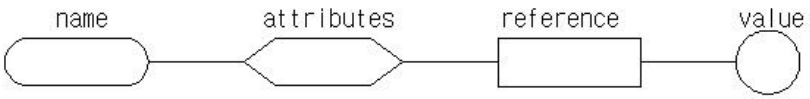
Chap. 5 변수, 바인딩, 식 및 제어문

5.1 Variable

cf) X = 3.14159;

name	① the name of storage location : X
attribute	② the name of description of its current contents : π, 3.14159
reference	③ the storage location holds the value.
value	④ the contents of storage location : 3.14159

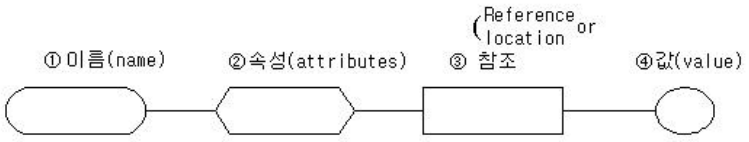
[Def] Variable은 name, attributes, reference, value등의 4요소로 구성된다.



5.2 속성과 바인딩 (Attributes & Binding)

5.2.1 바인딩 개념

* Variable(변수)



① const n=5	n	const	5
② var x:integer; x=2	x	< integer var	2

*Binding : 이름에 속성을 연결하는 과정

5.2.2 바인딩 시간의 종류

*Binding Time : Binding이 이루어지는 시점

{ Translation 時 (Static binding)
Execution 時 (run time, dynamic binding)

P. 105

① Execution Time(Run time, Dynamic binding)

- variable 의 value 확정

- var의 data structure 에 storage 할당

i) module 의 시작시간 : formal parameter $\overset{\text{binding}}{\longleftrightarrow}$ actual para.
local var. 에 storage 할당

ii) name 의 사용시점 : assignment statement : var $\overset{\text{binding}}{\longleftrightarrow}$ value
var 의 기억장소 할당

② Translation Time (Static Binding)
- Compile time
- Link time
- Load time

Data type
Data Structure size & Type) 확정
Record item 들의 type)

③ PL Definition Time
: 혼합형 연산의 type 정의

int i;	i=1;	k=i+r;
real r;	r=2.3;	k의 type에 따라 연산 type결정

④ PL Implementation Time
: 실제 컴퓨터에서 구현할 때에 그 특성의 일부를 확정
- 정수 자릿수
- 실수 유효숫자 개수
- 기계내에서 수의 표기법

eg.1) P106 (예) 5.1

```
y = x + 10;
```

eg.2) X = 1.4;
① the name of storage location : X
② the name or description of its current contents: 1.4
③ the storage location holds the value
④ the contents of storage location : 1.4

eg.3) X = X + 1;
계산순서 : ① get the value of X referencing X
dereferencing X
(accessing X)

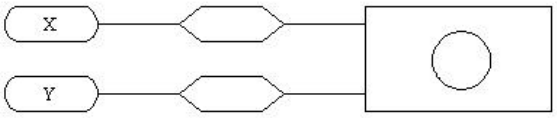
reference and returning its
associated value

② evaluate X + 1

③ store the result of ② into X

r-value	l-value
(right value)	(left address)
*value component	*reference

eg.4) aliasing



X is an alias of Y.
Y is an alias of X.

PP108-109

5.2.3 Binding 시간의 중요성

5.3 Declarations (선언문)

*Declarations : 실행 시 사용될 자료의 속성을 언어의 번역기에게 알려주는 프로그램 문장 (Data Attributes)

cf) data attribute : data
- type
- size
- name
- construction time
- destruction time
- index

eg) int X[10]

→ ① array X의 construction/destruction time (block 범위)
② data type (1-dimensional array)
③ size : 10
④ index range : 0-9
⑤ data element type : integer
⑥ array name : X

* Declaration 文 목적

① 보다 효율적인 주기억장치 사용과 접근 방법이 가능
: 주기억장치안에서 표기와 data structure에 접근하기 위한 계산을 최적화

할 수 있기 때문에 (data type, size를 알수 있음)

