บทที่ 6
การจัดสรรเนื้อที่หน่วยความจำ
(Storage management)

ภาษา PI/I มีระบบควบคุม การจัดสรรหน่วยความจำ สำหรับเนื้อที่เก็บข้อมูล, หน่วยความจำนั้นอาจจะมีการจัดสรร แบบคงที่ (static) ขณะคอมไพล์ โปรแกรม หรือ อาจมีการจัดสรรแบบ ไม่คงที่ (dynamic) ระหว่างการ execute โปรแกรม, การจัดสรรหน่วยความจำแบบไม่คงที่ อาจจะ ยกเลิกได้สำหรับ การใช้ใหม่ (re-use) ตัวแปรทุกตัวในโปรแกรม มี storage class attribute ขั้นหน่วยความจำ (storage class) จะเป็นตัวบ่งบอกว่า จะมีการจัดสรรหน่วยความจำสำหรับตัวแปรนั้น เมื่อไร และอย่างไร

PL/I-80 แบ่งขั้นหน่วยความจำ เป็น 3 ชนิดดังนี้

STATIC

AUTOMATIC

BASED

เพื่อปรับปรุง กลไก การกำหนดตำแหน่งที่อยู่ภายในหน่วยความจำ, หน่วยความจำแบบ AUTOMATIC จะถูกปฏิบัติ ในลักษณะเดียวกับ หน่วยความจำแบบ STATIC ยกเว้นใน procedure ซึ่งถูกกำหนดเป็น RECURSIVE, storage class attribute เป็นคุณสมบัติ (properties) ของ อีลิเม้นท์, อะเรย์ และ ตัวแปรชนิดโครงสร้างหลัก

attribute เหล่านี้ กำหนดให้กับ ชื่อ entry, ชื่อแฟ้มข้อมูล หรือสมาชิกของกลุ่มข้อมูล ไม่ได้

6.1 STATIC attribute

ตัวแปร ซึ่ง declare ด้วย attribute STATIC จะถูกจัดสรร ก่อนการ execute main procedure ตัวแปรที่เป็นชนิดแบบคงที่ อาจมีค่าเริ่มต้นได้

ตัวอย่าง STATIC attribute

```
DECLARE A FIXED STATIC;
DECLARE B(100) CHAR(Z) STATIC;
DECLARE 1 C STATIC,
          2 D CHAR(10),
          2 E FIXED;
```

6.2 INITIAL attribute

attribute INITIAL ใช้เพื่อกำหนดค่าคงที่เริ่มต้น ให้กับ data item ในการจัดสรรหน่วยความจำ ข้อมูลแรกนี้จะเป็น ส่วนหนึ่งของ program module ซึ่งจะถูกโหลด (load) เมื่อเริ่มมีการ execute โปรแกรม

รูปแบบของ INITIAL attribute มีดังนี้

INITIAL | INIT (value[,value] ...)

เมื่อ ค่าเริ่มต้นอยู่ในรูปแบบ

[(iteration-factor)] constant-exp

iteration-factor เป็นค่าคงที่ แทนการนับซ้ำ ซึ่งจะทำให้ค่า constant-exp มีเท่ากับจำนวนเลขนั้น

constant-exp ต้องเป็นค่าคงที่ literal ซึ่งเข้าได้กับ ชนิดข้อมูล ขณะเริ่มต้น ประกอบด้วย ค่าคงที่เลขคณิต อาจจะมีเครื่องหมายกำกับข้างหน้าหรือไม่ก็ได้, ค่าคงที่ string หรือค่า NULL พอยท์เตอร์, data items ของอะเรย์ สามารถกำหนดค่าเริ่มต้นด้วยคำสั่งหนึ่งคำสั่ง ในกรณีนี้ ต้องเริ่มต้นด้วย อีลิเม้นท์ตัวแรกของอะเรย์ และต่อ ๆ ไป ในลักษณะทีละโรว์ (row major order) จนกระทั่งจบเซทของค่าคงที่เริ่มต้น ซึ่งต้องไม่มากกว่าขนาดเริ่มต้นของอะเรย์, สมาชิกของโครงสร้างจะต้องถูกกำหนดค่าเริ่มต้นของใครของมัน การกำหนดค่าคงที่ให้กับตัวแปร ให้ใช้กฎของคำสั่ง assignment เครื่องจะใส่ เครื่องหมาย blank เพิ่มทางขวามือ เมื่อการ

กำหนด string นั้น สั้นกว่าตัวแปร ชนิด string

ในที่สุด เฉพาะตัวแปร STATIC เท่านั้น ที่สามารถมี INITIAL attribute เพื่อให้เข้ากันได้ (compatible) กับ ภาษา PL/I ชุดสมบูรณ์

ตัวอย่าง

```
DECLARE A FIXED BINARY STATIC INITIAL (0);
DECLARE B(8) CHARACTER(Z) INITIAL ((8)'AB') STATIC;
DECLARE
        1 FCB STATIC,
          2 FDRIVE FIXED(7) INITIAL (0),
          2 FNAME CHAR(8) INITIAL ('EMO'),
          2 FTYPE CHAR(3) INITIAL ('DAT'),
          2 FEXT BIT(8) INITIAL ('00'B4),
          2 FFILL(19) BIT(8);
```

