บทที่ 2 ข้อมูลต่าง ๆ (Data elements)

ข้อมูล (data) โดยทั่ว ๆ ไปหมายถึง มูลค่า หรือข้อเท็จจริงต่าง ๆ แต่ในภาษา PL/1 หมายถึง data item ซึ่งแบ่งออกเป็นสองชนิดคือ arithmetic ซึ่งอาจจะเป็นตัวแปรหรือมูลค่าคงที่ก็ได้ และ string ซึ่งอาจจะเป็นตัวแปรหรือมูลค่าคงที่ได้เช่นกัน

ตัวแปร (Variable) เป็นชื่อที่โปรแกรมเมอร์ตั้งขึ้นมา และมีมูลค่าจำนวนหนึ่ง ซึ่งอาจจะเปลี่ยนแปลงได้ระหว่างที่มีการ execute ภายใน object program

มูลค่าคงที่ (Constant) อาจจะเป็นชื่อก็ได้ เป็นมูลค่าจำนวนหนึ่ง ซึ่งจะไม่มีการเปลี่ยนแปลง

ตัวอย่าง

```
AREA = RADIUS**2 * 3.1416;
```

ในที่นี้

AREA, RADIUS เป็นตัวแปร

2, 3.1416 เป็นมูลค่าคงที่

มูลค่าของ RADIUS เรียกว่า data item และผลลัพธ์จากการคำนวณ expression ทางขวามือของเครื่องหมายเท่ากับ ก็จะเป็น data item ด้วยซึ่งกำหนดให้เป็นมูลค่าของ AREA ส่วนมูลค่าคงที่ 3.1416 ภายในคำสั่งข้างต้นถือว่าเป็น data item เช่นเดียวกัน

ลักษณะเฉพาะของตัวแปร ก็คือ เราจะไม่ทราบทันทีจากชื่อที่ตั้งฉะนั้นลักษณะเหล่านี้จึงเรียกว่า attribute และจะทราบได้จาก keywords ซึ่งจะกำหนดเอาไว้ ในคำสั่ง DECLARE ส่วนลักษณะเฉพาะของมูลค่าคงที่ทราบได้จากตัวเลขที่ปรากฏนั้น

ชนิดของข้อมูล (Data types)

ภาษา PL/1 แบ่งข้อมูลออกเป็น 2 ชนิดคือ Problem data และ Program control data

2.1 Problem Data

มี 2 ชนิดคือ arithmetic data และ string data

2.1.1 Arithmetic data หมายถึง item ที่มีมูลค่าเป็นตัวเลขมีลักษณะเฉพาะของ base, scale, precision และ mode ซึ่งลักษณะเหล่านี้กำหนดโดย attribute ในคำสั่ง DECLARE หรือ กำหนดโดยคำสั่ง DEFAULT

base	ได้แก่เป็นเลขฐานสิบ (decimal) หรือ เลขฐานสอง (binary)
scale	ได้แก่เลข fixed point หรือ floating - point
precision	ได้แก่จำนวนตัวเลขที่ประกอบขึ้นเป็น data item นั้น และจำนวนเลขที่อยู่หลังจุดทศนิยม
mode	ได้แก่การบอกว่าเป็นเลข real หรือเลข complex การกำหนด base, scale และ mode ของ arithmetic variables ให้ใช้ key-words ส่วน precision กำหนดโดยเลขจำนวนเต็มฐาน 10 อยู่ภายในเครื่องหมายวงเล็บ

a) Decimal Fixed - Point Data

a decimal fixed point constant ประกอบด้วยเลขฐานสิบ ตั้งแต่ 1 ตัวขึ้นไป อาจจะมีจุดทศนิยมหรือไม่มีก็ได้ ถ้ามีเครื่องหมาย กำกับเครื่องหมายนี้จะต้องเป็น character ซ้ายมือสุดของเลขจำนวนนั้น

ตัวอย่าง

0	1.
−7	−6.2
320	+4.18
3.1416	0.0098

keyword attributes ที่ใช้สำหรับ declare decimal fixed point variable มีอยู่ 2 คำคือ DECIMAL และ FIXED ส่วน precision กำหนดโดยเลขจำนวนเต็มฐานสิบ 2 ตัว เขียนแยกกันด้วยเครื่องหมาย comma 1 ตัว อยู่ภายในวงเล็บ, เลขตัวแรกต้องไม่มีเครื่องหมายกำกับ หมายถึงจำนวนตัวเลขทั้งหมด, เลขตัวที่สองเรียกว่า scale factor อาจจะมีเครื่องหมายกำกับหรือไม่มีก็ได้ หมายถึงจำนวนตัวเลขที่อยู่ทางขวามือของจุดทศนิยม

ถ้าตัวแปรนั้นเป็นเลขจำนวนเต็ม, ไม่จำเป็นต้องเขียน ส่วนที่เป็น scale factor และเครื่องหมาย comma ที่อยู่ข้างหน้า attribute 2 คำนี้ จะเรียงลำดับตัวไหนก่อนตัวไหนหลังก็ได้ แต่ precsion ต้องเขียนหลัง attribute ตัวใดตัวหนึ่ง

ตัวอย่าง

```
DECLARE A FIXED DECIMAL (5,4);
DECLARE B FIXED (6,0) DECIMAL;
หรือ    DECLARE B FIXED DECIMAL (6);
```

จำนวนเลขสูงสุดของ fixed point ไม่เกิน 15 หลัก ถ้าไม่กำหนด precision ต้องไม่เกิน 5 หลัก

b) Binary Fixed - Point Data

มูลค่าคงที่ชนิดนี้ ประกอบด้วยเลขฐานสองตั้งแต่ 1 ตัวขึ้นไป อาจจะมีจุดทวินิยม (binary point) หรือไม่มีก็ได้ เสร็จตามด้วยอักษร B ระหว่าง character เหล่านี้ต้องไม่มี blank อยู่ด้วยและอาจจะมีเครื่องหมายนำหน้าตัวเลขจำนวนนี้หรือไม่มีก็ได้ ถ้ามีต้องเป็น character ตัวซ้ายมือสุด

ตัวอย่าง

1B	+101B
0B	11101B
1.01B	−111.01B
−1111B	1010101B
+1011.11B	111.000B

มูลค่าคงที่ชนิดนี้มี precision เป็น (p,q) เมื่อ p เป็นจำนวนเลขฐานสองทั้งหมดในมูลค่าจำนวนนี้, q เป็นจำนวนตัวเลขที่อยู่ทางขวามือของจุดทวินิยม

ตัวอย่าง

111.01B มี precision (5,2)

0001101B มี precision (7,0)

ตัวแปรชนิดนี้จะใช้ keyword attribute BINARY และ FIXED precision กำหนดโดยเลขจำนวนเต็มฐานสิบ 2 ตัว อยู่ภายในเครื่องหมายวงเล็บ หมายถึงจำนวนเลขฐานสองทั้งหมดและจำนวนเลขฐานสองทั้งหมดที่อยู่ทางขวามือของจุดทวินิยม ตามลำดับ ถ้าตัวแปรค่าตัวนั้นมีมูลค่าเป็นเลขจำนวนเต็ม (Integer) ไม่ต้องเขียนตัวเลข precision ตัวที่ 2 และเครื่องหมาย comma, attributes 2 คำนี้ จะเรียงลำดับอย่างไรก็ได้ แต่ precision ต้องเขียนทีหลังคำว่า BINARY หรือ FIXED

ตัวอย่าง

```
DECLARE TOTAL FIXED (8) BIN;
DECLARE FACTORIAL BINARY FIXED (20,2);
```

สำหรับ identifier ที่ไม่กำหนด attribute ให้ถ้า character ตัวแรกของชื่อนั้นเป็น I,J, K,L,M หรือ N หมายถึง binary fixed point variable และจะต้องมีมูลค่าอยู่ในขอบเขตจำกัดระหว่าง −32768 และ 32767 เสมอ และเครื่องจะ default attribute เป็น BINARY FIXED (15,0)

c) Decimal Floating Point Data

มูลค่าคงที่ชนิดนี้ประกอบด้วยเลขฐานสิบ ซึ่งอาจจะมีจุดทศนิยมหรือไม่มีก็ได้, ตามด้วยตัวอักษร E แล้วตามด้วย decimal integer exponent ที่อาจจะมีเครื่องหมายกำกับหรือไม่มีก็ได้ มูลค่าคงที่ชนิดนี้ทั้งหมดอาจมีเครื่องหมาย + หรือ − กำกับในตำแหน่งซ้ายมือสุดได้

ตัวอย่าง constants

15E-23	15E+23
4E3	5.417E0
−4832E56	3141593E - 6
.00314159E3	

มูลค่าคงที่ชนิดนี้จะมี precision (p) เมื่อ (p) เป็นจำนวนตัวเลขทั้งหมดที่ปรากฏทางด้านซ้ายมือของอักษร E เรียกว่า mantissa