② 보다 효율적인 주기억장치 경영이 가능

: data size, creation/destruction time을 알 수 있으므로 효율적인 기억장소 할당 기법 이용

③ static type checking이 가능 p112

- type specific operation real X;
- generic / mixed operation integer Y;
- operator overloading X + Y = ?

cf)

static type checking	dynamic type checking
efficiency	flexibility

cf) default declaration

: explicit declaration이 없으면 implict declaration 로 간주

eg) FORTRAN에서

변수 I~N 으로 시작하면 integer type

그 외 real type

5.4 배정문 (Assignment Statement) : 변수의 내용을 변경

5.4.1 l-value 와 r-value

→ P114 <표 5.1>

Algol, Pascal	A : = B
APL	A <- B
Basic	LET A = B
Cobol	MOVE B TO A
C, JAVA, Fortran	A = B

* A = B

left right cf) name-attributes-location-val.

l-value r-value

(reference) (value)

→ P115 [예 5.2] reading

5.4.2 Assignment Statement의 구현

A = e

variable expression

case : A와 e의 data type이 다를 경우

해결방안

① e 의 값을 A의 속성과 동일하도록 형변환 시켜 A의 새로운 값으로 할당	② A의 속성을 e의 계산값과 동일한 형으로 변환시켜 A의 새로운 값으로 배정
o static binding (declaration)	o run time 에 data type이 변할 수 있어야 함(dynamic binding)
o compiler language	o interpreter 언어
o 효율적인 기억장소 경영	o flexibility
static type checking fast	

구현 방법 i) e 값을 register R에 저장

ii) e 값 type이 A의 속성과 다른 정수 e의 r-value를 A에 맡게 형변환

iii) A의 l-value 계산

iv) R의 내용을 A의 l-value에 저장

```
a = b = c
sum, total = 0

flag ? count1 : count2 = 0

if(flag) count1 = 0
else count2 = 0

sum = ++count

while((ch = getchar()) != EOF) { ... }
```

5.5 상수(Constants)와 초기화

: 값이 변경되지 않는 변수의 사용

공통적으로 기억하기 쉬운 이름을 부여하여 사용

eg) Pi = 3.1415926

* Constant : Identifier로 주어지며 프로그램 수행 중 결합된 값이 결코 변하지 않는다.

In pascal, const

Pi = 3.1415926

cf) 설계시 문제점 pp121-122 ①~④

In Ada

X : Constant INTEGER := 17 / 값이 17인 정수형 상수
Y : INTEGER := 17 / 초기값이 17인 정수형 변수

5.6 수식(Expressions)

* expression 목적 : 계산된 값을 기술

구성 { constants, variables, operators, function calls }] operands

cf) program < state space, environment

: program의 모든 변수들에 결합된 값들이 state space 형성

cf) referential transparency :
Expression의 계산 결과는 오직 값만을 생성하여야 하며,
state space를 변경시켜서는 안됨

cf) operator hierarchy P127. (표 5.3)
arithmetic operators
relational operators
logical operators

cf) logical operators
"x and y" : x가 거짓이면 y는 evaluate 되지 않음
"x or y" : x가 참이면 y는 evalueate 되지 않음

5.7 Conditional Statements

o if-the-else
o case
o 설계시 문제점 P.133 ①~⑤

5.8 Iterative Statements

o
```
loop
  ...
  exit
  ...
repeat
```

o
```
while <bcond> do
  <statements>
end
```
← 검사 후 수행

o
```
repeat
  <statements>
until <bcond>
```
← 수행후 검사

o for <control var> := <initial val> step <increment>
until <final-val>

do <statements>

* 설계시 문제점 P.138 ①~⑧

ex) for i := 1 step 2 until 10
do <statements>

5.9 GoTo 文

장점 : ① Computer에서 간결한 형태의 GoTo 文 제어구조는 곧바로 H/W로 제공됨
② GoTo _ LABEL 구조가 범용성 가짐
: 모든 control form을 표현

단점 : ① Hierarchical program structure의 미비
: 이해, debug 이 어려움
② 문장 순서가 실행 순서와 다름