × n2) + (j - low2 + 1)) × w ... (9.3)

©Copyright by 2016

6.5 배열 타입

- ✓여기서 low1과 low2는 i와 j의 하한 값이고, n2는 열의 크기로 j가 취할 수 있는 값의 크기이다. 즉 high2를 j에 대한 상한 값이라고 하면 n = high2 - low2 + 1 이다. A[i, j]의 상대 주 소를 계산하는데 컴파일 시간에 i와 j의 값만 모르고 다른 값들을 모두 안다면 식 (9.3)은 식 (9.4)와 같이 다시 쓸 수 있다.
- ✓ ((i × n) + (j + 1)) × w + (base - ((low1 × n) + low2) × w) ... (9.4)
- ✓식 (9.4)의 뒷부분은 컴파일 시간에 계산할 수 있다. 같은 방법으로 열 우선에 대해서도 2차 원 배열의 경우 A[i, j]의 상대 주소는 식 (9.5)와 같다.
- ✓ base + (((j – low2) × n1) + (i – low1 + 1)) × w ... (9.5)
- ✓n1은 행 단위의 원소 개수로 i가 취할 수 있는 값의 크기이다.

©Copyright by 2016

6.5 배열 타입

[예제 9-1] 행우선과 열우선에 따른 주소 계산하기

- 다음과 같이 선언된 2차원 행렬에 대해 하한 값을 1로 하여 행 우선과 열 우선으로 메모리에 저장하고, 행 우선과 열 우선에 대해 B(3,2)의 주소를 계산해보자.
- int B(4,4);
- **[풀이]**

✓행 우선 방법으로 메모리에 저장하면 다음과 같다.

✓열 우선 방법으로 메모리에 저장하면 다음과 같다.

©Copyright by 2016

6.5 배열 타입

- 원소 B(3,2)에 대한 상대 주소를 계산한다. 그림의 화살표를 보면 행 우선 방법은 열 번째이고 열 우선 방법은 일곱 번째임을 알 수 있다. 또한 각 원소가 정수형이므로 4바이트를 할당한다면 주소는 base로부터 행 우선의 경우 40, 열 우선의 경우 28이어야 한다.
- 식 (9.3)에 의하면 i는 3, j는 2, low1은 하한 값이므로 1, low2도 1, n2는 열의 크기이므로 4, w는 4이므로 다음과 같다.
- base + (((i - low1) × n2) + (j - low2 + 1)) × w
- = base + (((3 - 1) × 4) + (2 - 1 + 1)) × 4
- = base + (8 + 2) × 4
- = base + 40
- 같은 방법으로 열 우선으로 계산하기 위해 식 (9.5)를 사용한다. 여기서 i는 3, j는 2, low1은 하한 값이므로 1, low2도 1, n1은 행의 크기이므로 4, w는 4이므로 다음과 같다.
- base + (((j - low1) × n1) + (i - low2 + 1)) × w
- = base + (((2 - 1) × 4) + (3 - 1 + 1)) × 4
- = base + (4 + 3) × 4
- = base + 28

©Copyright by 2018 Doo-Soon Park

6.7 레코드 형

- 레코드 – 자료형이 heterogeneous array이고 그의 component(=field)가 identifier인 data structure , Pascal, PL/I (record를 structure라고 부름), COBOL, C, C++, C#의 구조체
  - 1)logical organization :
    - 1)data type(record)
    - 2)각 component의 type(heterogeneous)
    - 3)각 component의 개수(fixed size)
    - 4)component를 selection하기 위해서 사용되는 이름(field name)
  - 2)operation :
    - to select a component of the record - qualifier사용
  - 3)physical organization : sequential block
    - location R.I = C+ (size of R.J) C :base address R.J : j-th component
  - 4)variant record : record의 길이가 변하는 record, 가장 큰 크기를 잡음, Pascal CASE, COBOL condition name(level number 88)
  - 5)선언문 없는 언어에서의 record : fixed size, 각각의 type을 선언하지 않고 execution하는 동안에 치환문을 이용해서 dynamically하게 변화시킴, SNOBOL 4, LISP
    - 예) A=ARRAY(10)
    - A<1>='XYZ'
    - A<2>=27(type도 바뀜)

©Copyright by 2018 Doo-Soon Park

6.7 레코드 형

- 레코드 정의와 선언 예 : C

```
struct student {
   char name[20];
   int number;
   char address[30];
};

struct student A;
```

- 레코드 정의와 선언 예 : Ada

```
type student is record
   name: string (1..20);
   number: integer;
   address: string (1..30);
end record;

A: student;
```

© Copyright by 2018 Doo-Soon Park

제 9 장

부프로그램

©Copyright by 2018 Doo-Soon Park

제 9 장 부프로그램

❖ 목적 :