ตัวอย่าง

5.417E0 มี precision (4) และ mantissa คือ 5.417

0.012E5 มี precision (4) และ mantissa คือ 0.012

การ declare ตัวแปรชนิดนี้ให้ใช้ keyword attributes DECIMAL และ FLOAT ส่วน precision เป็นเลขจำนวนเต็มฐานสิบ 1 ตัว อยู่ภายในเครื่องหมายวงเล็บ หมายถึง จำนวน significant digit ต่ำสุดที่ต้องการ ถ้ามูลค่าของ item ที่กำหนดให้กับตัวแปร มีความยาวมากกว่าที่ declare precision ไว้เครื่องคอมพิวเตอร์ จะตัดตัวเลขส่วนที่เกินทางขวามือทิ้งไป attributes 2 คำนี้ จะเรียงลำดับตัวไหนก่อนตัวไหนหลังก็ได้ แต่ precision ต้องอยู่หลัง attribute ตัวใดตัวหนึ่งเสมอ

ตัวอย่าง

```
DECLARE TAX FLOAT (6) DEC;
DECLARE REGISTER_YEARS DECIMAL FLOAT (5);
```

precision สูงสุดที่อาจกำหนดให้กับมูลค่าชนิดนี้คือ (33) ถ้าไม่กำหนดถือว่ามี precision เป็น (6),exponent ต้องเป็นตัวเลขจำนวนเต็มฐานสิบไม่เกิน 2 หลัก ขอบเขตจำกัดของมูลค่าชนิดนี้ประมาณ 10^{-78} ถึง 10^{75}

สำหรับ Identifier ตัวใดก็ตามถ้า character ตัวแรกเป็นตัวใดตัวหนึ่งใน A ถึง H หรือ O ถึง Z หรือตัวใดตัวหนึ่ง จาก $,#,@ เครื่องจะถือว่ามีมูลค่าเป็น decimal floating-point เมื่อไม่ declare attribute ไว้ เครื่องคอมพิวเตอร์จะ default attribute เป็น DECIMAL FLOAT (6)

d) Binary Floating Point Data

มูลค่าคงที่ชนิดนี้มีลักษณะคล้ายกับ decimal floating point data คือ แบ่งเป็น 2 ส่วน ส่วนแรกเป็น mantissa และส่วนที่สองคือ exponent ตัวเลขที่ประกอบเป็น mantissa ใช้เลขฐานสอง อาจจะมีจุดทวินิยมหรือไม่มีก็ได้ แต่ส่วนที่เป็น exponent ใช้เลขฐานสิบ วิธีกำหนดเหมือนกับใน decimal floating point data, เลขจำนวนนี้ (Binary floating point constant) จะต้องมีอักษร B เป็น character ตัวขวามือสุดเสมอ และอาจจะมีเครื่องหมายบวกหรือลบกำกับไว้ซ้ายมือสุดหรือไม่มีก็ได้

ตัวอย่าง

```
    11.00E0B        -11011E-5B
-11011.11E+6B       10.0111E12B
```

เมื่อ 11.00E0B หมายถึง $11.00*2^0$

-11011E-5B หมายถึง $-11011*2^{-5}$

e) Numeric Character Data

ได้แก่มูลค่าของตัวแปร หนึ่ง ๆ ซึ่ง declare ด้วย picture attribute และ numeric specification จะมีมูลค่าเป็น decimal fixed point หรือ decimal floating-point ก็ได้

numeric picture specification หนึ่ง ๆ จะบรรยายลักษณะของ character string นั้นโดยที่

charaters A หรือ X ใช้เป็น picture สำหรับ non-numeric

character 9 ใช้เป็น picture สำหรับ digit

character V ใช้เป็น picture สำหรับบอกตำแหน่งของจุดทศนิยม

picture specification ต้องอยู่ภายในเครื่องหมายคำพูดเปิด/ปิด เสมอ

ตัวอย่าง

'999V99'

numeric picture specification ข้างต้นนี้บรรยายว่า data item นี้ประกอบด้วยตัวเลขฐานสิบ 5 ตัว ตัวเลข 2 ตัวขวามือสุดหมายถึงเลขจุดทศนิยม หรืออาจเขียนดังนี้ก็ได้ '(3)9V(2)9' รูปแบบในการ declare numeric character variable มีดังนี้

```
DECLARE identifier PICTURE 'numeric-picture-specification';
```

ตัวอย่าง 1

```
DECLARE PRICE PICTURE '999V99';
```

ตัวอย่าง 2

```
DECLARE PRICE PICTURE '$99V.99',
COST CHARACTER (6),
VALUE FIXED DECIMAL (6,2);
PRICE = 12.28;
COST = '$ 12.28';
```

แบบฝึกหัด

จงบอกว่าตัวเลขต่อไปนี้เป็น fixed-point decimal หรือ floating-point decimal หรือ integer หรือ ไม่ใช่ทั้ง 3 ชนิดที่กล่าวมาข้างต้น

a) 1
b) −1
e) 1.0
d) .002
e) −2.5E0
f) +3.42E−2
g) 17E−1
h) E+6
i) +E+3
j) 2.0E−2.7
k) 8.E1
l) 1/100
m) 99
n) 1−
o) 76.315492E−35

2.1.2 String data (Non-numeric data)

ได้แก่มูลค่าที่เอาไปคำนวณไม่ได้ แบ่งออกเป็นสองชนิดคือ character string และ bit string

a) Character string data

ประกอบด้วย character อะไรก็ได้เมื่อเขียนในโปรแกรม ต้องอยู่ภายในเครื่องหมายคำพูดเปิด/ปิด แต่ถ้าใน string นั้นมี character ที่เป็นเครื่องหมาย quotation mark 1 ตัว ให้แทนด้วยเครื่องหมาย quotation mark 2 ตัวติดต่อกัน, ความยาวของ character string หนึ่ง ๆ คือจำนวน characters ที่อยู่ภายในเครื่องหมาย quotation mark เปิด/ปิด ถ้าเครื่องหมาย quotation mark 2 ตัวติดกันที่อยู่ภายใน string ใช้แทนเครื่องหมาย quotation mark 1 ตัว ให้นับเป็น 1 ตัว

ตัวอย่าง character string constants

'PL/1'

'PAGE 15'

'AC 401-19'

'JOHN' 'SISTER'

'LOGARITHM TABLE'

'PROGRAMMING LANGUAGE 1'

'RAMKHAMHAENG-1985'

(2) 'WELLA' หมายถึง 'WELLAWELLA' ในที่นี้ (2) เป็น repetition factor

สำหรับ null character-string constant หรือ empty string หมายถึง string ที่ไม่มี character เลยความยาวของ string จะเท่ากับ 0 แทนด้วยเครื่องหมาย quotation mark 2 ตัวติดกัน ('') keyword attribue ที่ใช้สำหรับ declare character string variable ใช้คำว่า CHARACTER หรือ CHAR

ตัวอย่าง

DECLARE NAME CHARACTER (15);

หมายถึง NAME มีมูลค่าเป็น character string มีความยาว 15 ตัว ถ้า string นั้นมีความยาวไม่ถึง 15 ตัว เครื่องจะใส่ blank ทางขวามือให้จนเต็ม 15 ตัว แต่ถ้า string นั้นยาวเกินไป เครื่องจะตัด character ทางขวามือซึ่งเป็นส่วนที่เกินทิ้งไป

แต่ถ้า string นั้นไม่กำหนดความยาวไว้แน่นอน ให้ใช้ VARYING attribute

ตัวอย่าง

DECLARE NAME CHARACTER (15) VARYING;

หมายความว่า identifier NAME มีมูลค่าเป็น string ความยาวเท่าไหร่ก็ได้ ไม่เกิน 15 ตัว character string variable อาจจะ declare ด้วย PICTURE attribute ได้ด้วยรูปแบบดังนี้

PICTURE 'character -picture -specification'

character-picture-specification ประกอบด้วย character A,X, และ 9 เขียนภายในเครื่องหมายคำพูดเปิด/ปิด

picture character A หมายถึง field นั้นเป็นมูลค่า alphabetic character หรือ blank

picture character X หมายถึง field นั้นมีมูลค่าเป็น character อะไรก็ได้

picture character 9 หมายถึง field นั้นมีมูลค่าเป็น numeric character หรือ blank

ตัวอย่าง

DECLARE PART_NO PICTURE 'AA9999X999';

หรือ '(2)A(4)9X(3)9';

b) Bit string data

แบ่งออกเป็น 2 ชนิดเช่นเดียวกันคือ ตัวแปร และมูลค่าคงที่ในกรณีที่เป็น bit string constant จะประกอบด้วยเลขฐานสองเท่านั้น ตัวเลขนี้ต้องอยู่ภายในเครื่องหมายคำพูด และมีอักษร B เป็น character ตัวขวามือสุด

