บทที่ 4 ชนิดของคำสั่ง (Statement Classification)

ในบทนี้เราแบ่งคำสั่งในภาษา PL/1 ออกเป็นชนิดตามหน้าที่ของมัน รวมทั้งอธิบายวัตถุประสงค์ของแต่ละคำสั่ง และตัวอย่างการใช้คำสั่งนั้น ๆ ส่วนรายละเอียดเกี่ยวกับรูปแบบและกฎเกณฑ์ต่าง ๆ ให้ไปดูในภาคผนวก section F

คำสั่งต่าง ๆ สามารถจัดกลุ่มได้ 10 ชนิดดังนี้

Descriptive statement

Input/Output statement

Data Movement and computational statement

Program organization statement

Stroage Control statement

Control statement

Exception control statement

Listing Control statement

Diagnostic statement

การเลือกชื่อให้กับชนิดต่าง ๆ นั้น ตั้งเพื่อให้อธิบายวัตถุประสงค์โดยเฉพาะ ยกเว้นคำสั่ง preprocessor และคำสั่ง Listing control ซึ่งสองกลุ่มนี้ไม่มีนัยสำคัญพื้นฐานในเรื่องภาษา คำสั่งหนึ่ง ๆ อาจจะอยู่ในกลุ่มมากกว่าหนึ่งชนิดก็ได้ ถ้าหากว่าคำสั่งนั้นมีหน้าที่มากกว่าหนึ่งอย่าง

4.1 Descriptive Statement

เมื่อเครื่องคอมพิวเตอร์ executed โปรแกรม มันจะจัดระเบียบให้กับข้อมูลหลาย ๆ ชนิดที่แตกต่างกัน, data item แต่ละจำนวนยกเว้นมูลค่าคงที่คณิตศาสตร์ (arithmetic constant) หรือมูลค่าคงที่ string (string constant) ซึ่งถูกอ้างถึงในโปรแกรมด้วยชื่อหนึ่งชื่อ ภาษา PL/1 ต้องทราบคุณสมบัติ (หรือ attributes) ของ data items ระหว่างแปลในโปรแกรม แต่กฎเกณฑ์ข้อนี้ก็มีการยกเว้นเหมือนกัน ตัวอย่างเช่น ขอบเขตจำกัดของมิติของ arrays, ความยาวของ

string, ขนาดของเนื้อที่นั้น, มูลค่าเริ่มต้น และ attributes ของบาง file ซึ่งอาจต้องการทราบระหว่างที่มีการ execute ตัวโปรแกรม

Declare และ Default statement

คำสั่ง DACLARE ใช้สำหรับกำหนด attributes ให้กับแต่ละชื่อ, แต่ทุกชื่อในโปรแกรมไม่จำเป็นต้องปรากฏในคำสั่ง DECLARE เสมอไปเพราะ attribute ของมันจะหาได้ภายในเครื่องเองแบบมาตรฐาน แต่คำสั่ง DECLARE อาจจะถือได้ว่าเป็นส่วนที่สำคัญอย่างหนึ่งของ documentation ของโปรแกรมนั้น และผู้เขียนโปรแกรมก็มีอิสระเต็มที่ ในการกำหนดรายละเอียดของข้อมูล ทั้งนี้เพราะว่า ไม่มีข้อจำกัดเกี่ยวกับจำนวนคำสั่ง DECLARE คำสั่ง DECLARE ต่าง ๆ จึงอาจใช้ใน groups ต่าง ๆ ของชื่อได้ ซึ่งข้อนี้ทำให้การปรับปรุงโปรแกรมง่ายขึ้นและการแปลความหมายของ diagnostic ก็ชัดเจนขึ้น

ส่วนคำสั่งอธิบายอื่น ๆ เช่นคำสั่ง OPEN ซึ่งใช้สำหรับกำหนด attributes ให้กับมูลค่าคงที่ file (a file constant) หนึ่ง ๆ ที่จัดอยู่ในกลุ่มคำสั่ง descriptive ด้วย, attributes บางอย่างอาจจะกำหนดได้ ในคำสั่ง ALLOCATE สำหรับตัวแปรควบคุมหนึ่ง ๆ (a controlled variable) คำสั่ง FORMAT ก็เช่นเดียวกัน คือบอกรูปแบบของข้อมูลที่บันทึกในตัวกลางบันทึกข้อมูลภายนอก เช่น ในกระดาษต่อเนื่อง หรือบัตร 80 คอลัมน์

4.2 Input/Output statement

คำสั่งของกลุ่มนี้ใช้สำหรับเคลื่อนย้ายข้อมูลระหว่างข้อมูลภายในหน่วยความจำกับตัวกลางบันทึกข้อมูลภายนอก ส่วนคำสั่ง I/O อื่น ๆ ซึ่งมีผลให้เคลื่อนย้ายข้อมูลเช่นกัน ได้แก่คำสั่ง I/O control ทั้งหมดนี้แบ่งเป็น 2 กลุ่มคือ

4.2.1 Record I/O statements ได้แก่คำสั่ง READ, WRITE, REWRITE, LOCATE และ DELETE

4.2.2 stream I/O statements ได้แก่คำสั่ง GET, PUT

ส่วน I/O Control statement ได้แก่คำสั่ง OPEN, CLOSE และ UNLOCK

หมายเหตุ

คำสั่ง DISPLAY ก็นับรวมอยู่ในกลุ่มนี้ด้วย

ข้อแตกต่างระหว่างการเคลื่อนย้ายข้อมูลแบบ stream และ record ก็คือการเคลื่อนย้ายชนิด stream, data item แต่ละตัวถูกปฏิบัติเป็นอิสระจากกัน ในขณะที่การเคลื่อนย้ายแบบ

record เกี่ยวกับกลุ่มของ data items (records) ทั้งหมดเป็น 1 จำนวน ในการเคลื่อนย้ายชนิด stream นั้น, item แต่ละตัวอาจจะมีการบรรณาธิกรณ์ และเปลี่ยนรูปข้อมูลระหว่างที่เคลื่อนย้าย ส่วนในการเคลื่อนย้ายชนิด record นั้น record ในตัวกลางบันทึกข้อมูลภายนอก ก็คือ รูปแบบเดียวกันกับ record ในหน่วยความจำภายในเครื่องคอมพิวเตอร์นั่นเอง ไม่มีการบรรณาธิกรณ์ ไม่มีการเปลี่ยนรูปข้อมูลใด ๆ ทั้งสิ้น

ข้อแตกต่างที่กล่าวข้างต้นนี้ ทำให้การเคลื่อนย้ายชนิด record สามารถกระทำได้กับ file ใหญ่ซึ่งบันทึกในรหัสของรูปแบบภายใน เช่น รหัสของเลขฐานสอง หรือรหัสเลขฐานสิบได้ ส่วนการเคลื่อนย้ายชนิด stream อาจใช้กับการ process ชนิดข้อมูลในบัตรเจาะรู และผลลัพธ์ให้พิมพ์ออกมาทางกระดาษต่อเนื่อง ซึ่งต้องการให้มีการบรรณาธิกรณ์ด้วย

4.3 Data Movement and Computational Statement

การเคลื่อนย้ายข้อมูลในหน่วยความจำของเครื่องคอมพิวเตอร์ ก็คือการ กำหนดมูลค่าของ expression หนึ่ง ๆ ให้เป็นมูลค่าของตัวแปรซึ่งกำหนดไว้แล้ว (a specified variable) expression ในที่นี้อาจจะเป็นมูลค่าคงที่หนึ่งตัว หรือตัวแปรหนึ่งตัว, หรือ expression หนึ่งซึ่งได้ระบุให้มีการคำนวณ