6.3 AUTOMATIC attribute

ปกติ attribute AUTOMATIC ทำให้มีการจัดสรรเนื้อที่หน่วยความจำของข้อมูล ภายใต้ entry ของ PROCEDURE block หรือ BEGIN block ซึ่งตัวแปรนั้นปรากฏอยู่ ใน PL/I-80 หน่วยความจำ AUTOMATIC เป็นการจัดสรรเนื้อที่แบบคงที่เพื่อปรับปรุง ตำแหน่งที่อยู่ของตัวแปร และความเร็วในการ execute มีข้อยกเว้นหนึ่งอย่างคือ ใน presence ของ recursion โดยที่ ตัวแปร AUTOMATIC ต้องใช้กลไกหน่วยความจำ แบบไม่คงที่ เพื่อป้องกัน ข้อมูล overwrite บน recursive call

หมายเหตุ

default storage class attribute สำหรับตัวแปร ที่ไม่ declare เป็น STATIC หรือ BASED คือ AUTOMATIC ถ้าไม่ได้กำหนดเป็นอย่างอื่น

6.4 BASED attribute

ตัวแปรที่ declare ด้วย attribute BASED เป็นการกำหนดหน่วยความจำอย่าง explicitly ผ่านคำสั่ง ALLOCATE เมื่อใดก็ตามที่ตัวแปร BASED ถูกจัดสรร, ตัวแปรพอยท์เตอร์ที่สมนัยกันจะถูกกำหนดให้กับตำแหน่งที่อยู่ ของ ตัวแปร BASE ที่จัดสรรนั้น

รูปแบบของ BASED attribute

```
BASED   [(pointer-ref)]
```

เมื่อ pointer-ref เป็นตัวแปรพอยท์เตอร์ที่ไม่มี subscript หรือ function call ที่ไม่มี arguments ซึ่ง return ค่าพอยท์เตอร์

ถ้าตัวแปรใดก็ตาม มีการ declare ด้วย attribute BASED แต่ไม่มี pointer-ref ในการอ้างถึงตัวแปรแต่ละครั้ง ต้องมี pointer-qualifier ดังนี้

```
pointer-exp -> variable
```

เมื่อ pointer-exp เป็น pointer-valued expression, เพื่อบอกที่อยู่ของหน่วยความจำสำหรับการอ้างถึงตัวแปรนั้น ถ้ามี pointer-ref มันจะเป็นชนิด implicitly เช่นเดียวกับ เมื่อใดก็ตามที่อ้างถึงตัวแปร โดยไม่มี pointer-qualifier ในกรณีนี้ pointer-ref เป็น re-evaluated ทุกครั้งของการเกิด unqualified variable ตัวแปรพอยท์เตอร์ หรือ pointer-valued function name ที่อยู่ใน pointer-ref ถูกนำมาจาก ขอบเขตการ declare BASED ถ้ามี การ declare local มากกว่ายังคงอยู่ด้วย pointer-ref name เดียวกัน

ตัวอย่าง การ declare ตัวแปร BASED

```
DECLARE  A  CHAR(8)  BASED;

DECLARE  B  POINTER  BASED(Q);

DECLARE  C  FIXED  BASED(P);

DECLARE  D  BIT(8)  BASED(F());
```

6.5 คำสั่ง ALLOCATE

คำสั่ง ALLOCATE ใช้จัดเนื้อที่หน่วยความจำ ให้กับ ตัวแปร BASED โดยมีรูปแบบดังนี้

```
ALLOCATE based-variable SET (pointer-variable);
```

เนื้อที่หน่วยความจำส่วนหนึ่ง ได้มาจากเนื้อที่หน่วยความจำแบบไม่คงที่ ซึ่งขนาดใหญ่พอที่จะเก็บค่าของตัวแปร based ถ้าเนื้อที่ขนาดที่ต้องการใช้ไม่มี จะเกิดเงื่อนไข ERROR ขึ้น, based-variable ต้องเป็น variable reference ที่ไม่มี subscript เมื่อตัวแปรนั้น ได้ declare ไว้ในขอบเขต (scope) ของคำสั่ง ALLOCATE พร้อมกับ attribute BASED แอดเดรส ซึ่งเครื่องจัดให้ (allocation address) จะถูกเก็บในชื่อของ ตัวแปรพอยท์เตอร์ ใน SET clause ข้อสำคัญที่ควรจะสังเกตคือ การจัดหน่วยความจำในลักษณะนี้ ยังคงอยู่จนกว่า จะมีปฏิบัติการ FREE ที่สมนัยเกิดขึ้น การใช้แอดเดรสที่เครื่องจัดเนื้อที่เก็บโดย ตัวแปรพอยท์เตอร์ ทำหน้าที่เช่นเดียวกับตัวถูกกระทำ (operand)