ตัวอย่าง constants

'0'B '1'B '101011'B '11110000'B

สำหรับ variable การ declare ให้ใช้ attribute keyword คำว่า BIT

สรุป

1. ในกรณีที่โปรแกรมเมอร์ ไม่ declare identifier ไว้ในตัวโปรแกรม, โปรแกรม compiler จะจัด default attribute ให้ทั้งนี้ขึ้นอยู่กับว่า character ตัวแรกของชื่อ identifier ตัวนั้นคืออะไร

ถ้า character ตัวแรกของชื่อนั้นเป็น I,J,K,L,M, หรือ N เครื่องจะจัด BINARY FIXED (15) ให้แต่ถ้า character ตัวแรกของชื่อนั้นเป็นตัวอักษรอื่น ๆ ที่ไม่ใช่ทั้ง 6 ตัวที่กล่าวข้างต้น เครื่องจะ default ให้เป็น DECIMAL FLOAT (6)

2. สำหรับ identifiers ซึ่ง declare attribute ให้เพียงบางส่วน ภาษา PL/1 ได้กำหนดมาตรฐานการ default ไว้ด้วย กรณีที่มีเฉพาะ base หรือเฉพาะ scale อย่างเดียวดังตารางข้างล่างนี้

ให้ A,B,C,D,E,F เป็น identifiers

declarative	**default attribute**
A BINARY	BINARY FLOAT (21)
B DECIMAL	DECIMAL FLOAT (6)
C FIXED	DECIMAL FIXED (5)
D FLOAT	DECIMAL FLOAT (6)
E CHAR	CHAR (1)
F BIT	BIT (1)

ถ้าเรา declare base และ precision เท่านั้นแต่ไม่กำหนด scale (FIXED หรือ FLOAT) การ default สำหรับ scale ขึ้นอยู่กับ precision ที่กำหนดนั้นว่า เป็น (p) หรือ (p,q) ดังตารางข้างล่างนี้

ให้ G และ H เป็น identifiers

declarative	**default attribute**
G BINARY (15)	BINARY FLOAT (15)
H BINARY (15,5)	BINARY FIXED (15,5)

Entry data

ใช้เมื่อติดต่อกับ entry name เท่านั้น entry data item อาจจะเป็น entry constant หรือมูลค่าของ entry variable ก็ได้

entry constant เป็น identifier ปรากฏในโปรแกรมในลักษณะของ entry name ใช้เป็น prefix ของคำสั่ง PROCEDURE หรือคำสั่ง ENTRY อาจจะใช้อ้างถึง entry point ใน procedure หนึ่ง ๆ ก็ได้

ตัวอย่าง

```
P: PROCEDURE;
CALL P1;
CALL P2;
P1: PROCEDURE;
P2: ENTRY;
```

ในที่นี้ P1, P2 เป็น entry constants

เมื่อมีการอ้างถึง entry constant control ก็จะย้ายไปยัง procedure entry point นั้น ส่วน entry variable เป็น identifier ซึ่งอ้างถึง entry constant

ตัวอย่าง

```
DECLARE EV ENTRY VARIABLE,
(E1, E2) ENTRY;
    .
    .
    .
EV = E1;
CALL EV;
EV = E2;
CALL EV;
```

2.2 Program Control Data

Data ประเภทนี้ได้แก่ file, label, entry, event, locater, task และ area

File data file item หมายถึง information เกี่ยวกับ PL/1 file อาจจะเป็น file constant หรือมูลค่าของ file variable ก็ได้

Label data label data item หมายถึง label constant หรือมูลค่าของ label variable

Label constant เป็น identifier ใช้เป็น prefix ของ statement หนึ่ง ๆ เพื่อที่ว่าระหว่างที่มีการ execute โปรแกรม, การควบคุมโปรแกรมสามารถย้ายกลับไปยังคำสั่ง โดยอ้างถึง label นี้ให้ใช้เครื่องหมาย colon 1 ตัวแยกระหว่าง label กับคำสั่ง

ตัวอย่าง

```
ABC: MILES = SPEED* HOURS;
```

ในที่นี้ ABC เป็น statement label constant

หมายถึงคำสั่งนี้จะได้รับการ execute ตามลำดับคำสั่งที่ปรากฏ หรือโดยการย้าย control กลับมายังคำสั่งนี้จากตำแหน่งอื่นภายในโปรแกรมก็ได้ โดยการใช้คำสั่ง GO TO

ส่วน statement label variable เป็น identifier อ้างถึง statement label variable

ตัวอย่าง

```
A:  statemant ;
B:  statement ;
        X = A;
        GO TO X;
```

ในที่นี้ A,B เป็น statement label constants

X เป็น statement label variableสำหรับ statement label variable ต้อง declare ด้วย attribute LABEL ดังนี้

```
DECLARE X LABEL;
```

ตัวอย่างโปรแกรม

เลขต่อไปนี้ 1, 1, 2, 3, 5, 8, 13, 21,.....มีชื่อเรียกว่าอนุกรม fibonacci ตัวเลขแต่ละเทอมได้จากการเอาตัวเลข 2 เทอมที่อยู่ก่อนหน้านั้น บวกเข้าด้วยกัน ส่วนเทอมแรกและเทอมที่สองมีค่าเท่ากับ 1

จงเขียน flowchart และโปรแกรมคำนวณและพิมพ์ตัวเลขของอนุกรม Fibonacci 50 เทอมแรก

วิธีที่ 1

```
PL/I OPTIMIZING COMPILER        SUM: PROCEDURE OPTIONS (MAIN);

                        SOURCE LISTING

STMT    LEV  NT

   1          0  SUM: PROCEDURE OPTIONS (MAIN);
                 /*THIS PROGRAM COMPUTE THE FIRST NUMBER IN THE
                 FIBONACCI SERIES */
                 /*WHICH EACH TERM IS OBTAINED BY ADDING TOGETHER
                 THE PREVIOUS */
                 /*TWO TERM */
   2     1     0      DCL T (50);
   3     1     0      DCL N FIXED BIN;
   4     1     0      T (1) = 1;
   5     1     0      T (2) = 1;
   6     1     0      PUT LIST (T(1),T(2));
   7     1     0      DO N=3 TO 50;
   8     1     1      T(N)=T(N-1)+T(N-2);
   9     1     1      PUT LIST (T(N));
  10     1     1      END;
  11     1     0  END SUM;
```

ลักษณะของ output จะเป็นดังนี้

1.C0000E+00	1.00000E+00	2.00000E+00	3.00000E+00	5.00000E+00
8.C0000E+00	1.30000E+01	2.10000E+01	3.40000E+01	5.50000E+01
8.S0000E+01	1.44000E+02	2.33000E+02	3.77000E+02	6.10000E+02
S.E7000E+02	1.59700E+03	2.58400E+03	4.18100E+03	6.76500E+03
1.C94o0E+04	1.77110E+04	2.86570E+04	4.63680E+04	7.50250E+04
1.21393E+05	1.96418E+05	3.17811E+05	5.14229E+05	8.32040E+05
1.34626E+06	2.17830E+06	3.52457E+06	5.70288E+06	9.22746E+06
1.49303E+07	2.41578E+07	3.90881E+07	6.32459E+07	1.02334E+08
1.65580E+C8	2.67914E+08	4.33494E+08	7.01408E+08	1.13490E+09
1.83631E+09	2.97121E+09	4.80752E+09	7.77872E+09	1.25862E+10

วิธีที่ 2

PL/I OPTIMIZING COMPILER /* FIBONACCI SEQUENCE */

SOURCE LISTING

```
STMT LEV NT
                        /* FIBONACCI SEQUENCE */
   1      0  EX206 : PROC OPTIONS (MAIN);
   2   1  0    DCL A (20) REAL DECIMAL FIXED (10) INIT (1,1, (18) 0);
   3   1  0    DCL (I,J,N) REAL DECIMAL FIXED (10) ;
   4   1  0    DO N = 3 TO 20;
   5   1  1      A (N) = A (N-1) + A (N-2);
   6   1  1      END;
   7   1  0    PUT LIST ('THE FIRST 20 FIBONACCI NUMBERS ARE : ');
   8   1  0    DO J = 1 TO 20 BY 3;
   9   1  1      PUT SKIP LIST  ((A(I) DO I=J TO J+2 WHILE (I<21)));
  10   1  1      END;
  11   1  0      END EX206;
```

```
THE FIRST 20 FIBONACCI NUMBERS ARE :
           1           1           2
           3           5           8
          13          21          34
          55          89         144
         233         377         610
         987        1597        2584
        4181        6765
```