คำสั่ง assignment เป็นคำสั่งที่ใช้มากที่สุดในการเคลื่อนย้ายข้อมูลภายใน เช่นเดียวกับกำหนดให้มีการคำนวณ อย่างไรก็ตาม คำสั่ง GET และคำสั่ง PUT ซึ่งมี option ก็สามารถใช้สำหรับการเคลื่อนย้ายข้อมูลภายในได้เช่นกัน คำสั่ง PUT อาจระบุให้มีการคำนวณได้ด้วย

Assignment Statement

คำสั่ง assignment ไม่มี keyword กำหนดโดยการใช้เครื่องหมาย = (assignment symbol) รูปแบบทั่วไปให้พิจารณาจากตัวอย่างข้างล่างนี้

```
ATOT    = TOT;
A       = (A*N+T*P) / (N+P);
```

รูปแบบแรก ใช้สำหรับเคลื่อนย้ายข้อมูลภายในโดยตรง, มูลค่าของตัวแปร (หรือมูลค่าคงที่) ทางขวามือของเครื่องหมายเท่ากับ ถูกกำหนดให้กับตัวแปรทางด้านซ้ายมือ

รูปแบบที่สอง มูลค่าของ operational expression ถูกกำหนดให้กับตัวแปรทางซ้ายมือของเครื่องหมายเท่ากับ รูปแบบชนิดนี้กำหนดให้มีการคำนวณเช่นเดียวกับการเคลื่อนย้ายข้อมูล

เนื่องจาก attribute ของตัวแปรทางซ้ายมืออาจจะไม่เหมือนกับ attribute ของผลลัพธ์ของ expression (หรือ ของตัวแปร หรือมูลค่าคงที่) ทางขวามือ ฉะนั้น คำสั่ง assignment จึงอาจจะมีการเปลี่ยนรูปข้อมูลและบรรณาธิกรณ์ข้อมูลด้วย

ตัวแปรทางซ้ายมือ อาจจะเป็นชื่อของ array หรือ a structure expression ทางขวามือก็อาจจะเป็น an array หรือ structure value ก็ได้, คำสั่ง assignment อาจจะใช้เคลื่อนย้ายกลุ่มข้อมูลได้ เช่น เคลื่อนย้ายข้อมูลเดี่ยว (single value)

Multiple assignment

มูลค่าของ expression หนึ่งๆ ในคำสั่ง assignment อาจกำหนดให้กับตัวแปรในคำสั่งนั้นได้มากกว่าหนึ่งตัวก็ได้ โดยที่ระหว่างตัวแปรแต่ละตัวให้ขั้นด้วยเครื่องหมาย comma หนึ่งตัว ดังรูปแบบข้างล่างนี้

```
A,X = B + C;
```

คำสั่ง 1 คำสั่งข้างต้นจะถูก execute โดยวิธีเดียวกับการ execute คำสั่งเดียว (a single statement) เพียงแต่ว่า มูลค่าของ B+C กำหนดให้กับ A และ X โดยทั่วๆ ไป คำสั่งข้างต้นนี้จะมีผลอย่างเดียวกับสองคำสั่งต่อไปนี้

```
A = B + C;
X = B + C;
```

ตัวอย่าง

```
AMOUNT, VALUE, PRICE = 0;
```

มีความหมายอย่างเดียวกับ

```
AMOUNT = 0;
VALUE = 0;
PRICE = 0;
```

4.4 Program Organization Statement

หมายถึงคำสั่งที่ใช้สำหรับกำหนดขอบเขตของส่วนต่าง ๆ ของแต่ละโปรแกรม ให้เป็นบล๊อก (blocks) และจัดการกับบล๊อกเหล่านี้ ได้แก่คำสั่ง PROCEDURE, END, ENTRY, BEGIN, FETCH, และคำสั่ง RELEASE

การแบ่งส่วนต่าง ๆ ของโปรแกรมหนึ่ง ให้เป็นหลาย ๆ บล๊อก เป็นเรื่องที่ถูกต้องทำให้เขียนโปรแกรมง่าย, ตรวจสอบโปรแกรมก็ง่าย โดยเฉพาะอย่างยิ่ง เมื่อโปรแกรมเมอร์หลายคนช่วยกันเขียนโปรแกรมเดียวกัน รวมทั้งเป็นการใช้หน่วยความจำอย่างมีประสิทธิภาพด้วย เนื่องจากว่า dynamic storage จะถูกจัดสรรให้กับ บล๊อก ที่มีการ declared ไว้อย่างอัตโนมัติ

Procedure statement

หน้าที่หลักของ procedure block หนึ่ง ๆ ซึ่งถูกกำหนดด้วยคำสั่ง PRUCEDURE และคำสั่ง END คือการบอกให้ทราบถึงลำดับที่ของ operations ที่จะกระทำกับข้อมูลซึ่งระบุไว้ ลำดับที่ของ operations ต้องกำหนดชื่อให้หนึ่งชื่อเรียกว่า label ของคำสั่ง PROCUDERE และจะถูกอ้างถึงจากตำแหน่งใดตำแหน่งหนึ่ง ซึ่งมีชื่อนั้น

แต่ละโปรแกรมในภาษา PL/1 ต้องมีคำสั่ง PROCEDURE อย่างน้อยที่สุดหนึ่งคำสั่งและคำสั่ง END อีกหนึ่งคำสั่ง โปรแกรมหนึ่งอาจจะประกอบด้วยจำนวน procedures หลายชุดแยกต่างหากจากกัน เขียนเชื่อมโยงเข้าด้วยกัน

Procedure หนึ่งอาจมี procedures อื่น ๆ อยู่ภายใน procedures ซึ่งอยู่ข้างใน (internal procedures) อาจจะมีการ declare ชื่ออีกต่างหาก (ถ้าหากว่าชื่อเหล่านั้นยังไม่เคยกำหนดไว้) และชื่อเหล่านี้ไม่ได้เอาไปใช้นอก บล๊อก ของตน ชื่อเหล่านี้ไม่มีการเรียกใช้ใน procedure นอกเช่นนี้ก็ได้ การติดต่อระหว่าง procedures สองชุด ให้ผ่านทาง arguments จาก procedure เรียกไปยัง procedure ที่ถูกเรียกแล้วจะมีมูลค่าจาก procedure ที่ถูกเรียกกลับไปยัง procedure เรียกด้วยชื่อ ซึ่งทราบทั้งสอง procedures, procedure หนึ่ง ๆ อาจจะ operate ด้วย ข้อมูลแตกต่างกันเมื่อเรียกจากตำแหน่งต่าง ๆ กันก็ได้ มูลค่าซึ่งส่งกลับไปจาก function reference หนึ่ง ๆ ไปยัง function reference ให้ใช้คำสั่ง RETURN

Begin statement

ความหมายอื่น ๆ ของชื่อที่ใช้นอกกลุ่ม อาจอยู่ใน begin block ซึ่งมีขอบเขตจำกัดด้วยคำสั่ง BEGIN และคำสั่ง END ทั้ง 2 คำสั่งนี้ระบุว่าคำสั่งอื่น ๆ ที่อยู่ภายใน จะถูกกระทำทั้งหมดด้วยทิศทางของ control, Begin block จะถูก executed ในทิศทางปกติของโปรแกรมหนึ่ง, สิ่งหนึ่งที่ใช้กันมากที่สุด ใน begin block ก็คือ คำสั่ง ON ซึ่งมีผลให้ทิศทางปกติของ control จะไม่ได้รับ executed ถ้าหากว่าไม่เกิดเงื่อนไขที่ระบุนั้น ซึ่งนับว่าเป็นประโยชน์สำหรับกำหนดขอบเขตจำกัดของแต่ละส่วนในโปรแกรมหนึ่ง ๆ ด้วย, แต่ละ begin block ต้องอยู่ภายใน procedure block หรือ begin block อื่น

