บทที่ 5
Data attributes and declare statement

นอกจากค่าคงที่ และชื่อ บิลท์-อิน ฟังก์ชันแล้ว data items ทุกตัวที่อยู่ในโปรแกรม PL/I-80 จะต้อง explicitly defined โดยใช้คำสั่ง DECLARE และ data attributes

5.1 คำสั่ง DECLARE

ชื่อตัวแปรทุกตัวในโปรแกรม ซึ่งไม่ใช่ชื่อ บิลท์-อิน ฟังก์ชัน หรือ ตัวแปรเทียม (pseudo variable) ต้องนิยาม (defined) ในคำสั่ง DECLARE, ค่าคงที่แฟ้มข้อมูล (file constant) และ ตัวแปรแฟ้มข้อมูล (file variable) ก็เช่นเดียวกัน ต้องนิยาม ในคำสั่ง DECLARE, สำหรับ control constant เช่น statement label และ procedure name ถูก declare อย่าง implicitly โดยการใช้ในตัวโปรแกรม

คุณลักษณะ (properties) ที่กำหนดให้กับชื่อ ซึ่งบอกให้รู้ว่า มันเป็น data item ตัวหนึ่ง เรียกว่า attribute ของชื่อนั้น, attribute อาจจะ นิยาม implicitly โดยการ default หรือ นิยาม explicitly โดยการกำหนดคุณลักษณะเฉพาะของมัน ในคำสั่ง DECLARE

รูปแบบทั่วไป ของคำสั่ง DECLARE

```
DECLARE DCL [level] name [attribute-list] ,,,
    [,[level] name  [attribute--list]];
```

เมื่อ name เป็น variable identifier

level เป็น เลขจำนวนเต็มบวก ใช้ในการ declare โครงสร้าง ได้อธิบายไว้แล้วในบทที่ 4

attribute-list หมายถึง อธิบายลักษณะเฉพาะของชื่อนั้น ถ้าไม่มี attribute-list ในคำสั่ง DECLARE เครื่องจะ default attribute ให้เป็น FIXED BINARY(15)

ตัวอย่าง คำสั่ง DECLARE

```
DECLARE A CHARACTER(Z) BASED,
        B FIXED BINARY INITIAL(0),
        C(100) FIXED DECIMAL(5,2);
DCL     1 BOOK,
          2 TITLE     CHARACTER(20),
          2 AUTHOR,
            3 LASTNM  CHAR(10),
            3 FIRSTNM CHAR(10),
          2 PUBLISHER CHAR(20);
```

ตัวอย่าง การ declare อย่างง่ายของข้อมูลแต่ละชนิดและกลุ่มข้อมูล ได้กล่าวแล้วในบทที่ 3 และบทที่ 4

ขอบเขต (scope) ของชื่อที่ declare คือเขต (region) ของโปรแกรมซึ่ง ชื่อ และ attribute ของมัน ทราบค่า โดยทั่วไปแล้ว การเกิดชื่อ ในคำสั่ง DECLARE จะ implicitly define ขอบเขต ของ ชื่อที่ declare ให้เป็น internal นั่นคือ block ซึ่งมีการ declare เกิดขึ้น ถ้าชื่อนั้นมี EXTERNAL attribute ขอบเขตของมันจะรวมทก block ซึ่งมันได้ถูก declare ด้วย EXTERNAL attribute

ถ้าตัวแปรใดก็ตาม ถูก declare ด้วย EXTERNAL attribute จะต้องมี STATIC attribute อยู่ด้วย เนื่องจาก ชื่อจะเกี่ยวข้องกับเนื้อที่ในหน่วยความจำ 1 แห่ง แต่ละชื่อ จะ declare มากกว่า 1 ครั้ง ภายใน block เดียวกันไม่ได้

ยกเว้น ชื่อของสมาชิกของ โครงสร้าง โดยวิธีนี้ การอ้างถึงอย่างชัดเจน จะต้องมี การกำหนดให้ ชื่อแต่ละตัว ชื่อแต่ละชื่อ ต้องไม่ declare attribute ที่ขัดแย้งกัน ใน 2 blocks ใด ๆ ที่มีมี EXTERNAL attribute

ในภาษา PL/I-80 เพื่อความสะดวกและทำให้ง่าย เราอาจใช้รูปแบบอื่น ของ คำสั่ง DECLARE ได้

รูปทั่วไป ชุดของคำสั่ง DECLARE จะเป็นดังนี้

```
DECLARE definition-l;
DECLARE definition-2;

DECLARE definition-n;
```

ซึ่งจะมีความหมายเหมือนกับ

```
DECLARE definition-l, definition-2, ... definition-n;
```