วิธีที่ 3

```
PL/I OPTIMIZING COMPILER                    FIBO : PROC OPTIONS(MAIN);

                        SOURCE LISTING

    STMT LEV NT
      1      0  FIBO : PROC OPTIONS (MAIN) ;
                   /* THIS PROGRAM IS THE FIBONACCI SERIES */
      2    1 0   DCL T (50) FIXED DEC (11), SYSPRINT OUTPUT;
      3    1 0     DCL I FIXED DEC (2) ;
      4    1 0   PUT PAGE EDIT ('THIS PROGRAM IS THE FIBONACCI
                   SERIES')
                     (COL (49), A) ;
      5    1 0     T (1), T (2) = 1;
      6    1 0   DO I = 3 TO 50;
      7    1 1     T(I)=T(I-1)+T(I-2); END;
      9    1 0     PUT EDIT ((( 'T(',I,') = ', T (I)) DO I = 1 TO 50))
                     (SKIP (2), A,F (2),A,F (11));
     10    1 0     END FIBO;


                THIS PROGRAM IS THE FIBONACCI SERIES
                        T( 1) =          1
                        T( 2) =          1
                        T( 3) =          2
                        T( 4) =          3
                        T( 5) =          5
                        T( 6) =          8
                        T( 7) =         13
                        T( 8) =         21
                        T( 9) =         34
                        T(10) =         55
                        T(11) =         89
```

T(12) =	144
T(13) =	233
T(14) =	377
T(15) =	610
T(16) =	987
T(17) =	1597
T(18) =	2584
T(19) =	4181
T(20) =	6765
T(21) =	10946
T(22) =	17711
T(23) =	28657
T(24) =	46368
T(25) =	75025
T(26) =	121393
T(27) =	196418
T(28) =	317811
T(29) =	514229
T(30) =	832040
T(31) =	1346269
T(32) =	2178309
T(33) =	3524578
T(34) =	5702887
T(35) =	9227465
T(36) =	14930352
T(37) =	24157817
T(38) =	39088169
T(39) =	63245986
T(40) =	102334155
T(41) =	165580141

T(42) =	267914296
T(43) =	433494437
T(44) =	701408733
T(45) =	1134903170
T(46) =	1836311903
T(47) =	2971215073
T(48) =	4807526976
T(49) =	7778742049
T(50) =	12586269025

2.3 การจัดระเบียบข้อมูล (Data Organization)

ในภาษา PL/1 data item อาจจะเป็น data elements ตัวเดียวหรือกลุ่มของ data ที่เรียกว่า arrays และ strúctures

ตัวแปรที่มีมูลค่าเป็น single element เรียกว่า element variable (หรือ scalar variable) ส่วนตัวแปร ที่มีมูลค่าเป็น กลุ่มของ data elements เรียกว่า array variable หรือ structure variable

2.3.1 Arrays

data elements ที่มีลักษณะอย่างเดียวกัน เช่นเป็นชนิดเดียวกันมี precision เหมือนกัน และความยาวเท่ากัน อาจจะรวมเข้าด้วยกันแล้ว form เป็น array ชุดหนึ่งก็ได้

array หนึ่ง ๆ หมายถึงกลุ่มของอีลิเม้นต์ n มิติ ทุกตัวมี attribute เหมือนกัน กำหนดโดยชื่อ item แต่ละตัวของ array นี้เวลานำมาใช้อ้างถึงตำแหน่งที่ภายใน array นั้น

ตัวอย่าง

DECLARE A(8) FIXED DECIMAL (3);

DECLARE B(2,3) FIXED DECIMAL (3);

ตัวอย่างในบรรทัดแรก A เป็น array 1 มิติมีอีลิเม้นต์ อยู่ทั้งหมด 8 ตัว แต่ละตัวเป็นเลขฐานสิบ ประกอบด้วยตัวเลข 3 หลัก ส่วนบรรทัดที่ 2 B เป็น array 2 มิติมีอีลิเม้นต์ทั้งหมด 6 ตัว แต่ละตัวมีลักษณะอย่างเดียวกันกับอีลิเม้นต์ในบรรทัดแรก

ตัวเลขที่อยู่ในเครื่องหมายวงเล็บตามชื่อ array เป็น dimension attribute specification ใช้สำหรับกำหนดจำนวนมิติและขอบเขตของ array นั้น หรือขอบเขตของแต่ละมิติ ถ้ามีขอบเขตเดียวกัน array นั้นก็เป็น 1 มิติถ้ามีสองขอบเขตเขียนแยกกันด้วยเครื่องหมาย comma, array นั้นก็เป็น 2 มิติ

ขอบเขตของแต่ละมิติ หมายถึงการเริ่มต้นและการสิ้นสุดของมิตินั้น ถ้าตัวเลขที่ปรากฏนั้นเป็นเลขจำนวนเต็มตัวเดียว หมายถึงขอบเขตล่าง (lower bound) มีค่าเท่ากับ 1 เช่นจากตัวอย่าง array A (8) มีขอบเขตล่างเป็น 1 และขอบเขตบนเป็น 8, ส่วน array B(2,3) มีขอบเขต 1 และ 2 กับ 1 และ 3 ถ้าขอบเขตล่างของมิตินั้นไม่ใช่ 1 จำเป็นต้องกำหนดให้ชัดเจนว่าขอบเขตล่างเท่ากับอะไรและขอบเขตบนเท่ากับอะไรเป็นตัวเลขสองตัวเขียนแยกกันด้วยเครื่องหมาย colon 1 ตัว

ตัวอย่าง

DECLARE A (4:11);

DECLARE B (-4:3);

จากคำสั่งสองบรรทัดข้างต้นทั้ง array A และ array B เป็น array 1 มิติ มีจำนวนอีลิเม้นต์เท่ากันคือ 8 ตัวเรียกว่า extent 8 ถึงแม้ว่าขอบเขตจะไม่เหมือนกันก็ตาม

สมมุติว่าข้อมูลทั้ง 8 ตัวข้างล่างนี้ กำหนดให้กับ array A

2 5 10 3 1 −1 7 40

หมายถึง

A(4) มีมูลค่าเท่ากับ 2

A(5) มีมูลค่าเท่ากับ 5

A(6) มีมูลค่าเท่ากับ 10

A(7) มีมูลค่าเท่ากับ 3

A(8) มีมูลค่าเท่ากับ 1

A(9) มีมูลค่าเท่ากับ −1

A(10) มีมูลค่าเท่ากับ 7

A(11) มีมูลค่าเท่ากับ 40

ตัวเลขที่อยู่ภายในวงเล็บตามหลังชื่อ array เรียกว่า subscript เป็นการระบุ data item ตัวที่ต้องการภายใน array ส่วน subscripted name เช่น A(5) หมายถึง อีลิเม้นต์ตัวเดียว และเป็น an element variable, ถ้าต้องการให้หมายถึง อีลิเม้นต์ทั้งหมดของ array ให้ใช้ชื่อ array โดยไม่ต้องมี subscript เช่น A ในที่นี้ A จะเป็น array variable

จากตัวเลข 8 ตัวชุดเดิม ถ้ากำหนดให้กับ array B

หมายถึง

B(-4) มีมูลค่าเท่ากับ 2

B(-3) มีมูลค่าเท่ากับ 5

B(-2) มีมูลค่าเท่ากับ 10

B(-1) มีมูลค่าเท่ากับ 3

B(0) มีมูลค่าเท่ากับ 1

B(1) มีมูลค่าเท่ากับ -1

B(2) มีมูลค่าเท่ากับ 7

B(3) มีมูลค่าเท่ากับ 40

จำนวนมิติของ array สามารถกำหนดได้ตั้งแต่ 1 มิติจนถึง 15 มิติ

Expression as Subscripts

subscripts ของ subscripted name ไม่จำเป็นต้องเป็นมูลค่าคงที่ (constant) เสมอไป ใน expression หนึ่ง ๆ ซึ่งเป็น subscripts นั้น ถ้าการประเมินผลแล้วได้เป็นตัวเลขที่ไม่ใช่เลขจำนวนเต็ม เครื่องคอมพิวเตอร์จะเปลี่ยนให้เป็นเลขจำนวนเต็มให้เอง

บ่อย ๆ ครั้งที่ subscript อาจจะเป็น ตัวแปรอื่น ๆ หรือ expression อื่น ๆ

ตัวอย่าง

C (I, J*K)

หมายถึง อีลิเม้นต์ ตัวไหนก็ได้ ของ array C เมื่อ กำหนดมูลค่าของ I, J และ K ให้

Cross-Section of Arrays (การแบ่งข้อมูลใน array)

หมายถึงการใช้เครื่องหมาย asterisk (*) แทน subscript หนึ่ง ๆ ที่อยู่ภายใน subscripted name, asterisk เป็นการระบุจำนวน extent ทั้งหมดที่ใช้