End statement

คำสั่งนี้ใช้เป็นคำสั่งสุดท้ายในแต่ละ block, do-group หรือใน select-group, ทุก ๆ block, do-group หรือ select-group ต้องมีคำสั่ง END อย่างไรก็ตาม คำสั่ง END อาจจะเป็น explicit หรือ implicit ก็ได้, คำสั่ง END หนึ่งคำสั่ง อาจจะใช้กับกลุ่มของบล๊อกที่ซ้อน ๆ กัน, do-groups และ select-group โดยอ้างถึง label ของ block, do-group หรือ select-group

คำสั่ง END อื่น ๆ ก็จะถูก implied ด้วยคำสั่ง END หนึ่งคำสั่ง ซึ่งมี label และไม่จำเป็นต้องกำหนดเป็น explicit ถ้าในคำสั่ง END ไม่มี label ตามหลังคำสั่งนั้นหมายถึงมีเพียงหนึ่ง block, do-group หรือ select-group ที่คลุมอยู่นั้น หลังจาก keyword END

4.5 Storage Control statement

ภาษา PL/1 ให้ความสะดวกมากเกี่ยวกับการจัดสรรเนื้อที่ในหน่วยจำ เช่นต้องการจัดสรรเนื้อที่ให้กับตัวแปร และเมื่อโปรแกรมเมอร์ ไม่ต้องการใช้เนื้อที่ส่วนนั้นแล้ว ให้ใช้คำสั่ง ALLOCATE และ คำสั่ง FREE เป็นต้น

ALLOCATE and FREE statement

คำสั่ง ALLOCATE ใช้สำหรับกำหนดเนื้อที่ให้กับ controlled และ based data เป็นอิสระจากขอบเขตของ controlled arrays ขอบเขตจำกัดของ controlled areas, ความยาวของ controlled string และขนาดของ controlled areas เช่นเดียวกับมูลค่าเริ่มต้นต่าง ๆ ซึ่งกำหนดให้ในขณะที่คำสั่ง ALLOCATE ถูก executed ส่วนคำสั่ง FREE ใช้เมื่อไม่ต้องการใช้เนื้อที่เหล่านั้นอีก

Dynamic Storage Allocation

ก่อนที่เครื่องจะ execute โปรแกรม, compiler จะค้นหาตัวแปรต่าง ๆ แล้วกำหนดที่อยู่ให้กับตัวแปรทั้งหมด การกำหนดที่อยู่คือ การจองเนื้อที่ตายตัวไว้ตลอดเวลาที่โปรแกรมกำลังทำงานในบางครั้ง การจองที่ไว้ตลอดเวลาจะไม่จำเป็น และอาจจะกินเนื้อที่มากเกินไป ภาษา PL/1 มีคำสั่งจัดที่เก็บข้อมูลใหม่โดยดึงข้อมูลที่ใช้แล้ว หรือไม่จำเป็นต้องใช้ ในช่วงเวลานั้นออก การควบคุมที่เก็บข้อมูลเหล่านี้เราเรียกว่า dynamic storage allocation ใช้คำสั่ง

```
DCL A CONTROLLED;
```

หรือ DCL A CTL;

คำสั่งนี้จะบอก compiler ว่า ไม่ต้องจองที่ให้ A ก่อนที่จะ execute เมื่อต้องการใช้ A จึงค่อยจัดที่ให้ A โดยใช้คำสั่ง

ALLOCATE A;

หรือ ALLOC A;

เมื่อเลิกใช้ A แล้วจะคืนที่เก็บก็ใช้คำสั่ง

FREE A;

การควบคุมการจัดเนื้อที่นี้ นำไปดัดแปลงใช้กับตัวแปรชนิดอื่นได้ แต่ที่นิยมใช้กันมากที่สุดใช้กับตัวแปรพวก array เพราะ array มักจะใช้เนื้อที่มาก

4.6 Control statement

คำสั่งต่าง ๆ ในภาษา PL/1 โดยทั่วไปแล้วเครื่องจะ executed ให้ตามลำดับ ที่ปรากฏอยู่ยกเว้นในกรณีที่ ทิศทางของการควบคุมถูกเปลี่ยนไปโดยการเกิด interrupt อย่างใดอย่างหนึ่ง หรือเมื่อเครื่อง execute คำสั่งใดคำสั่งหนึ่งของ control statements ต่อไปนี้

GO TO	RETURN
IF	END
SELECT	STOP
DO	EXIT
LEAVE	HALT
CALL	

คำสั่ง GO TO

ใช้เมื่อต้องการเคลื่อนย้าย control ไปยังตำแหน่งอื่นในโปรแกรมอย่างไม่มีเงื่อนไข ถ้าคำสั่ง GO TO นั้นถูกกำหนดโดย label variable มันจะถูกใช้เหมือนกับ switch โดยการกำหนด label control ให้เป็นมูลค่าของ label variable

ถ้า label variable มี subscripted, switch นั้นจะถูกควบคุมโดยการเปลี่ยนแปลงมูลค่าของตัว subscript การใช้คำสั่ง GO TO ก็คือการกำหนด function reference ซึ่ง returns มูลค่าของ label ไม่ว่าจะเป็นการใช้ label variable หรือ function reference ก็ตามไม่มีผลแตกต่างกัน ฉะนั้นวิธีที่ถูกคือพยายามใช้คำสั่งควบคุมชนิดง่าย จะมีประสิทธิภาพมากที่สุด

Keyword ของคำสั่ง GO TO อาจจะเขียนแยกกันเป็นสองคำ ด้วยเครื่องหมาย blank ได้ตั้งแต่หนึ่งตัวขึ้นไป หรืออาจเขียนเป็นคำเดียวว่า GOTO ก็ได้

คำสั่ง IF

คำสั่งนี้ใช้เมื่อต้องการเคลื่อนย้าย control ชนิดมีเงื่อนไข และโดยปกติใช้กับ a simple comparison expression ซึ่งตามหลังคำว่า IF

ตัวอย่าง

```
IF  A = B
    THEN    action-if-true
    ELSE    action-if-false
```

สำหรับ THEN หรือ ELSE clause อาจจะประกอบด้วย a single หรือ compound statement, a do - group, a select-group หรือ a begin block, ถ้าผลลัพธ์ของการเปรียบเทียบเป็นจริงเครื่องจะ execute ส่วนที่เป็น THEN clause โดยข้ามส่วนที่เป็น ELSE clause แล้ว execute คำสั่งถัดไป โปรดสังเกตว่า ในส่วนที่เป็น THEN clause อาจจะมีคำสั่ง GO TO หรือคำสั่ง control อื่น ๆ ซึ่งอาจจะให้ผลลัพธ์เป็นการเคลื่อนย้าย control ไปยังที่อื่น ๆ อีกก็ได้

ถ้าผลลัพธ์จากการเป็นเปรียบเทียบนั้นเป็นเท็จ เครื่องก็จะ execute ให้เฉพาะส่วนที่เป็น ELSE clause เท่านั้น แล้ว control ก็จะทำงานไปตามปกติ

ตัวอย่าง

คำสั่ง IF อาจจะเขียนได้ดังนี้

```
IF A = B
        THEN C = D;
        ELSE C = E;
```

มีความหมายว่า ถ้า A มีมูลค่าเท่ากับ B, มูลค่าของ D, จะถูกกำหนดให้เป็นมูลค่าของตัวแปร C โดยที่เครื่องจะไม่ execute ส่วนที่เป็น ELSE clause

แต่ถ้า A มีมูลค่าไม่เท่ากับ B, มูลค่าของ E จะถูกกำหนดให้เป็นมูลค่าของตัวแปร C โดยที่เครื่องจะไม่ execute ส่วนที่เป็น THEN clause

แต่ทั้ง THEN clause หรือ ELSE clause อาจจะประกอบด้วย a control statement ซึ่งจะทำให้เกิดการเคลื่อนย้าย control อย่างมีเงื่อนไข หรือไม่มีเงื่อนไข ถ้าส่วนที่เป็น THEN มีคำสั่ง GO TO ก็ไม่จำเป็นต้องมีส่วนที่เป็น ELSE clause ตามตัวอย่างต่อไปนี้

```
IF A = B THEN GO TO LABEL_1;
        next-statement
```