- ➢ 부프로그램의 원리
- ➢ 부프로그램의 설계시 고려사항
- ➢ 매개변수전달 방법

©Copyright by 2018 Doo-Soon Park

9.2 부프로그램의 원리

- 일반적 특성
  - 단일의 진입점
  - 피호출자 실행시에 호출자 실행 중단
  - 피호출자 종료시에 제어가 호출자에 반환
- 매개변수
  - 형식 매개변수(formal parameters)
  - 실 매개변수(actual parameters)

©Copyright by 2018 Doo-Soon Park

9.3 부프로그램의 설계고려사항

- 지역 변수는 정적 또는 동적으로 할당되는가?
- 어떤 매개변수 전달 방법이 사용되는가?
- 실 매개변수의 형이 형식매개변수의 형에 대한 검사를 하는가?
- 부프로그램이 매개변수로 전달될 경우, 그 참조 환경은?
- 부프로그램의 중복(overloaded) 가능성 –
  - 연산자 중복 처럼 동일 참조 환경 내에서 다른 부프로그램과 같은 이름을 갖는 부프로그램
- 부프로그램의 일반적(generic) 가능성
  - 다른 호출에는 다른 자료 형의 자료에 계산을 수행하는 부프로그램.
- 개별 혹은 독립 컴파일 가능성

©Copyright by 2018 Doo-Soon Park

부프로그램 순서 제어 구조(추가)

- 부프로그램(subprogram) sequence control
  - simple subprogram call-return - copy rule로 설명
  - copy rule에서 반드시 요구하는 항목 -
    - ✓ subprogram들은 recursion을 허용하지 않는다 ➔ 프로그램이 한없이 커진다.
      - ♣ recursion의 종류 - direct와 indirect recursion
    - ✓ explicit CALL문장을 요구한다 ➔ subprogram은 CALL하는 위치를 translator에게 분명하게 명시해야 한다.
      - ♣ exception handling - 예외적인 처리가 가능하도록
      - ♣ - PL/I의 ON조건, ADA의 raise문
    - ✓ call된 subprogram은 반드시 완전히 수행된 후에 return되어야 한다
      - ♣ coroutine - subprogram의 수행이 완전히 끝나기 전에 call하는 프로그램으로 control을 resume시키는 subprogram을 coroutine
    - ✓ call되는 시점에서 즉시 문장의 control이 이전된다
      - ♣ scheduled subprogram - CALL B AFTER A,
      - ♣ CALL B WHEN X=5 AND Z > 0
    - ✓ single execution sequence – 폰 노이만형의 단일 프로세서
      - ♣ nonsequential execution - concurrent or parallel program,
      - ♣ 병렬처리(parallel processing) : 다수의 프로세서들이 여러 개의 프로그램들 또는 한 프로그램의 분할된 부분들을 동시에 처리하는 기술로서 하드웨어와 관련된 용어(반드시 요구하는 조건 - 순차 처리와 똑같은 결과를 발생)
      - ♣ 병행 처리(concurrent processing) :병렬 처리와 같은 개념으로 실제기계에서는 어떻게 수행되든지 관계없이 언어적 표현에서 동시에 실행하는 것을 의미

©Copyright by 2018 Doo-Soon Park

9.5 매개변수-전달 방법

- 매개변수 전달(**parameter passing**) : 매개변수 사이에 값을 주고 받는 것
  - **형식 매개변수(formal parameter)** - 하나의 부 프로그램 안에 있는 local data object로서 부 프로그램이 실행되는 동안에 식 또는 다른 이름을 대신하여 그 프로그램 안에서 사용되는 변수
  - **실 매개변수(actual parameter)** – 형식 매개 변수에 대응되는 data object
- **매개 변수 전달 방법 :**
  - 1) 참조 호출(**call by reference, address, location**) – 실 매개변수에 해당되는 주소를 대응되는 형식 매개변수에 보내는 방법.
    - ✓ 대표적으로 포트란, PL/I과 C 언어의 포인터 변수
    - ✓ 이 방법은 2개 이상의 단위 프로그램에서 2개 이상의 변수가 기억 장소를 공유함으로써 지역 변수 이외의 변수 값을 변화시키는 **부작용**(side effect)이 발생, 이는 프로그램을 읽고 이해하는 데 어려움을 줌. 부작용을 일으키는 대표적인 예로 포트란의 COMMON 문장.
    - ✓ 하나의 단위 프로그램에서 2개 이상의 변수가 기억 장소를 공유할 때 발생하는 **별칭**(alias). 별칭의 대표적인 예는 포트란의 EQUIVALENCE 문장이다.