เมื่อ definition แต่ละตัว คั่นด้วยเครื่องหมาย comma และอาจจะมี blank หรือไม่มีก็ได้ การจบคำสั่ง DECLARE ให้ใส่เครื่องหมาย semicolon 1 ตัว นอกจากนี้ แล้ว attributes ที่ใช้ร่วมกัน (shared) โดย item definitions หลาย ๆ ตัว สามารถจัดไปไว้ทางขวา นั่นคือ ชุดของ definition ในรูปแบบข้างล่างนี้

```
item-l attr-A, item-2 attr-A, . . . item-n attr-A
```

เขียนในรูป factored form ที่มีความหมายเหมือนกัน จะเป็นดังนี้

```
(item--l, item-2, . . . item-n) attr-A
```

เราสามารถใช้กฎนี้ ซ้ำได้เช่นเดียวกัน

ตัวอย่าง

```
DECLARE ((A,B) FIXED BINARY, C FLOAT BINARY) STATIC EXTERNAL;
```

ซึ่งมีความหมายเหมือนกับ

```
DECLARE A FIXED BINARY STATIC EXTERNAL,
        B FIXED BINARY STATIC EXTERNAL,
        C FLOAT BINARY STATIC EXTERNAL;
```

โดยทั่วไปแล้ว การเรียงลำดับของ attributes ไม่สำคัญ ยกเว้นเฉพาะ dimension list attribute สำหรับอะเรย์ ซึ่งต้องอยู่หลัง ชื่ออะเรย์ และอยู่ก่อน attributes อื่น และเลขบอกระดับของสมาชิกของโครงสร้าง ต้องอยู่หน้าชื่อสมาชิก, attributes ทั้งสองตัวนี้เอามาจัดพวกได้ เนื่องจาก dimension list จะต้องอยู่ทางขวามือชื่อที่มันจะมีผล (apply) จึงต้องจัดให้อยู่ทางขวา แต่ เลขบอกระดับ อยู่ก่อนหน้าชื่อสมาชิก เมื่อมีการจัดพวกจึงต้องไว้ทางซ้ายมือ ดังนี้

```
level-k item-l, level-k item-2, . . . level-k item-n
```

จะมีความหมายเหมือนกับ

```
level-k (item-l, item-2, . . . . item-n)
```

<u>ตัวอย่าง</u>

```
DECLARE 1 A BASED, 2 (B FIXED BINARY, C CHARACTER(Z));
```

จะมีความหมายเหมือนกับ

```
DECLARE 1 A BASED,
        2 B FIXED BINARY,
        2 C CHARACTER(Z);
```

attribute list แต่ละชุด จะต้องไม่ประกอบด้วย attribute ที่ขัดแย้งกันเอง เช่น มีข้อมูลอยู่ 2 ชนิด หรือ storage class attribute 2 ชุด และอาจจะไม่ต้องเขียน set ของ attribute ให้สมบูรณ์ก็ได้ นั่นคือ เพียง attribute ที่พอบอกลักษณะเฉพาะของข้อมูลนั้น ในกรณีนี้ คอมไพเลอร์ของ PL/I-80 จะจัด attribute ให้ โดยยึดกฎการ default ดังนี้

1) ถ้าไม่กำหนด attribute เครื่องจะจัด FIXED BINARY(15) ให้

2) ถ้ากำหนด DECIMAL หรือ BINARY แต่ไม่มี FIXED หรือ FLOAT เครื่องจะจัด FIXED ให้

3) ถ้ากำหนด FIXED หรือ FLOAT แต่ไม่มี BINARY หรือ DECIMAL เครื่องจะจัด BINARY ให้

4) ถ้าไม่กำหนด precision ให้กับ FIXED BINARY เครื่องจะจัด FIXED BINARY(15) ให้

5) ถ้าไม่กำหนด precision ให้กับ FIXED DECIMAL เครื่องจะจัด FIXED DECIMAL(7,0) ให้

6) ถ้าไม่กำหนด precision ให้กับ FLOAT BINARY เครื่องจะจัด FLOAT BINARY(24) ให้

5.2 List of data attributes

พารากราฟต่อไปนี้ จะกล่าวถึง attributes ต่าง ๆ ที่ใช้กับข้อมูลในโปรแกรม attribute บางตัว ใช้คำย่อได้ รายละเอียดที่สมบูรณ์ของ attributes ให้ดูในหัวข้อที่เกี่ยวข้องกัน

ALIGNED เป็น data attribute ซึ่งปกติจะทำให้เกิด storage boundary alignment ของตัวแปรขึ้น แต่ attribute ตัวนี้ ใช้ไม่ได้ใน PL/I-80

AUTOMATIC เป็น storage class attribute หมายถึงเนื้อที่หน่วยความจำ ได้ถูกจัดสรรให้กับตัวแปร ภายใต้การปฏิบัติการ ของ block ที่มีการ declare ใน PL/I-80 เนื้อที่หน่วยความจำอัตโนมัติ จะถูกจัดสรรแบบ static ยกเว้นในเรื่อง recursion