มีความหมายว่า ถ้า A มีมูลค่าเท่ากับ B คำสั่ง GO TO ใน THEN clause จะส่ง control ไปยังคำสั่งแรกในตำแหน่ง LABEL_1 โดยไม่มีเงื่อนไข แต่ถ้า A มีมูลค่าไม่เท่ากับ B เครื่องจะไม่ execute ส่วนที่เป็น THEN แต่จะส่ง control ไปยังคำสั่งถัดไป ในคำสั่ง IF หนึ่ง ๆ อาจจะมีส่วนที่เป็น ELSE clause หรือไม่มีก็ได้

หมายเหตุ

ถ้า THEN clause ไม่ได้ส่ง control ไปยังที่อื่น และไม่ได้ตามด้วยส่วนที่เป็น ELSE clause ไม่ว่าส่วนที่เป็น THEN จะได้รับการ execute หรือไม่ก็ตาม หลังจากนั้น เครื่องจะ execute คำสั่งถัดไป

expression ที่ตามหลัง keyword IF ต้องเป็น an element expression เท่านั้น จะเป็น array หรือ structure expression ไม่ได้ แต่อย่างไรก็ตาม expression ในที่นี้อาจจะเป็น a logical expression ที่มีตัว operator มากกว่า 1 ตัวก็ได้ ดังตัวอย่างข้างล่างนี้

```
IF A = B & C = D
  THEN GO TO R;
```

การตรวจสอบชนิดเดียวกันนี้อาจจะอยู่ในคำสั่ง nested IF ได้ตามตัวอย่างข้างล่างนี้

ตัวอย่างที่ 1

```
IF A =   B & C = D
         THEN GO TO R;
   B = B + 1;
```

ตัวอย่างที่ 2

```
IF   A = B
       THEN IF C = D
              THEN GO TO R;
  B = B + 1;
```

ตัวอย่างที่ 3

```
IF A ¬ = B THEN GO TO S;
IF C ¬ = D THEN GO TO S;
GO TO R;
S : B = B + 1;
```

คำสั่ง SELECT

คำสั่งนี้เป็นหัวเรื่องของ a select-group ใน select-group หนึ่ง จะมีการส่ง control ได้หลายวิธี อย่างมีเงื่อนไข โดยมีรูปแบบดังนี้

```
SELECT (E);
    WHEN (E1,E2,E3)      action-1;
    WHEN (E4,E5)         action-2;
        .
        .
        .
    OTHERWISE            action-n;
END;
NL : next statement;
```

ตามตัวอย่างนี้ E, E1, E2... เป็น expression เมื่อ control มาถึงคำสั่ง SELECT เครื่องจะประเมินผล expression E แล้วเก็บมูลค่านี้ไว้แล้วจึงประเมินผล expressions ใน WHEN clause ตามลำดับที่ปรากฏและมูลค่าแต่ละจำนวนจะถูกนำมาเปรียบเทียบกับมูลค่า E, ถ้าปรากฏว่ามีมูลค่าใดที่เท่ากับมูลค่าของ E เครื่องจะ execute action ที่ตามหลัง WHEN clause แต่ถ้าไม่มี expression ใด ๆ เลยใน WHEN clause ที่มีมูลค่าเท่ากับ expression ในคำสั่ง SELECT เครื่องจะ execute action ที่ตามหลัง OTHERWISE clause อย่างไม่มีเงื่อนไข

คำว่า 'action' ซึ่งตามหลัง WHEN หรือ OTHERWISE clause อาจจะเป็น a single หรือ compound statement, a do-group, a select-group หรือ a begin block หลังจากที่เครื่อง execute 'action' แล้ว control ก็จะถูกส่งไปยัง คำสั่ง executable แรกที่ตามหลัง select-group ยกเว้นไว้แต่ว่า ทิศทางของ control นั้นถูกเปลี่ยนไปโดย action ที่กำหนดให้

ถ้าในคำสั่ง SELECT ไม่มี expression, expression แต่ละชุดใน WHEN clause เมื่อประเมินผลถ้าจำเป็นจะถูกเปลี่ยนรูปเป็น bit string ถ้ามีบิทใดบิทหนึ่ง ในผลลัพธ์หนึ่งของ bit string เป็น '1' B, action ที่ตามหลัง WHEN cluase จะถูกกระทำ (performed)

ตัวอย่าง

```
SELECT;
    WHEN (A>B) CALL BIGGER;
    WHEN (A=B) CALL SAME;
    OTHERWISE CALL SMALLER;
END;
```

ถ้าใน select-group หนึ่งๆไม่มี WHEN clause เลย, เครื่องจะ execute action ใน OTHERWISE อย่างไม่มีเงื่อนไข ถ้าใน select-group ไม่มี OTHERWISE clause การ execute ใน select-group จะไม่เกิดผลลัพธ์ในการเลือกของ WHEN cluase ใดๆ เลย แล้วเกิดเงื่อนไข ERROR ขึ้น

ตัวอย่างต่อไปนี้เป็นการแสดง nested select-group ใช้กำหนด (set) ตัวแปรขึ้นมา 1 ตัว ให้เป็นจำนวนวันในเดือนใดเดือนหนึ่ง ที่กำหนดให้

```
DECLARE MONTH CHAR(3),
        YEAR PIC '99',
        NO-DAYS FIXED BINARY;

.
.
.

SELECT (MONTH);
    WHEN ('FEB') SELECT (MOD (YEAR,4));
                    WHEN (0) NO__DAYS = 29;
                    OTHERWISE NO__DAYS = 28;
                    END;
    WHEN ('APR', 'JUN', 'SEP', 'NOV')
                    NO__DAYS = 30;
    OTHERWISE   NO__DAYS = 31;
END;
```

ในตัวอย่างข้างต้นนี้ MOD เป็น built-in function ให้ผลลัพธ์เป็นเศษ (remainder) ของมูลค่า YEAR หารด้วย 4

ตัวอย่างโปรแกรม ใช้คำสั่ง SELECT คำนวณหาเกรดของคะแนนสอบของนักศึกษาจำนวนหนึ่ง โดยมีหลักเกณฑ์ในการคิดเกรดเหมือนกับแบบฝึกหัดข้อ 13 ในหน้า 271