ตัวอย่าง

A(*,1) หมายถึงอีลิเม้นต์ทุกตัวที่อยู่ในคอลัมน์ที่ 1 ของ array A

A(2,*) หมายถึงอีลิเม้นต์ทุกตัวที่อยู่ในแถวที่ 2 ของ array A

A(*,*) หมายถึงอีลิเม้นต์ทั้งหมดของ array A

การใช้ Array

1. การกำหนดมูลค่า ให้กับ array name จะให้ผลลัพธ์เหมือนกับการกำหนดมูลค่าให้อีลิเม้นต์ทุกตัวที่อยู่ใน array นั้น

ตัวอย่าง

```
DCL A(2,3);
A = 1.5E1;
```

หมายความว่า อีลิเม้นต์ทั้ง 6 ตัวของ array A ทุกตัวมีค่าเท่ากับ 1.5E1

ถ้าต้องการกำหนดมูลค่าให้กับอีลิเม้นต์เพียงบางตัวต้องกำหนด subscript ไว้ด้วยเช่น A(2,1) = 2.4E0;

2. การคำนวณกับ array name จะให้ผลลัพธ์เหมือนกับการคำนวณที่กระทำกับอีลิเม้นต์ทุกตัวใน array นั้น เช่น A = A + 1; มีความหมายอย่างเดียวกับ

```
A(1,1) = A(1,1) + 1;
A(1,2) = A(1,2) + 1;
A(1,3) = A(1,3) + 1;
A(2,1) = A(2,1) + 1;
A(2,2) = A(2,2) + 1;
A(2,3) = A(2,3) + 1;
```

หรือ `A = 5 * A;`

หมายความว่า อีลิเม้นต์ ทุกตัวใน array A คูณด้วย 5

3. ถ้ามีการคำนวณระหว่าง array หลายชุด, array เหล่านั้นต้องมีขนาดเดียวกันและอีลิเม้นต์ตัวที่อยู่ตำแหน่งที่ตรงกันของ array ต่าง ๆ จะถูกนำมาคำนวณ

ตัวอย่าง

จงแสดงผลลัพธ์ของ array B,C, และ D จากโปรแกรมข้างล่างนี้

```
DCL (A,B,C,D) (2:3,2) FIXED DEC;
  A(2,1) = 1;
  A(2,2) = 2;
  A(3,1) = 3;
  A(3,2) = 4;
    B = 2 * A;
    C = A + B;
    D = A/B;
```

เฉลย

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad B = \begin{bmatrix} 2 & 4 \\ 6 & 8 \end{bmatrix}$$

$$C = \begin{bmatrix} 3 & 6 \\ 9 & 12 \end{bmatrix}, \quad D = \begin{bmatrix} .5 & .5 \\ .5 & .5 \end{bmatrix}$$

4. การใช้เครื่องหมาย asterisk (*)

สำหรับ array 2 มิติ ถ้าเรียกชื่อ array จะได้อีลิเม้นต์ทุกตัวถ้ากำหนด subscript จะได้เฉพาะอีลิเม้นต์ตัวที่ต้องการ ถ้าหากต้องการทั้งโรว์ (row) หรือทั้งคอลัมน์ (column) ให้ใส่เครื่องหมาย * ลงไปใน dimension specification แทนตัวเลข

ตัวอย่าง

A(3,*) หมายถึงอีลิเม้นต์ทุกตัวในโรว์ที่ 3 ของ array A

A(*,7) หมายถึงอีลิเม้นต์ทุกตัวใน คอลัมน์ ที่ 7 ของ array A

ตัวอย่าง

```
DCL A(5,5);
PUT LIST (A(3,*));
```

ทั้ง 2 คำสั่งนี้ จะมีความหมายอย่างเดียวกับ 2 คำสั่งต่อไปนี้

```
DCL A(5,5);
PUT LIST ((A(3,I) DO I = 1 TO 5));
```

หมายเหตุ

ถ้าเขียน A(*,*) มีความหมายอย่างเดียวกับ A

ตัวอย่าง การ transpose matrix

```
PQR : PROCEDURE OPTIONS (MAIN);
    DCL MATRIX (3,3) FIXED DEC (3) INIT (1,2,3,4,5,6,7,8,9);
    DCL TRANS (3,3) FIXED DEC (3);
    DO K = 1 TO 3;
    DO L = 1 TO 3;
        TRANS (L,K) = MATRIX (K,L);
    END;
    END;
    PUT SKIP LIST (TRANS);
END PQR;
```

หมายเหตุ ภาษาพีแอลวัน เครื่องจะเก็บข้อมูลแบบ by rows

ผลลัพธ์จะเป็นดังนี้

$$TRANS_{3\times3} = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

ซึ่งใน do-group **ที่ซ้อนกันอยู่นั้นเราสามารถแทนด้วยคำสั่งข้างล่างนี้ได้**

```
DO K = 1 TO 3;
TRANS (*,K) = MATRIX (K,*); END;
```

ตัวอย่างโปรแกรม

สร้างและพิมพ์ matrix A ขนาด 10 โรว์ 10 คอลัมน์ มีมูลค่าดังนี้ โดยไม่มีการอ่านข้อมูลเข้ามา

$$A_{10\times10} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 & 26 & 27 & 28 & 29 & 30 \\ 31 & 32 & 33 & 34 & 35 & 36 & 37 & 38 & 39 & 40 \\ & & & & \vdots & & & & & \\ & & & & \vdots & & & & & \\ & & & & \vdots & & & & & \\ & & & & \vdots & & & & & \\ 91 & 92 & 93 & 94 & 95 & 96 & 97 & 98 & 99 & 100 \end{bmatrix}$$

```
1       0 PROG5 :   PROC OPTIONS (MAIN);
2   1   0           DCL SYSPRINT OUTPUT;
3   1   0           DCL (I,J) FIXED DEC (2);
4   1   0           DCL A(10,10) FIXED DEC (3);
5   1   0           DO J = 1 TO 10;
6   1   1           A(1,J) = J;
7   1   1           END;
8   1   0           DO I = 2 TO 10;
9   1   1           A(I,*) = A(1,*) + 10*(I-1);
10  1   1           END;
11  1   0           PUT SKIP EDIT ('MATRIX A') (COL(45),A);
12  1   0           PUT SKIP EDIT ('SIZE = 10*10')(COL(45),A);
13  1   0           PUT EDIT (((A(I,J) DO J = 1 TO 10) DO I = 1 TO 10))
                        (SKIP, COL(35), (10) (F(3),X(2)));
14  1   0 END PROG5;
```

MATRIX A

SIZE = 10*10

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

ตัวอย่าง ประมวลผลจากเครื่องไมโครคอมพิวเตอร์ ขนาด 8 บิท

```
ARRAY:PROCEDURE OPTIONS(MAIN);
      DCL A(10,10) FIXED(5,0),
          (I,J) FIXED BIN;
      DO I = 1 TO 10;
       DO J = 1 TO 10;
        A(I,J) = (I*10)-10+J;
       END;
      END;
      PUT LIST('MATRIX a');
      PUT SKIP EDIT(((A(I,J)DO J= 1 TO 10)DO I = 1 TO 10))
                    (COL(5),(10(F(5),X(2))));
      END ARRAY;
```

```
B>
MATRIX a
    1     2     3     4     5     6     7     8     9    10
   11    12    13    14    15    16    17    18    19    20
   21    22    23    24    25    26    27    28    29    30
   31    32    33    34    35    36    37    38    39    40
   41    42    43    44    45    46    47    48    49    50
   51    52    53    54    55    56    57    58    59    60
   61    62    63    64    65    66    67    68    69    70
   71    72    73    74    75    76    77    78    79    80
   81    82    83    84    85    86    87    88    89    90
   91    92    93    94    95    96    97    98    99   100
End of Execution
```

ตัวอย่างโปรแกรม

กำหนด square matrix A, B, C และ D ขนาด 4 โรว์ (rows) 4 คอลัมน์ (columns) จงเขียนโปรแกรมสร้าง matrix E ขนาด 4 × 4 เช่นกัน โดยมีหลักเกณฑ์ดังนี้

โรว์ที่ 1 ของ matrix E = คอลัมน์ที่ 1 ของ matrix A + โรว์ที่ 2 ของ matrix B

โรว์ที่ 2 ของ matrix E = คอลัมน์ที่ 3 ของ matrix B * โรว์ที่ 2 ของ matrix C

โรว์ที่ 3 ของ matrix E = คอลัมน์ที่ 4 ของ matrix A − 7 * โรว์ที่ 3 ของ matrix D

โรว์ที่ 4 ของ matrix E = โรว์ที่ 1 ของ matrix C / คอลัมน์ที่ 3 ของ matrix A

ข้อมูลสมมุติ

$$A_{4\times4} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{bmatrix} \quad , B_{4\times4} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$C_{4\times4} = \begin{bmatrix} 6 & -3 & 6 & 3 \\ 2 & -1 & 2 & -1 \\ 2 & -1 & 2 & -1 \\ 2 & -1 & 2 & -1 \end{bmatrix} \quad , D_{4\times4} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 \end{bmatrix}$$