BASED หรือ BASED(p) หรือ BASED(q()) เป็น storage class attribute
หมายถึง การจัดสรรเนื้อที่ ให้กับตัวแปร ซึ่งเป็น user-controlled
ในกรณีนี้ p จะเป็น pointer variable และ q เป็น pointer
valued function

BINARY หรือ BINARY(p) และ BIN หรือ BIN(p) เป็นการ นิยาม ตัวแปร
ชนิด BINARY ให้ precision เป็น p

BIT(n) เป็นการ นิยาม bit string ความยาวเท่ากับ n

BUILTIN หมายถึงชื่อที่กำลังถูก declare นี้ เป็น บิลท์-อิน ฟังก์ชัน
ของภาษา PL/I-80 ถ้าชื่อของ บิลท์-อิน ฟังก์ชัน มีการ
นิยามใหม่ ใน block อื่น ดังนั้น การอ้างถึง บิลท์-อิน
ฟังก์ชัน ใน contained block ชื่อของบิลท์-อิน ฟังก์ชัน
ต้อง นิยามใหม่ ด้วย attribute BUILTIN ใน
contained block

CHARACTER(n) และ CHAR(n) เป็นการ นิยาม character
string ความยาวเท่ากับ n

DECIMAL [(p[,q])] และ DEC [(p[,q])] เป็นการ นิยามเลข
ชนิด DECIMAL ให้ precision เป็น (p,q)
ถ้าไม่กำหนด q เครื่องจะถือว่ามีค่าเป็นศูนย์ สำหรับ
default precision คือ (7,0)

ENTRY [(att-1, att-2, ... att-n)] เป็นการ นิยามค่า
เอนทรี (ENTRY value) เมื่อ att-1 ถึง att-n
เป็น attribute list ของพารามิเตอร์ ซึ่งกำหนดใน
definition PROCEDURE ของค่าเอนทรี

ENVIRONMENT (options) หรือ ENV (options) เป็นการ นิยาม แฟ้มข้อมูลชนิด fixed และ variable length RECORD เมื่อ option อาจจะเป็นตัวใดตัวหนึ่ง ข้างล่างนี้

F(lrecl)

B(lbuff)

F(lrecl),B(lbuff)

expression lrecl เป็นความยาวเรคคอร์ด ของแฟ้มข้อมูลชนิด fixed. length record และ lbuff เป็นขนาดของบัฟเฟอร์ (system buffer size)

EXTERNAL หรือ EXT เป็น นยาม ขอบเขตของ item ที่ declare แล้วให้เป็น EXTERNAL, นั่นคือ item นั้นสามารถรู้ค่าได้ ในทกนิยาม ของ block ซึ่ง declare มันเป็น EXTERNAL

FILE เป็นการ นิยาม file data, สำหรับ รายละเอียดของ attribute ตัวนี้ให้ดูบทที่ 10

FIXED [(p[,q])] เป็นการ นิยาม ข้อมูลชนิด fixed-point arithmetic ให้ precision เป็น (p,q)

FLOAT [(p)] เป็นการ นิยาม ข้อมูลชนิด floating-point arithmetic ให้ precision เป็น p

INITIAL (value-list) และ INIT (value-list) เป็นการให้ค่าเริ่มต้น กับตัวแปร ชนด STATIC กอนการ execute โปรแกรม, value-list เป็นชดของค่าคงที่ แต่ละตัวคั่นด้วยเครื่องหมาย comma ซึ่งจะถูกเปลี่ยนรูป (converted) ให้เป็น ชนิดตัวแปร ขณะเริ่มต้น ค่าคงที่แต่ละตัว ในชุด (list) นั้น อาจจะมี repetition factor ซึ่งอยู่ในวงเล็บ อยู่ข้างหน้ามันได้

LABEL เป็นการ นิยาม ตัวแปรชนิด LABEL

POINTER เป็นการ นิยาม ตัวแปรชนิด พอยท์เตอร์

RETURNS (attribute-list) จะถูกกำหนดให้กับ attribute ENTRY เพื่อบอกลักษณะ attribute-list ของค่าที่ส่งกลับมา โดยฟังก์ชัน

STATIC เป็น storage class attribute ซึ่งทำให้ มีการจัดสรรเนื้อที่หน่วยความจำ ก่อนการ execute โปรแกรม

VARIABLE ใช้กับ attribute FILE หรือ ENTRY เพื่อ นิยาม item นั้นให้เป็นตัวแปร และ ไม่เป็นค่าคงที่แฟ้มข้อมูล หรือค่าคงที่เอนทรี

VARYING และ VAR เป็นการ นิยาม ความยาวของ character string ให้ไม่คงที่ (vary)