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

                                  SOURCE LISTING

STMT LEV NT
   1     0  EXAM: PROCEDURE OPTIONS (MAIN);
   2   1 0         DCL STD__CODE FIXED DEC(8);
   3   1 0         DCL STD__NAME CHAR(25);
   4   1 0         DCL STD__SCORE FIXED DEC(3);
   5   1 0         DCL SYSIN INPUT, SYSPRINT OUTPUT;
   6   1 0         DCL GRADE FIXED DEC;
   7   1 0             ON ENDFILE (SYSIN) STOP;
   8   1 0         PUT EDIT ('CODE', 'NAME', 'EXAM-MARK', GRADE-POINT')
                   (SKIP,X(42), A, COL(60), A, COL(81), A, COL(95), A);
   9   1 0  NEXT:  GET SKIP EDIT (STD__CODE, STD__NAME, STD__SCORE)
                                  (F(8), A(25), F(3));
  10   1 0         SELECT;
  11   1 1             WHEN (STD__SCORE > 89) GRADE = 4;
  12   1 1             WHEN (STD__SCORE > 79) GRADE = 3;
  13   1 1             WHEN (STD__SCORE > 71) GRADE = 2;
  14   1 1             WHEN (STD__SCORE > 64) GRADE = 1;
  15   1 1             OTHERWISE GRADE = 0;
  16   1 1         END;
  17   1 0             PUT EDIT (STD__CODE, STD__NAME, STD__SCORE, GRADE)
                   (SKIP(2), X(40), F(8), COL(54), A(25), COL(84), F(3), COL(100), F(1));
  18   1 0         GO TO NEXT;
  19   1 0  END EXAM;
```

ลักษณะของ OUTPUT

CODE	NAME	EXAM-MARK	GRADE-POINT
21502136	KARN RAKTAUM	96	4
18745623	SURACHAI JAISERT	64	0
21700557	WITHYA SAEMU	65	1
21501236	ANAN SRISAWAT	100	4
1^702456	KIWIT BUOPROM	71	1
21750023	KARN RUNPASS	72	2
18703201	SUPAPORN BOONSAGAI	79	2
21703536	JINDA POORAHONG	80	3
21500001	LUCHAI VARASIT	90	4
18700002	UDOM SOMBOON	89	3

ตัวอย่างโปรแกรม

จงเขียน flowchart และโปรแกรมอ่านตัวเลขซึ่งเป็นปี ค.ศ. ใดก็ได้แล้วสั่งให้เครื่องคอมพิวเตอร์พิมพ์ จำนวนวันในแต่ละเดือนของปีนั้น นักศึกษาต้องกำหนดรูปแบบให้สวยงาม และมีข้อสังเกตว่าปกติเดือนที่ลงท้ายด้วย "คม" เช่น มกราคม, พฤษภาคม, ... และธันวาคม จะมีจำนวนวันที่เท่ากับ 31 วัน แต่ถ้าลงท้ายด้วย "ยน" เช่น กันยายน ..... และพฤศจิกายน จะมีจำนวนวันเท่ากับ 30 วัน สำหรับเดือนกุมภาพันธ์มีหลักเกณฑ์คิดดังนี้ ถ้าเลขสองตัว ท้ายของปี ค.ศ. นั้นหารด้วย 4 ลงตัวเดือนนั้นจะมี 29 วัน แต่ถ้าเลขสองตัว ท้ายของปี ค.ศ. นั้นหารด้วย 4 ไม่ลงตัว เดือนนั้นจะมี 28 วัน

ตัวอย่างเช่น

ปี ค.ศ. 1980 เดือนกุมภาพันธ์มี 29 วัน เป็นต้น

```
TING: PROC OPTIONS (MAIN);
    DCL   SYSIN INPUT, SYSPRINT OUTPUT,
          MOD BUILTIN,
          YEAR FIXED DEC (4),
          K PIC '9',
          MONTH(12) CHAR(10) VAR,
          (I,J) FIXED BIN,
```

```
        DAYS FIXED DEC;
    ON ENDFILE (SYSIN) STOP;
    GET LIST(YEAR,K);
    GET LIST ((MONTH(I) DO I=1 TO 12));
    DO I=1 TO K;
        PUT PAGE EDIT ('NUMBER DAYS OF MONTH IN', YEAR) (X(52),A,F(4));
        PUT SKIP EDIT ('MONTH', 'NUMBER-DAYS') (COL(53),A,X(12),A);
        PUT SKIP EDIT ((32)'*') (COL(51),A);
        DO J=1 TO 12;
            SELECT (MONTH(J));
                WHEN ('FEBRUARY')
                        SELECT (MOD(YEAR,4));
                        WHEN (0) DAYS=29;
                        OTHERWISE DAYS=28;
                        END;
                WHEN ('APRIL','JUNE','SEPTEMBER','NOVEMBER')
                        DAYS=30;
                OTHERWISE DAYS=31;
            END;
            PUT SKIP(2) EDIT (MONTH(J),DAYS) (X(51),A,COL(75),F(2));
        END;
        PUT SKIP EDIT ((32)'*') (COL(51),A);
        YEAR=YEAR+1;
    END;
END TING;
```

NUMBER DAYS OF MONTH IN 1985

MONTH	NUMBER-DAYS
JANUARY	31
FEBRUARY	28
MARCH	31
APRIL	30
MAY	31
JUNE	30
JULY	31
AUGUST	31
SEPTEMBER	30
OCTOBER	31
NOVEMBER	30
DECEMBER	31

NUMBER DAYS OF MONTH IN 1986

MONTH	NUMBER-DAYS
JANUARY	31
FERRUARY	28
MARCH	31
APRIL	30
MAY	31
JUNE	30
JULY	31
AUGUST	31
SEPTEMBER	30
OCTOBER	31
NOVEMBER	30
DECEMBER	31

คำสั่ง DO

คำสั่ง DO ต้องใช้ควบคู่กับคำสั่ง END ซึ่งเป็นตัวบอกขอบเขตของกลุ่มของคำสั่งทั้งหลาย ซึ่งทั้งหมดนี้เรียกว่า do-group

ปกติเขาจะใช้คำสั่ง DO ในการกำหนด จำนวนครั้งที่จะให้เครื่องคอมพิวเตอร์ execute กลุ่มของคำสั่งชุดหนึ่ง โดยที่ตัวแปรควบคุม (control variable) จะมีมูลค่าเพิ่มขึ้นทุกครั้ง ใน loop, กลุ่มของคำสั่งอาจจะมีรูปแบบดังนี้

```
DO I = 1 TO 10;
.
.
END;
```

ตัวอย่างข้างต้นนี้ กลุ่มของคำสั่งเหล่านี้ จะถูก execute สิบครั้ง โดยที่มูลค่าของ ตัวแปรควบคุม I เท่ากับ 1 ถึง 10 ผลลัพธ์ของคำสั่ง DO และคำสั่ง END จะเหมือนกับกลุ่มของคำสั่งต่อไปนี้

```
    I = 1;
A:IF I>10 THEN GO TO B;
.
.
    I = I + 1;
    GO TO A;
B : next statement
```

โปรดสังเกตว่ามูลค่าของตัวแปรควบคุมเพิ่มขึ้น ก่อนที่จะมี ตรวจสอบโดยปกติ control จะย้ายไปยังคำสั่งที่ตามหลัง กลุ่มของคำสั่งชุดนี้ ก็ต่อเมื่อมูลค่าของตัวแปรควบคุม เกินกว่าขอบเขตจำกัดในคำสั่ง DO ถ้าตัวแปรควบคุมเป็น reference หลังจาก iteration ครั้งสุดท้ายเสร็จเรียบร้อยแล้ว มูลค่าของตัวแปรจะเพิ่มขึ้นที่ละ 1 หลังจากขอบเขตที่กำหนด

ถ้าไม่ได้กำหนดมูลค่าเพิ่ม (increment) เอาไว้ในคำสั่ง DO เครื่องจะถือว่ามูลค่าเพิ่มของตัวแปรควบคุม จะเท่ากับ 1 ตามตัวอย่างข้างต้นนั้น

```
DO I = 2 TO 10 BY 2;
.
.
END;
```

ตัวอย่างข้างต้น loop นี้จะได้รับการ execute 5 ครั้ง โดยที่มูลค่าของตัวแปรควบคุม I เท่ากับ 2, 4, 6, 8 และ 10 ตามลำดับ

ถ้ามูลค่าเพิ่มเป็นลบ คำสั่ง DO ต้องมี option BY ดังนี้

```
DO I = 10 TO 1 BY -1;
.
.
.
.
END;
```

option TO และ BY เป็นตัวกำหนดให้ มูลค่าของตัวแปรควบคุมเปลี่ยนได้ในมูลค่าเพิ่มที่อาจเป็นบวก หรือ ลบ ก็ได้แต่ตรงกันข้าม option REPEAT ซึ่งอาจจะใช้แทน option TO และ option BY เป็นตัวกำหนด ให้มูลค่าของตัวแปรควบคุม เปลี่ยนได้ในมูลค่าที่เป็นเส้นตรง

ตัวอย่าง

```
DO I = 1 REPEAT 2*I;
.
.
.
END;
```

ในตัวอย่างข้างต้นนี้ ในการ execute ครั้งแรก ตัวแปรควบคุม I จะมีมูลค่าเท่ากับ 1 ในการ execute ครั้งต่อ ๆ ไป, expression ใน option REPEAT (หมายถึง 2*I) เมื่อประเมินผลและกำหนดให้เป็นมูลค่าของตัวแปรควบคุม I จะมีมูลค่าดังนี้ 1, 2, 4, 8, 16, และต่อ ๆ ไป

ผลลัพธ์ของตัวอย่างข้างต้น จะมีความหมายเช่นเดียวกับชุดของคำสั่งข้างล่างนี้