©Copyright by 2018 Doo-Soon Park

9.5 매개변수-전달 방법

- 2) 값 호출(**call by value**) – 실 매개변수와는 별도로 형식 매개변수에 대한 기억 장소를 별도로 할당하는 방법
  - ✓ 부작용이 발생하지 않지만 기억 장소가 추가로 필요하다는 단점. 블록 구조 언어인 C나 파스칼에서 기본적으로 값 호출 방법을 사용.
- 3) 이름 호출(**call by name**) – 형식 매개변수의 이름이 사용될 때마다 그에 대응하는 실 매개변수 자체가 사용된 것처럼 매번 다시 계산하여 시행하는 방법
  - ✓ 초기 프로그래밍 언어인 알골 60에서 사용.
  - ✓ 구현이 어려움,
    - ♣ Interpreter 사용해서 구현 - 나올 때 마다 복사
    - ♣ 컴파일러 사용해서 구현 – 실 매개변수의 주소를 계산할 수 있는 thunk 사용 , ALGOL60
- 4) 결과 호출(**call by result**) – called된 부 프로그램에서 시행된 값을 받는 방법으로 call by value와 같이 하다가 끝에 형식 매개변수의 값을 실 매개변수에 복사하는 방법
- 5) 값-결과-전달(pass-by-value-result) - 값-호출과 결과-호출의 혼합형
  - ✓ 실 매개변수는 형식 매개변수를 초기화하고, 형식 매개변수는 지역 변수로 사용되고, 부프로그램 종료시 형식 매개변수의 값은 실 매개변수로 반환된다.
  - ✓ 단점:
    - ♣ 값-전달의 단점
    - ♣ 결과-전달의 단점

©Copyright by 2018 Doo-Soon Park

9.5 매개변수 전달 방법

- 예]
  - 1) PL/I형태의 program

```
P : PROCEDURE;
    DECLARE A(3),I;
    I=1; A(1)=2;A(2)=4;
    CALL Q(A(I));
Q : PROCEDURE(B)
    A(1)=3; I=2; PUT LIST(B);
    END Q;
END P;
```

  - 1) call by reference - 3
  - 2) call by value - 2
  - 3) call by name - 4

©Copyright by 2018 Doo-Soon Park

9.5 매개변수 전달 방법

- 2)ALGOL형태의 program

```
BEGIN INTEGER A,B;
        PROCEDURE F(X,Y,Z); INTEGER X,Y,Z;
          BEGIN Y := Y+2;
                Z:=Z+X
          END F;
      A:=3;B:=5;F(A+B,A,A);
      PRINT A
    END
```

©Copyright by 2016

9.5 매개변수 전달 방법

[풀이]

① 참조 호출의 경우 :

처음에 A = 3, B = 5이다. 프로시저 F가 호출되면 F(A+B, A, A)에서 addr(A+B) = addr(X), addr(A) = addr(Y), addr(A) = addr(Z)이다. Y := Y + 2에서 Y = 3 + 2 = 5, Z := Z + X에서 Z = 5 + 8 = 13 PRINT A에서 A의 주소는 Z의 주소와 같으므로 13을 출력한다.

② 값 호출의 경우 :

처음에 A = 3, B = 5이다. 프로시저 F가 호출되면 F(A+B, A, A)에서 X, Y, Z에 대한 기억 장 소를 별도로 할당한 다음 X = 8, Y = 3, Z = 3을 치환한다. Y := Y + 2에서 Y = 3 + 2 = 5, Z := Z + X에서 Z = 3 + 8 = 11 그런데 프로시저가 리턴되면서 X, Y, Z에 대한 기억 장소가 삭제된다. PRINT A에서 3을 출력한다.

③ 이름 호출의 경우 :

처음에 A = 3, B = 5이다. 프로시저 F가 호출되면 F(A+B, A, A)가 호출된 다음 Y := Y + 2에서 A = A + 2 = 3 + 2 = 5, Z := Z + X에서 A = A + (A + B) = 5 + 5 + 5 = 15 PRINT A에서 15를 출력한다.

©Copyright by 2018 Doo-Soon Park

9.5 매개변수 전달 방법

- 3) call by value와 call by value-result 차이점

```
PROCEDURE PRINT
  INTEGER Y
  PROCEDURE INCR(X)
    INTEGER X
      BEGIN
        Y <-- 1
        X <-- X+2
      END
  BEGIN
    Y <-- 0
    INCR(Y)
    WRITE Y
  END
```

- 1) call by reference - 3 2) call by value - 1
- 3) call by value-result - 2 4) call by name - 3

©Copyright by 2018 Doo-Soon Park

9.5 매개변수 전달방법

언어별 parameter passing 방법 :

call by ref	call by value	call by value-result	call by name
FORTRAN	PASCAL	ALGOL68	ALGOL 60
PASCAL(var사용)	C	ADA	
C(point)	LISP		
PL/I	SNOBOL		
SNOBOL	APL		
(간접조회연산자 $)			