SOURCE LISTING

```
STMT LEV NT
   1   1   0 PROC: PROC OPTIONS (MAIN);
   2   1   0       DCL SYSIN INPUT, SYSPRINT OUTPUT,
                   (I,J) FIXED DEC (1),
                   (A,B,C,D,E) (4,4) FIXED DEC (3,1);
   3   1   0       ON ENDFILE (SYSIN) STOP;
   4   1   0       GET LIST (A,B,C,D);
   5  10   0       E (1,*) = A (*,1) + B (2,*);
   6   1   0       E (2,*) = B (*,3) * C (2,*);
   7   1   0       E (3,*) = A (*,4) - 7 * D (3,*);
   8   1   0       E (4,*) = C (1,*) / A (*,3);
   9   1   0       PUT SKIP EDIT ('MATRIX E') (COL (45), A);
  10   1   0       PUT SKIP EDIT ('SIZE = 4*4') (COL (45), A);
  11   1   0       PUT EDIT (((E(I,J) DO J = 1 TO 4) DO I = 1 TO 4))
                                (SKIP, COL (35), (4) (F(5,1)));
  12   1   0       END PROC;
```

MATRIX E

SIZE = 4*4

1.0	1.0	1.0	1.0
2.0	0.0	2.0	0.0
−17.0	−17.0	−17.0	−17.0
2.0	−1.0	2.0	−1.0

2.3.2 โครงสร้าง (Structures)

data items ซึ่งไม่จำเป็นต้องมีลักษณะเหมือนกันทุกประการ แต่ทุกตัวมีความสัมพันธ์ทางด้าน logical ซึ่งกันและกัน สามารถนำมารวมกันแล้วเรียกว่า โครงสร้าง (structures)

เช่นเดียวกับในเรื่อง array เราต้องกำหนดชื่อให้กับโครงสร้างทั้งหมดซึ่งหมายถึงข้อมูลทั้งหมด แต่ไม่เหมือนกับ array ตรงที่ว่า อีลิเม้นต์แต่ละตัวของโครงสร้างนั้นต้องกำหนดชื่อแต่ละชื่อให้อีกด้วย

โครงสร้างหนึ่ง ๆ เป็นการจัดกลุ่มเรียงลำดับความสำคัญของชื่อต่าง ๆ ที่เบื้องล่างสุดของฐานเป็นกลุ่มของอีลิเม้นต์แต่ละตัวอาจจะเป็น single data item หรือ an array ก็ได้ ส่วนยอดสุดของ ลำดับความสำคัญนี้ก็จะเป็นชื่อโครงสร้าง ซึ่งหมายถึง element variables ทั้งหมด

ตัวอย่าง กลุ่มของ element variables ต่อไปนี้ต้องใช้ในการคำนวณหารายได้ต่อสัปดาห์ของลูกจ้างแต่ละคน

LAST_NAME
FIRST_NAME
REGULAR_HOURS
OVERTIME_HOURS
REGULAR_RATE
OVERTIME_RATE

ตัวแปรเหล่านี้เมื่อรวมกันเข้าเป็นโครงสร้างและ กำหนดชื่อให้กับโครงสร้างว่า PAYROLL. PAYROLL จะหมายถึง อีลิเม้นต์ ทั้งหมด

PAYROLL

LAST_NAME	REGULAR_HOURS	REGULAR_RATE
FIRST-NAME	OVERTIME_HOURS	OVERTIME_RATE

เมื่อใดก็ตามที่อ้างถึง PAYROLL หมายถึงต้องการ element variables ทุกตัว

ตัวอย่าง

```
GET DATA (PAYROLL);
```

คำสั่งนี้หมายถึง กำหนดข้อมูลให้กับ element variable ทุกตัวในโครงสร้าง PAYROLL

แต่เพื่อความสะดวกเขามักจะแบ่งกลุ่มนี้ทั้งหมด ให้เป็นกลุ่มย่อยลงไปอีก จากตัวอย่างข้างต้นสามารถแบ่งออกได้เป็นอีก 3 กลุ่ม คือ ชื่อ, จำนวนชั่วโมงทำงาน, และอัตราค่าจ้างต่อชั่วโมง ในกลุ่มย่อยนี้ เราต้องกำหนดชื่อให้เช่นกัน

PAYROLL

NAME	HOURS	RATE
FIRST	REGULAR	REGULAR
LAST	OVERTIME	OVERTIME

โปรดสังเกตว่าลำดับความสำคัญของชื่อ แบ่งออกได้เป็นหลายระดับ ระดับแรกเป็นชื่อโครงสร้าง เรียกว่า major structure names ส่วนระดับรองลงไปจากชื่อของ substructures เรียกว่า minor structure names และระดับต่ำที่สุด เป็น elementary name เรียกว่า elementary name เมื่อ elementary name หนึ่ง ๆ ภายในโครงสร้างอาจจะเป็น array ก็ได้ (array variable) แต่เป็น element variable ไม่ได้

การจัดระเบียบโครงสร้างหนึ่ง ๆ ให้ใช้คำสั่ง declare กับ level number สำหรับ major structure name ให้ declare ด้วย leve number 1 ส่วน minor structures และ elementary names ให้ declare ด้วย level number ซึ่งเป็นตัวเลขจำนวนเต็มฐานสิบมากกว่า 1 ทั้งนี้ระหว่าง level number กับ ชื่อที่เกี่ยวข้องกันให้เว้นอย่างน้อย 1 blank

ตัวอย่าง

มูลค่าทั้งหมดที่ใช้คำนวนหาค่าจ้าง ต่อสัปดาห์ของลูกจ้างแต่ละคน declare ได้ดังนี้

```
DECLARE 1 PAYROLL,
          2 NAMES,
            3 LAST,
            3 FIRST,
          2 HOURS,
            3 REGULAR,
            3 OVERTIME,
          2 RATE,
            3 REGULAR,
            3 OVERTIME;
```

ข้อสังเกต

(1) ในการ declare structure ของ PAYROLL จริง ๆ นั้น ต้องมี attribute กำหนดให้กับ elementary names ทั้ง 6 ชื่อ และคำสั่ง declare ทั้งหมดข้างต้น อาจจะเขียนในบรรทัดเดียวกันก็ได้เช่น

DELARE 1 PAYROLL, 2 NAME, 3 LAST,...

(2) ส่วนตัวเลขที่ใช้เป็น level numbers นั้นที่มีมูลค่ามากกว่ากันไม่จำเป็นต้องเรียงกันตามลำดับมูลค่า เพียงแต่ระบุลำดับความสำคัญของชื่อก็พอ เช่น minor structure ที่มี level n จะประกอบด้วย ชื่อทั้งหมดที่มี level numbers เป็นตัวเลขมากกว่า n ซึ่งอยู่ระหว่าง minor structure name นั้นกับชื่อถัดไป ที่มี level number น้อยกว่าหรือเท่ากับ n จากตัวอย่างข้างต้น อาจจะเขียนได้อีกหนึ่งรูปแบบคือ

```
DECLARE 1 PAYROLL,
          4 NAMES,
            5 LAST,
            5 FIRST,
          2 HOURS,
            6 REGULAR,
            5 OVERTIME,
          2 RATE,
            3 REGULAR,
            3 OVERTIME;
```

(3) ตัวเลขสูงสุดของ level number ต้องไม่เกิน 15

(4) รายละเอียดของ major structure name หนึ่ง ๆ จะสิ้นสุด เมื่อมีการ declare item อื่นด้วย level number 1 หรือมีการ declare item อื่นที่ไม่มี level number หรือ จบด้วยเครื่องหมาย semi colon 1 ตัว เช่นในตัวอย่าง

(5) level number ที่กำหนดให้กับ structure names จะใช้ในคำสั่ง declare เท่านั้น กรณีที่ต้องการควบคุมโครงสร้างต่าง ๆ ในคำสั่ง ALLOCATE โดยการอ้างถึง โครงสร้างหรืออีลิเม้นต์ในโครงสร้างนั้น ไม่ต้องใช้ level number

Qualified Names

minor structure หนึ่ง ๆ หรือ structure element ตัวใดตัวหนึ่งก็ตาม อาจถูกนำไปใช้โดย minor structure name หรือ elementary name ตามลำพังก็ได้ ถ้าชื่อเหล่านี้จะมีความหมายเพียงอย่างเดียวเท่านั้น แต่อย่างไรก็ตาม ในตัวอย่างข้างต้น ชื่อ REGULAR และ OVERTIME ปรากฏสองครั้งในการ declare โครงสร้างของ PAYROLL ดังนั้นการอ้างถึงชื่อ ตัวใดตัวหนึ่งนั้น อาจจะมีความหมายมากกว่า 1 อย่าง ถ้าหากว่าไม่มีการกำหนดคุณสมบัติบางอย่างให้กับชื่อนั้นเพื่อทำให้เป็นชื่อที่มีความหมายเพียงอย่างเดียว