```
    I = 1;
A:  .
    .
    I = 2*I;
    GO TO A;
```

หมายเหตุ

option REPEAT ไม่ได้เป็นตัวกำหนดให้เป็นเงื่อนไขหยุด execute ในกลุ่มของคำสั่ง

DO ดังนั้นการ execute ในกลุ่มของคำสั่ง DO จึงดำเนินต่อไป ยกเว้นแต่จะหยุดได้โดยการใช้ option WHILE หรือ option UNTIL หรือ control ถูกส่งออกไปยังนอกกลุ่มคำสั่งชุดนี้

option WHILE และ option UNTIL เป็นวิธีของการจัดให้เครื่องคอมพิวเตอร์ execute คำสั่งใน do-group ภายใต้เงื่อนไขที่โปรแกรมเมอร์กำหนดขึ้น โดยมีรูปแบบเบื้องต้นดังนี้

```
DO WHILE (A = B);
DO UNTIL (A = B);
```

ในคำสั่ง DO WHITE, expression ใน option WHILE จะถูกประเมินผลก่อน (before) ที่จะ execute คำสั่งแต่ละคำสั่งของ do-group, ถ้า expression เป็นจริง เครื่องก็จะ execute do-group แต่ถ้า expression ไม่เป็นจริง control จะถูกส่งออกไปยังคำสั่ง executable แรกที่ตามหลัง do-group ซึ่งมีความหมายเหมือนคำสั่ง ข้างล่างนี้

```
S: IF A = B THEN;
        ELSE GO TO R;
.
.
.
GO TO S;
R: next statement
```

ในคำสั่ง DO UNTIL, expression ใน option UNTIL จะถูกประเมินผลหลัง (after) เมื่อเครื่อง execute แต่ละคำสั่งใน group แล้ว ถ้า expression เป็นจริง control จะถูกส่งไปยังคำสั่ง executable แรกที่ตามหลัง do-group แต่ถ้า expression ไม่เป็นจริง เครื่องก็จะ execute คำสั่งใน group อีก ซึ่งจะมีความหมายเช่นเดียวกับคำสั่งต่อไปนี้

```
S:
   :
   IF (A = B) THEN GO TO R;
   GO TO S;
R: next statement
```

หมายเหตุ

จากกฎเกณฑ์ทั้งหมดข้างต้น สรุปได้ว่า ใน do-group หนึ่ง ๆ ซึ่งมีหัวเรื่องเป็นคำสั่ง DO UNTIL เครื่องจะ execute อย่างน้อยที่สุดหนึ่งครั้ง แต่ใน do-group ที่มีหัวเรื่องเป็นคำสั่ง DO WHILE อาจจะไม่มีการ execute เลยก็ได้ นั่นหมายความว่า คำสั่ง DO WHILE (A = B)

และ คำสั่ง DO UNTIL (A ¬ = B) มีความหมายไม่เหมือนกันเลย

ตัวอย่าง 1

```
DO WHILE (A = B) UNTIL (X = 10);

.

.

.

END;
```

ในตัวอย่างข้างต้นนี้, expression ใน option WHILE จะถูกตรวจสอบก่อนที่เครื่องจะ execute ทุกคำสั่ง ใน group และ expression ใน option UNTIL ถูกตรวจสอบภายหลังจากที่เครื่อง execute แต่ละคำสั่งใน group เสร็จแล้ว

ถ้าในคำสั่ง DO เริ่มต้นโดย A ¬ = B คำสั่งทั้งหมดใน group จะไม่ถูก execute เลย แต่คำสั่ง A=B คำสั่งใน group นั้น จะได้รับการ execute อย่างน้อยที่สุดหนึ่งครั้ง ถ้าหลังจาก execute คำสั่งใน group แล้ว X = 10 จะไม่มีการ performed ใน do-group ซ้ำอีก แต่ถ้า X ไม่เท่ากับ 10 เครื่องจะ performed ซ้ำ ๆ อีก ตราบเท่าที่ A ยังมีมูลค่าเท่ากับ B

ตัวอย่าง 2

```
DO I = 1 TO 10 UNTIL (Y = 1);

.

.

.

END;
```

ตัวอย่างข้างต้นนี้ คำสั่งใน group จะถูก execute อย่างน้อยที่สุดหนึ่งครั้ง เมื่อตัวแปรควบคุม I มีมูลค่าเท่ากับ 1

หลังจาก execute คำสั่งทั้งหมด ใน group แล้ว ถ้า Y = 1 จะไม่มีการ performed คำสั่งใน group อีก แต่ถ้า Y ไม่เท่ากับ 1 มูลค่าเพิ่ม (BY 1) จะบวกกับ ตัวแปรควบคุม I และมูลค่าใหม่ของ I จะถูกนำมาเปรียบเทียบกับ 10 ถ้า I มีมูลค่ามากกว่า 10 เครื่องจะไม่ performed คำสั่งใน group อีก แต่ถ้า I ไม่มากกว่า 10 เครื่องจะ performed คำสั่งใน group ซ้ำ ๆ อีกต่อไป

ตัวอย่าง 3

```
DO I = 1 REPEAT 2*I UNTIL (I=256);

.
.
.

END;
```

ในที่นี้ เมื่อเริ่มต้น execute คำสั่งใน group ตัวแปรควบคุม I= 1 หลังจาก execute แล้ว expression ใน UNTIL option จะถูกตรวจสอบ, ถ้า I=25 จริง จะไม่มีการ performed คำสั่งใน group อีก แต่ถ้าไม่จริง expression ใน REPEAT option จะถูกประเมินผลแล้วกำหนดให้เป็นมูลค่าของตัวแปรควบคุม I แล้วเครื่องก็จะ performed คำสั่งใน group ซ้ำอีก

ผลของการคำนวณคำสั่งต่าง ๆ ใน do-group **สามารถสรุปเป็นหัวข้อ** ได้ดังนี้

1. ถ้ามีตัวแปรควบคุม มูลค่าเริ่มต้น (initial value) จะถูกกำหนดให้เป็นมูลค่าของตัวแปรนั้น
2. ถ้ามี option TO ให้ตรวจสอบมูลค่าของตัวแปรควบคุมกับ expression

ใน option TO ถ้ามูลค่าของตัวแปรควบคุมไม่อยู่ในขอบเขต (range) นั้น control จะถูกส่งออกไปนอก group

3. ถ้ามี option WHILE ให้ตรวจสอบ expression ใน option WHILE ถ้ามีมูลค่าเป็นเท็จ control จะถูกส่งออกไปนอก group
4. execute คำสั่งต่าง ๆ ใน group
5. ถ้ามี option UNTIL ให้ตรวจสอบ expression ใน option UNTIL

ถ้ามีมูลค่าเป็นจริง control จะถูกส่งออกไปนอก group

6. ถ้ามีตัวแปรควบคุม
   a) ถ้ามีทั้ง option TO และ option BY ให้บวกมูลค่าเพิ่มกับตัวแปรควบคุม
   b) ถ้ามี option REPEAT ให้ประเมินผล expression ใน option REPEAT แล้วกำหนดให้เป็นมูลค่าของตัวแปรควบคุม
   c) ถ้าไม่มี option TO, BY, และ REPEAT control จะถูกส่งออกไปนอก group
7. กลับไปทำข้อ 2 ใหม่

(รูปแบบเพิ่มเติมของ interative do-group ให้ไปดูใน Section F หน้า 229

ถ้าในคำสั่ง DO มีตัวแปรควบคุม (DO I = ...;) ส่วนของคำสั่งที่มีอยู่หลังเครื่องหมายเท่ากับ เรียกว่า specification ในคำสั่ง DO หนึ่งคำสั่งอาจจะมีมากกว่า 1 specification ก็ได้ ดูตามตัวอย่างคำสั่ง DO ต่อไปนี้