ตัวอย่าง การใช้ attribute BASED กับคำสั่ง ALLOCATE

```
DECLARE
    (P,Q) POINTER,
    X CHARAXTER(2) BASED,
    Y FIXED BINARY BASED(P);
      :
    ALLOCATE X SET(Q);
    ALLOCATE Y SET(P);
      :
    Q -> X = 'AB';
    Y = Y + 1;
```

6.6 Null บิลท์-อิน ฟังก์ชัน

ฟังก์ชันนี้ ให้ผลลัพธ์เป็นค่าพอยท์เตอร์ ซึ่งเป็น non-valid storage address ที่มีความหมายเดียว, แอดเดรส (address) ที่มีความหมายเดียวนี้ มีประโยชน์ ทำให้ค่าพอยท์เตอร์ต่าง ๆ ว่าง (empty) โดยเฉพาะที่ใช้ ในการสร้าง linked list เพื่อบอกให้ทราบว่า เป็น อีลิเม้นท์ตัวสุดท้ายของ list นั้น รูปแบบของ ฟังก์ชัน มี 2 ชนิดดังนี้

NULL และ NULL()

หมายเหตุ ค่าพอยท์เตอร์ ไม่จำเป็นต้องเริ่มต้นด้วย ค่า NULL เมื่อมีการเริ่มทำโปรแกรม ยกเว้นในกรณีค่า NULL ใช้ใน option INITIAL ในการ declare ตัวแปร การใช้ตัวแปร BASED และฟังก์ชัน NULL ได้กล่าวไว้อย่างละเอียดในหนังสือ "PL/I Application Guide"

6.7 ADDR บิลท์-อิน ฟังก์ชัน

ฟังก์ชันตัวนี้ ให้ผลลัพธ์เป็นค่าพอยท์เตอร์ บอกแอดเดรสของเนื้อที่หน่วยความจำที่ใช้เก็บชื่อตัวแปร (เป็น argument) โดยมีรูปแบบดังนี้

ADDR (variable-name)

หมายเหตุ variable-name ต้องมีแอดเดรสในหน่วยความจำที่เราเป็นผู้กำหนด และจะไม่เป็นผลลัพธ์ที่สร้างชั่วคราว ผ่าน application ของฟังก์ชัน และตัวปฏิบัติการ

การใช้ตัวแปร BASED ร่วมกับ บิลท์-อิน ฟังก์ชัน ADDR ทำให้มีการ share เนื้อที่หน่วยความจำ ใน PL/I-80 ในกรณีนี้ ตัวแปร based ต้องไม่เป็น explicitly given storage ด้วยคำสั่ง ALLOCATE แต่จะทำหน้าที่เป็น template ซึ่ง overlays กับตัวแปรที่มีอยู่, pointer base สำหรับตัวแปร based ถูกกำหนดให้กับ แอดเดรสของตัวแปรที่มีอยู่โดยใช้ฟังก์ชัน ADDR การเข้าถึงตัวแปร based คือ การเข้าถึง overlayed variable

ตัวอย่าง บางส่วนของโปรแกรมแสดงการ share หน่วยความจำ

```
DCL
    I FIXED,
    P  POINTER,
    A CHAR(8),
    B(8)  BIT(8)  BASED(P);
    P = ADDR(A);
    GET LIST(A);
    PUT EDIT ((B(I) DO I =1 TO 3))(B4(2));
```

หมายถึง ค่าของ character string ถูก overlay ด้วย bit string vector, output จากโปรแกรมนี้ คือ ค่าของ character string เขียนในรูปแบบของ hexadecimal hit string

6.8 คำสั่ง FREE

ตัวแปร BASED จะยังคงมีการจัดเนื้อที่หน่วยความจำอยู่ จนกว่าจะยกเลิกด้วยคำสั่ง FREE รูปแบบของคำสั่ง FREE มีดังนี้

```
FREE [pointer-variable -> ] based-variable;
```

เมื่อ pointer-variable คือแอดเดรส การจัดเนื้อที่ของหน่วยความจำ ซึ่งต้องมีมาก่อนแล้วจาก เนื้อที่หน่วยความจำแบบไม่คงที่ โดยใช้คำสั่ง ALLOCATE ถ้าไม่ได้กำหนด pointer-variable ในคำสั่ง FREE, based-variable ต้องถูก declare ด้วย pointer-ref option ในกรณีนี้ หน่วยความจำมีการกำหนดแอดเดรสด้วย pointer-ref กลับไปยัง เนื้อที่หน่วยความจำแบบไม่คงที่, runtime subroutines ซึ่งยังคงอยู่ในเนื้อที่หน่วยความจำแบบไม่คงที่จะเป็น coalesce contiguous storage segments โดย

อัตโนมัติ เช่นเดียวกับมัน ถูกยกเลิก ผ่านคำสั่ง FREE

ตัวอย่าง คำสั่ง FREE ในส่วนของ non-functional program

```
DECLARE
    (P,Q,R) POINTER,
    A CHARACTER (10) BASED,
    B FIXED BASED (R) ;

    ALLOCATE A SET(P);
    ALLOCATE B SET(R);
    ALLOCATE A SET(Q) ;

    FREE P -> A;
    FREE Q -> A;
    FREE B;
```