ภาษา PL/1 เขาใช้ qualified names เพื่อหลีกเลี่ยงไม่ให้ชื่อหนึ่งมีความหมายมากกว่าหนึ่งอย่าง qualified name เป็น elementary name หรือ minor structure name ซึ่งทำให้มีความหมายเพียงอย่างเดียว (unique) โดยการกำหนดคุณสมบัติเพิ่มชื่อตั้งแต่ 1 ชื่อขึ้นไปที่มี level สูงกว่าชื่อนั้น

ตัวอย่าง

ในตัวอย่างโครงสร้าง PAYROLL, REGULAR และ OVERTIME สามารถทำให้มีความหมายอย่างเดียวไม่ซ้ำกันได้ โดยใช้ qualified names ต่อไปนี้

HOURS.REGULAR, HOURS.OVERTIME,

RATE.REGULAR, RATE.OVERTIME

ชื่อที่แตกต่างกันใน qualified name หนึ่ง ๆ ให้เขียนต่อเนื่องกันด้วยเครื่องหมาย period (.) 1 ตัว ก่อน period และ หลัง period อาจจะมี blank หรือไม่มีก็ได้ การเขียน qualification นั้นให้เขียนเรียงตามลำดับ

เช่น ให้เขียนชื่อที่มี level สูงสุดก่อนแล้วตามด้วยชื่อที่มี level ถัดไปตามลำดับและชื่อที่มี level ต่ำสุดต้องเขียนเป็นตัวสุดท้าย

ชื่อต่าง ๆ ในโครงสร้างหนึ่ง ๆ ยกเว้น major structure name ไม่จำเป็นต้องมีความหมายเดียวภายใน procedure ที่ DECLARE ตัวอย่างเช่น qualified name PAYROLL.HOURS.REGULAR ต้องการทำให้มีความหมายเดียว เนื่องจากว่าในโครงสร้างอื่น เช่นในชื่อว่า WORK ก็ใช้ชื่อ REGULAR ใน minor structure HOURS ด้วย ซึ่งทำให้ unique ได้โดยใช้ชื่อ WORK.HOURS. REGULAR

qualification จำเป็นต้องเริ่มตั้งแต่ตัวแรก จนถึงตัวสุดท้ายเพื่อทำให้ชื่อนั้นมีความหมายเดียว แต่อย่างไรก็ตาม qualified name ซึ่งอยู่ถัดจากตัวแรกและตัวกลางทั้งหมด อาจจะไม่เขียนไว้ก็ได้

2.3.3 Arrays of structures

ชื่อโครงสร้าง (structure name) ไม่ว่าจะเป็น major หรือ minor ก็ตาม อาจจะกำหนด dimension attribute ในคำสั่ง declare เพื่อเป็นการระบุว่าเป็น an array of structures ได้

An array of structures หมายถึง array หนึ่งซึ่งมีอีลิเม้นต์เป็นโครงสร้างมีชื่ออย่างเดียวกัน, levels เท่ากัน, จำนวนอีลิเม้นต์เหมือนกัน

ตัวอย่าง

โครงสร้างที่ชื่อว่า WEATHER ซึ่งใช้ในการ process ข้อเท็จจริงทางด้านอุตุนิยมวิทยาของแต่ละเดือนภายใน 1 ปี อาจจะ declare ได้ดังนี้

```
DECLARE 1 WEATHER (12),
          2 TEMPERATURE,
            3 HIGH DECIMAL FIXED (4,1),
            3 LOW DECIMAL FIXED (3,1),

          2 WIND_VELOCITY,
            3 HIGH DECIMAL FIXED (3),
            3 LOW DECIMAL FIXED (2),

          2 PRECIPITATION,
            3 TOTAL DECIMAL FIXED (3,1),
            3 AVERAGE DECIMAL FIXED (3,1);
```

ทั้งหมดที่เขียนข้างต้นเป็น array ที่ใช้แทนสภาพภูมิอากาศภายใน 1 ปี ถ้าโปรแกรมเมอร์ต้องการข้อมูลสภาพของภูมิอากาศ ในเดือนกรกฎาคมกำหนดได้ดังนี้ WEATHER (7)

ส่วนประกอบของสภาพภูมิอากาศของเดือนกรกฎาคม เรียกใช้ดังนี้ TEMPERATURE (7), WIND_VELOCITY (7), และ PRECIPITATION (7) แต่ TOTAL (7) หมายถึง ความกดอากาศทั้งหมดระหว่างเดือนกรกฎาคม

TEMPERATURE.HIGH (3) หมายถึง อุณหภูมิสูงสุดในเดือนมีนาคม ชื่อนี้เป็น subscripted qualified name

ตัวอย่าง an array of structures ซึ่งประกอบด้วย minor structures ที่เป็น array

```
DECLARE 1 A(6,6),
        2 B(5),
          3 C,
          3 D,
        2 E;
```

A และ B เป็น arrays of structures ในการเรียกใช้ data item หนึ่ง ๆ จึงต้องใช้ชื่อ 3 ชื่อ และ 3 subscripts ด้วยเช่น

A (1,1). B(2). C หมายถึงส่วนที่เป็น C เป็นอีลิเม้นต์ของ B และอยู่ในโครงสร้าง A

2.4 Other attributes

keyword attrributes สำหรับ data variables เช่นคำว่า BINARY และ DECIMAL เราได้กล่าวมาแล้วอย่างย่อ ๆ ในบทที่ 1 ส่วน attributes อื่น ๆ ซึ่งไม่ค่อยได้ใช้ในการกำหนดชนิดและการจัดระเบียบให้กับข้อมูลมีคำว่า DEFINED, LIKE, ALIGNED, UNALIGNED, และ INITIAL

2.4.1 DEFINED attribute ใช้คำย่อว่า DEF

เป็นการระบุว่าชื่อของ data element, structure, หรือ array ใช้เนื้อที่ภายในหน่วย ความจำของคอมพิวเตอร์ที่เดียวกับข้อมูลตัวอื่น

ตัวอย่าง

```
DECLARE LIST (10,10),
LIST_A (10,10) DEFINED LIST;
```

ในที่นี้ LIST เป็น array 2 มิติขนาด 10 โรว์ 10 คอลัมน์

LIST_A เป็น array เดียวกันกับ array LIST

นั่นคือ อีลิเม้นต์ แต่ละตัวใน LIST_A เป็น อีลิเม้นต์ ตัวเดียวกันกับใน array LIST นั่นเอง

attribute DEFINED ที่ใช้คู่กัน attribute POSITION หรือ POS ใช้สำหรับแบ่ง data item ออกเป็นส่วน ๆ

ตัวอย่าง

```
DECLARE LIST CHARACTER (50),
    LIST_A CHARACTER (10) DEFINED LIST,
    LIST B CHARACTER (10) DEFINED LIST
                          POSITION (11),
    LIST_C CHARACTER (30) DEFINED LIST
                          POSITION (21);
```

ในที่นี้ LIST_A หมายถึง character 10 ตัวแรกของ LIST

LIST_B หมายถึง character 10 ตัวถัดไป LIST

LIST_C หมายถึง character 30 ตัวสุดท้ายของ LIST

DEFINED attribute อาจจะใช้สำหรับระบุส่วนต่าง ๆ ของ array ได้โดยใช้คู่กับ iSUB variables เพื่อที่จะสร้าง array ใหม่ขึ้นมา

iSUB variables เป็น dummy variable

เมื่อ i หมายถึง มูลค่าคงที่จำนวนเต็มฐานสิบ 1, 2, 3, ...หรือ n

(n เป็นขนาดของมิติสำหรับ defined item)

มูลค่าของ iSUB variable จะมีขอบเขตระหว่างขอบเขตล่าง ถึง ขอบเขตบน ของ มิติที่ 1 ของ defined array

ตัวอย่าง

```
DECLARE A(20,20),
    B(10) DEFINED A (2* iSUB, 2* iSUB);
```

ในที่นี้ B เป็น subset ของ A ประกอบด้วย อีลิเม้นต์ ทั้งหมด 10 ตัว ที่อยู่ในแนวเส้นทแยงมุม (diagonal) ของ array A

นั่นคือ B(1) หมายถึง A(2,2)

B(2) หมายถึง A(4,4)

B(10) หมายถึง A(20,20)

2.4.2 LIKE attribute

ใช้เมื่อต้องการกำหนดให้ ชื่อซึ่งกำลังจะ declare มีโครงสร้างอย่างเดียวกับ major structure หรือ minor structrue name ที่ตามหลัง attribute LIKE.