```
DO I = J, K, L;
DO I = 1 TO 10, 13 TO 15;
DO I = 1 TO 10, 11 WHILE (A=B);
DO I = 1 TO 9, 10 REPEAT 2*I
                    UNTIL (I > 10000);
```

คำสั่งแรก หมายถึงในการ execute คำสั่งใน do-group ครั้งแรก I มีมูลค่าเท่ากับ J, ครั้งต่อไป I มีมูลค่าเท่ากับ K และครั้งสุดท้าย I มีมูลค่าเท่ากับ L ตามลำดับ

คำสั่งที่สอง หมายถึงในการ execute คำสั่งใน do-group เครื่องจะทำทั้งหมด 13 ครั้ง สิบครั้งแรก I มีมูลค่าเป็น 1 ถึง 10 อีกสามครั้งต่อไป I มีมูลค่าเป็น 13 ถึง 15 ตามลำดับ

คำสั่งที่สาม หมายถึงให้ execute คำสั่งใน group อย่างน้อยที่สุด 10 ครั้ง และ (ถ้า A มีมูลค่าเท่ากับ B) อีก 1 ครั้ง ถ้ามี BY 0 ตามหลัง 11 การ execute จะดำเนินต่อไปอีก โดยที่ I=11 เท่าที่ A ยังคงมีค่าเท่ากับ B อยู่

คำสั่งที่สี่ หมายถึง ให้ execute คำสั่งใน group 9 ครั้ง โดยที่มูลค่าของ I เท่ากับ 1 ถึง 9 และเมื่อ I เท่ากับ 10, 20, 40, และ ต่อ ๆ ไป การ execute ซ้ำ ๆ กันนี้ จะหยุดแล้วส่ง control ไปนอก group เมื่อ I มีมูลค่ามากกว่า 10000

หมายเหตุ

คำสั่งทั้งหมดข้างต้น ให้ใช้เครื่องหมาย comma 1 ตัว ขั้นระหว่าง specifications เพื่อเป็นการบอกว่า specification นั้นจะถูก execute ก็ต่อเมื่อได้มีการ execute specification ก่อนหน้านั้นเสร็จแล้ว

ตัวอย่าง

```
DO A = 2 TO 6 BY 2, 8 TO 10, 16;
```

ตัวแปรควบคุมในคำสั่ง DO สามารถนำมาใช้เป็น subscript ในคำสั่งต่าง ๆ ภายใน do-group ได้เพื่อที่ว่าในการ execute ซ้ำกันนั้น กระทำกับอีลิเม้นต์ของตารางข้อมูล หรือ array

ตัวอย่าง

```
DO I = 1 TO 10;
        A (I) = I;
END;
```

ตามตัวอย่างข้างต้นนี้อีลิเม้นต์สิบตัวแรกของ A มีมูลค่าเท่ากับ 1, 2, 3, ... 10 ตามลำดับ

คำสั่ง noniterative DO

คำสั่ง DO ชนิดนี้จะไม่มีการ execute ในคำสั่งต่าง ๆ ใน do-group ซ้ำ ๆ กัน มีวิธีใช้ดังนี้

```
DO;
  .
  .
  .
END;
```

การใช้คำสั่ง DO เช่นนี้เป็นการระบุคำสั่งใน do-group ทั้งหมดจะถูกปฏิบัติราวกับว่าเป็นหนึ่งคำสั่ง ปกติจะใช้กำหนดคำสั่งชุดหนึ่ง ใน THEN หรือ ELSE clause ของคำสั่ง IF หรือหลัง WHEN clause หรือ OTHERWISE clause ของ select–group หนึ่ง ๆ ซึ่งการควบคุมให้ control ทำงานไปตามลำดับโดยไม่ใช้ begin block

คำสั่ง LEAVE

คำสั่งนี้ใช้สำหรับย้าย control จากภายใน do-group หนึ่ง ๆ ให้ไปยังคำสั่ง executable แรกที่ตามหลังคำสั่ง END ซึ่งเป็นตัวบอกขอบเขตของ group นั้น

ตัวอย่าง

```
DO ... ...;
  .
  .
  .
LEAVE;
  .
  .
END;
next statement;
```

ตัวอย่างข้างต้นนี้ คำสั่ง LEAVE ย้าย control ไปยังคำสั่ง 'next statement'

ถ้าคำสั่ง LEAVE ประกอบด้วย reference ไปยัง a statement label (ตัวอย่าง LEAVE A), control จะถูกส่งไปยังคำสั่งซึ่งตามหลังคำสั่ง END ที่ใช้ปิด do-group ของคำสั่ง DO ซึ่งมี label ที่กำหนดนั้น

ตัวอย่าง

```
A: DO I = 1 TO 10;
    DO J = 1 TO 5;
    IF (X, J) = 0 THEN LEAVE A;
    ELSE.....;
    END;
    statement within group A;
END;
    statement after group A;
```

ในที่นี้คำสั่ง LEAVE A จะส่ง control ไปยัง 'statement after group A'

ตัวอย่าง

```
P10: DO...;
P20:  DO...;
P30:  DO...;
        IF CODE = '36' THEN LEAVE;
        END P30;
    END P20;
END P10;
```

คำสั่ง CALL, RETURN, และ END

การเรียก subroutine หนึ่ง ๆ ให้ใช้คำสั่ง CALL ชื่อ subroutine control ก็ถูกส่งมาปฏิบัติงาน และเมื่อพบคำสั่ง RETURN ใน subroutine หรือคำสั่ง END เครื่องจะหยุด execute subroutine นั้น

คำสั่ง RETURN ซึ่งมี expression อยู่ภายในวงเล็บตามหลังใช้ใน function procedure เพื่อ return มูลค่าจำนวนหนึ่ง ไปยัง function refernce หนึ่ง ๆ

ปกติการหยุด execute โปรแกรมหนึ่ง ๆ เกิดขึ้นเมื่อเครื่อง execute คำสั่ง END สุดท้ายของ main procedure หรือคำสั่ง RETURN ใน main procedure หรือ return control กลับไปยัง โปรแกรมเรียก (calling program) ส่วนการหยุด execute ในโปรแกรม โดยวิธีอื่น ๆ นั้นถือว่าผิดปกติ (abnormal)

คำสั่ง STOP และ EXIT

ทั้งสองคำสั่งนี้ ใช้ในการหยุดโปรแกรมชนิดผิดปกติ, คำสั่ง STOP หยุด execute โปรแกรมทั้งหมด ส่วนคำสั่ง EXIT หยุดเฉพาะงาน (task) ชิ้นที่กำลัง execute นั้น

คำสั่ง HALT

คำสั่งนี้จะมีผลก็ต่อเมื่ออยู่ในการ ดำเนินงานชนิด โต้ตอบกันใน batch processing หมายถึงไม่ต้องมี operation ถ้าอยู่ใน source โปรแกรม มันจะหยุด execute โปรแกรมชั่วคราว แล้ว control จะถูกส่งไปยัง เทอร์มินัล (terminal)

4.7 Exception Control Statements

คำสั่ง control ที่กล่าวถึงในหัวข้อที่ผ่านมาแล้วนั้น, การเลือกทิศทางของ control อยู่ที่จะให้ executed หรือไม่ อีกวิธีหนึ่ง ซึ่งลำดับที่ต้องการ execute สามารถเลือกได้จากการเกิด program interrupt ด้วยเงื่อนไขซึ่งเป็นข้อยกเว้นเกิดขึ้น

โดยทั่ว ๆ ไปแล้ว เงื่อนไขซึ่งเป็นข้อยกเว้นหนึ่ง ๆ จะเกิดขึ้นโดยที่เราไม่ได้คาดคิดไว้ก่อน เช่น ข้อผิดพลาดเกี่ยวกับ overflow หรือ action ซึ่งคาดคิดไว้ได้เช่น end of file ซึ่งเกิดได้ในเวลาซึ่งไม่อาจคาดคะเนได้ รายละเอียดของเงื่อนไขเหล่านี้ ให้ศึกษาได้ในบทซึ่งจะกล่าวถึงภายหลัง

แบบฝึกหัด

1. จากคำสั่งข้างล่างนี้

```
Z= X;
IF X = Y THEN DO;X = X**2; Y = (X + Y)/2;END;
        ELSE IF X<Y THEN DO; Y = Y **2; Z = Y - X; END;
        ELSE IF X>0 THEN Z = X/Y; ELSE Y = 200;
```

ถ้ากำหนดให้มูลค่าเริ่มต้นของ X และ Y เป็น

a) 2 และ 7

b) 25 และ 8

c) 15 และ 15

d) -3 และ 3

หลังจากเครื่องคอมพิวเตอร์คำนวณคำสั่งข้างต้นแล้ว จงบอกมูลค่าของ X, Y และ Z

2. จงเขียน flowchart และโปรแกรมหาผลรวมของมูลค่ายกกำลังสองของเลขทุกตัวที่หารด้วย 3 ลงตัว เป็นเลขระหว่าง 1 ถึง 1,000 แล้วพิมพ์ผลลัพธ์ออกมาทาง line-printer
3. จงเขียน flowchart และโปรแกรมอ่านข้อมูล ซึ่งเป็นเลขฐานสิบจำนวนหนึ่ง แล้วคำนวณหาผลบวกของเลขทุกตัวที่มีมูลค่าเป็นบวก ผลบวกของเลขทุกตัวที่มีมูลค่าเป็นลบ, และนับว่ามีเลขที่มีมูลค่าเป็นศูนย์อยู่ทั้งหมดกี่ตัว จากนั้นพิมพ์ผลลัพธ์ทั้งหมด
4. จงเขียนโปรแกรมคำนวณและพิมพ์ตารางแสดงมูลค่าของฟังก์ชั่น

$$f(x,y) = \frac{x^2 - y^2}{x^2 + y^2}$$ เมื่อ x = 2, 4, 6, 8

และ y = 6, 9, 12, 15, 18, 21

output ให้พิมพ์มูลค่า x,y และ f(x,y) ทุกชุด

5. จงเขียน flowchart และโปรแกรมคำนวณหาผลรวมของอนุกรมต่อไปนี้จำนวน 100 เทอม

$$1 - \frac{1}{2} + \frac{1}{4} - \frac{1}{6} + \frac{1}{8} - \frac{1}{10} + \frac{1}{12} - \frac{1}{14} + \cdots$$

6. จงเขียน flowchart และโปรแกรมคำนวณหาผลรวมของอนุกรมต่อไปนี้จำนวน 100 เทอม

$$1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \frac{1}{6} + \frac{1}{7} - \frac{1}{8} + \frac{1}{9} - \frac{1}{10} \cdots\cdots$$

7. จงศึกษาส่วนหนึ่งของโปรแกรมต่อไปนี้ แล้วบอกมูลค่าของอีลิเม้นต์ 17 ตัวแรกของ array A

```
DECLARE (A(0:50), K, N) DECIMAL FIXED(7);
K = 3;
DO N = 0 TO 10;
A(N) = 3*N; END;
DO N = 0 TO 4, 7, 8;
IF A(N) < 10 THEN A(N) = 10* A(N);
        ELSE A(2*N) = A(N)/3; END;
```

8. จงเขียนโปรแกรมอ่านความยาวของด้าน 3 ด้าน (S1, S2 และ S3) ของสามเหลี่ยมรูปหนึ่ง แล้วคำนวณหาพื้นที่ของสามเหลี่ยมรูปนี้ จากสูตร

$$\text{พื้นที่} = \sqrt{T(T-S1)\,(T-S2)\,(T-S3)}$$

$$\text{เมื่อ } T = \frac{S1+S2+S3}{2}$$

9. จงเขียน flowchart และโปรแกรมอ่านความยาวด้าน 3 ด้านของสามเหลี่ยม 1 รูป แล้วคำนวณว่าเป็นรูปสามเหลี่ยมชนิดใด โดยมีหลักเกณฑ์การคำนวณดังนี้

ให้ A เป็นความยาวมากที่สุดของด้าน S1, S2, S3

B และ C เป็นความยาวของอีก 2 ด้านที่เหลือแล้ว

ถ้า $A \geqslant B+C$ ไม่สามารถทำเป็นรูปสามเหลี่ยมได้ (no triangle)

ถ้า $A^2 = B^2+C^2$ เป็นรูปสามเหลี่ยมมุมฉาก (right-angle triangular)

ถ้า $A^2 > B^2+C^2$ เป็นรูปสามเหลี่ยมมุมป้าน (obtuse triangle)

ถ้า $A^2 < B^2+C^2$ เป็นรูปสามเหลี่ยมมุมแหลม (acute triangle)

แล้วพิมพ์ข้อความบอกชนิดของรูปสามเหลี่ยมนั้น

10. จงเขียนโปรแกรมอ่าน character string 1 ชุด

a) แล้วนับว่าใน string ชุดนี้มีตัวอักษร s กี่ตัว

b) ใน string ชุดนี้มีตัวสระ A, E, I, O และ U อย่างละกี่ตัว

11. ในการคัดเลือกนายกองค์การนักศึกษาของมหาวิทยาลัยหัวหมาก มีผู้สมัครเข้ารับการคัดเลือก 5 คน นักศึกษาทุกคนเป็นผู้มีสิทธิออกเสียงลงคะแนนวิธีการหยั่งเสียง ให้นักศึกษาแต่ละคนกรอกรหัสประจำตัวของตน และเลือกหมายเลขผู้สมัครเพียงหมายเลขเดียว คือ 1, 2, 3, 4, หรือ 5 ลงในกระดาษคอมพิวเตอร์เพื่ออ่านเข้าเครื่องต่อไป จงเขียน flowchart และโปรแกรมนับคะแนนการหยั่งเสียงครั้งนี้ แล้วพิมพ์คะแนนการหยั่งเสียงสุดท้ายของผู้สมัครแต่ละคน และประกาศหมายเลขผู้ได้คะแนนสูงสุด

เฉลยข้อ 8

```
/*-------------------------------------------------------------------------*/
     THIRD:PROCEDURE OPTIONS(MAIN);
           DCL (B,C,ZERO) FIXED DEC (7,0),
                A(20) FIXED DEC (4,0) INIT(-9,-8,-7,-6,-5,-4,-3,-2,
                -1,0,0,1,2,3,4,5,6,7,8,9)STATIC,
                 I FIXED BIN;
           PUT EDIT((A(I) DO I=1 TO 20))
                (SKIP,COL(10),(10(F(3),X(2))));
            DO I = 1 TO 20;
             IF A(I) < 0 THEN C=C+A(I);
                 IF A(I) = 0 THEN ZERO = ZERO + 1;
                   IF A(I) > 0 THEN B=B+A(I);
             END;
            PUT SKIP EDIT ('SUM ADD POSITIVE NUMBER =',B)
                         (COL(5),A,F(7,0));
            PUT SKIP EDIT ('SUM ADD NAGATIVE NUMBER =',C)
                         (COL(5),A,F(7,0));
            PUT SKIP EDIT ('NUMBER OF ZERO =',ZERO)
                         (COL(5),A,F(7,0));
           END;
          END THIRD;
```

```
third

         -9   -8   -7   -6   -5   -4   -3   -2   -1    0
          0    1    2    3    4    5    6    7    8    9
    SUM ADD POSITIVE NUMBER =      45
    SUM ADD NAGATIVE NUMBER =     -45
    NUMBER OF ZERO =       2
```