ตัวอย่าง

```
DECLARE 1 BUDGET,
            2 RENT,
            2 FOOD,
              3 MEAT,
              3 EGGS,
              3 BUTTER,
            2 TRANSPORTATION,
              3 WORK,
              3 OTHER,
            2 ENTERTAINMENT,
1 COST_OF_LIVING LIKE BUDGET;
```

ในที่นี้ COST_OF_LIVING มีโครงสร้างอย่างเดียวกับ BUDGET นั่นคือในบรรทัดสุดท้าย ในตัวอย่างข้างต้นมีความหมายดังนี้

```
DECLARE 1 COST_OF_LIVING,
            2 RENT,
            2 FOOD,
              3 MEAT,
              3 EGGS,
              3 BUTTER,
            2 TRANSPORTATION,
              3 WORK,
              3 OTHER,
            2 ENTERTAINMENT;
```

ข้อสังเกต

LIKE attribute เพียงแต่ copy โครงสร้าง ชื่อ และ attribute ของ โครงสร้าง ที่อยู่ใน level ที่ต่ำกว่าชื่อที่กำหนดเท่านั้น แต่ไม่ copy dimension ของชื่อนั้น

ตัวอย่างเช่น ถ้าเรากำหนด BUDGET ใหม่โดย DECLARE 1 BUDGET (12) เมื่อ DECLARE COST_OF_LIVING LIKE BUDGET เครื่องคอมพิวเตอร์จะไม่กำหนด dimension attribute ให้กับ COST_OF_LIVING แต่ถ้าต้องการให้ COST_OF_LIVING มี dimension ต้องเขียนดังนี้

1 COST_OF_LIVING (12) LIKE BUDGET;

ส่วน a minor structure name อาจจะ declare LIKE major structure name หรือ LIKE minor structure name อื่น ๆ ก็ได้ และในทำนองเดียวกัน major structure name หนึ่ง ๆ ก็อาจจะ declare LIKE minor structure หรือ LIKE major structure name อื่น ๆ ได้เช่นกัน

ตัวอย่าง

บริษัทขายต้น, เมล็ดพันธุ์ ของดอกไม้และผักแห่งหนึ่งเปิดบริการให้สินเชื่อแก่ลูกค้า โดยลูกค้าต้องส่ง ชื่อลูกค้า ที่อยู่ รายการจำนวนของที่สั่งซื้อ โปรแกรมข้างล่างนี้ ตรวจสอบว่าจำนวนเงินที่บริษัทให้สินเชื่อแก่ลูกค้านั้น เพียงพอที่จะจ่ายให้ครบทุกรายการที่สั่งหรือไม่ ถ้าไม่พอ ให้เจ้าหน้าที่ติดต่อกลับไปยังลูกค้า โดยจะยังไม่ส่งของให้

กำหนดราคาสินค้าไว้ดังนี้

เมล็ดดอกไม้	ราคากรัมละ	20 บาท
เมล็ดผัก	ราคากรัมละ	20 บาท
ดอกไม้	ราคาต้นละ	10 บาท
ผัก	ราคาต้นละ	2 บาท

```
GARDEN :   PROCEDURE OPTIONS (MAIN);
           DCL 1 CUSTOMER
                 2 NAME CHAR (20) VAR,
                 2 ADDRESS CHAR (40),
                 2 CREDIT FIXED DEC (6,2),
                 2 ORDER,
                    3 SEEDS,
                    (4 FLOWER,
                    4 VEGETABLE) FIXED DEC (4),
                    3 PLANTS LIKE SEEDS;
           DCL COST FIXED DEC (6,2);
           GET LIST (CUSTOMER);
```

```
COST = (SEEDS. FLOWER + SEEDS. VEGETABLE) *20
+ PLANTS. FLOWER * 10 + PLANTS.VEGETABLE *2;
IF COST > CREDIT THEN
        PUT SKIP LIST ('THE ORDER OF', NAME, 'IS
        NOT TO BE FILLED');
ELSE PUT SKIP LIST ('THE ORDER OF', NAME, 'IS O.K');
END GARDEN;
```

2.4.3 INITIAL attribute

ใช้สำหรับกำหนดมูลค่าเริ่มต้น ให้กับ ตัวแปร หนึ่ง ๆ ขณะที่มีการจัดสรรเนื้อที่นั้น

ตัวอย่าง 1

```
DECLARE NAME CHARACTER (10) INIT ('JOHN DOE');
DECLARE PT FIXED DEC (5,4) INIT (3.1416);
DECLARE A INIT ((B*C));
DECLARE B INIT (SQRT (7));
```

เมื่อคอมพิวเตอร์จัดสรรเนื้อที่ในหน่วยความจำให้กับ NAME มันก็จะกำหนด character string 'JOHN DOE' ให้แต่เนื่องจากไม่ครบ 10 ตัวเครื่อง จะให้ blank ทางขวามือจนครบเนื้อที่นั้น

ตัวอย่าง 2

```
DECLARE X BIN FIXED (15,0) INIT (23);
DECLARE Y BIN FIXED (15) INIT (10111B);
```

ตัวอย่าง 3

```
DCL TAB(5) FIXED DEC (5,2) INIT (0);
```

ในที่นี้เฉพาะ TAB(1) เท่านั้นที่มีมูลค่าเป็น 0

```
DCL TAB(5) FIXED DEC (5,2) INIT (0,0,0,0,0);
DCL TAB(5) FIXED DEC (5,2) INIT ((5)0);
DCL TAB(5) FIXED DEC (5,2) INIT (0,*,*,5,5);
DCL TAB(5) FIXED DEC (5,2) INIT ((3)0, (2)5);
```

ตัวอย่าง 4

```
DCL (A,B,C) FIXED DEC (5,2) INIT (-1.2);
```

หมายความว่า A=-1.2

B=-1.2

C=-1.2

ข้อสังเกต

(1) INITIAL attribute จะกำหนดให้กับ entry constants, file constatns, DEFINED data, entire structure, หรือ parameters ไม่ได้ (ยกเว้น CONTROLLED parameters)

(2) INITIAL attribute อาจจะกำหนดให้ arrays ก็ได้ เช่นเดียวกับ element variables ส่วนการ declare structure เฉพาะ elementary names เท่านั้นที่กำหนด INITIAL attribute ได้

(3) สำหรับ array หรือ array of structure หนึ่งๆ อาจจะกำหนดเริ่มต้นเพียงบางส่วนหรือครบทุกตัว สำหรับบางตัวที่ไม่กำหนดมูลค่าให้ใน INITIAL attribute อาจจะใช้เครื่องหมาย asterisk บอกก็ได้

ตัวอย่าง

```
DECLARE A(15) CHARACTER (13) INITIAL
    ('JOHN DOE',*,
'RICHARD ROW'
'MARY SMITH') VARYING,
        B(10,10) DECIMAL FIXED (5)
            INITIAL ((25)0, (25)1, (50)0),
        1C(8),
            2 D INITIAL (0),
            2 E INITIAL ((8)0);
```

ในที่นี้ เฉพาะ อีลิเม้นต์ ตัวแรก, ตัวที่ 3, และตัวที่ 4 ของ array A เท่านั้นที่กำหนดมูลค่าให้ส่วนที่เหลือไม่ได้กำหนดมูลค่าให้, แต่ อีลิเม้นต์ ของ array B กำหนดมูลค่าให้ทุกตัว 25 ตัวแรกมีมูลค่าเท่ากับ 0, 25 ตัวถัดไปมีมูลค่า เท่ากับ 1 และ 50 ตัวสุดท้าย มีมูลค่า เท่ากับ 0 (ตัวเลข 25, 25, และ 50 ที่อยู่ภายในเครื่องหมายวงเล็บเรียกว่า iteration factors) ส่วนโครงสร้าง C เมื่อใช้ dimension (8) อีลิเม้นต์ ตัวแรกของ D เท่านั้นที่มีค่าเป็น 0, แต่ อีลิเม้นต์ ของ E ทั้ง 8 ตัว มีค่าเป็น 0 หมดทุกตัว

ตัวอย่าง 3

```
DECLARE 1 G,
        2 H INITIAL (0),
        2 I INITIAL (0),
        1 J(8) LIKE G;
```

ในที่นี้ เฉพาะ J(1).H และ J(1).I เท่านั้นที่มูลค่าเป็น 0

ตัวอย่าง

```
DECLARE T(50) CHARACTER (10)
    INITIAL ((10) 'A',(25) (10) 'B',
    (24) (1) 'C');
```

ในตัวอย่างนี้ INITIAL attribute specification มีทั้ง iteration factors (เลข 25 และ=24) และ repetition factors (เลข 10,10, และ 1)

คำสั่งข้างต้นมีความหมายว่า

อีลิเม้นต์ ตัวแรกของ array T คือ 'AAAAAAAAAA' (string ของ A สิบตัว)

อีลิเม้นต์ อีก 25 ตัวของ array T แต่ละตัวคือ 'BBBBBBBBBB' (string ของ B สิบตัว)

และอีลิเม้นต์ อีก 24 ตัว สุดท้ายของ array T คือ 'C' (string ของ C หนึ่งตัว)