บทที่ 3
การเพิ่มคำสั่งแบบโครงสร้างให้อินเตอร์พริเตอร์

บทที่ 2 ที่ผ่านมาได้กล่าวถึงแบบการทำงาน, โครงสร้างโปรแกรม, โครงสร้างข้อมูล, ตลอดจนตารางต่าง ๆ ของอินเตอร์พริเตอร์ เพื่อเป็นพื้นฐานที่จะทำความเข้าใจหลักการทำงานของอินเตอร์พริเตอร์ส่วนวิธีการที่จะดัดแปลงอินเตอร์พริเตอร์เพื่อเพิ่มความสามารถต่าง ๆ จะมากล่าวไว้ในบทนี้ โดยมีรายละเอียดต่าง ๆ ดังนี้คือ

1 การเพิ่มคำสั่งใหม่ให้กับอินเตอร์พริเตอร์
2 คำสั่งและรูปแบบของคำสั่งใหม่ที่เพิ่มเข้าไป
3 หลักการทำงาน ของคำสั่งใหม่แต่ละคำสั่ง
4 ขั้นตอน (ALGORITHM) ของแต่ละคำสั่ง

1 การเพิ่มคำสั่งใหม่ให้กับอินเตอร์พริเตอร์

เพื่อให้บรรลุจุดประสงค์ของวิทยานิพนธ์ฉบับนี้คือการเพิ่มคำสั่งแบบโครงสร้างให้กับอินเตอร์พริเตอร์ สิ่งที่ต้องทำมีดังนี้คือ

ก. แก้ไขอินเตอร์พริเตอร์เพื่อรองรับการเพิ่มคำสั่ง
ข. เขียนโปรแกรมของแต่ละคำสั่งที่เพิ่มขึ้น

ก. การแก้ไขอินเตอร์พริเตอร์เพื่อรองรับการเพิ่มคำสั่ง

ในการแก้ไขอินเตอร์พริเตอร์ต้องแก้ไขในส่วนต่าง ๆ ดังนี้

1) เล็กซิเคล แอนนาไลซีส (LEXICAL ANALYSIS) คือทำให้ส่วนนี้รับรู้คำสั่งใหม่ที่เพิ่มเข้าไป

2) กำหนดรหัสของแต่ละคำสั่งที่เพิ่มเข้าไปและแก้ไขโปรแกรมส่วนที่เรียกว่า เท็กอะตอม ไมเซชั่น ให้รับรู้กับรหัสใหม่ที่ตั้งขึ้นว่าคำสั่งใดมีรหัสเป็นอะไร

3) ทำให้ส่วนที่ทำการควบคุมการทำงานของอินเตอร์พริเตอร์ รับรู้ว่าถ้าได้รับคำสั่งใหม่ที่เพิ่มเติมเข้าไปแล้วจะต้องส่งการควบคุมไปให้กับโปรแกรมย่อยส่วนไหน

4) แก้ไขโปรแกรมของคำสั่งบางคำสั่งที่มีอยู่แล้วเดิม เนื่องจากคำสั่งใหม่บางครั้งมีผลทำให้โครงสร้างของข้อมูลและวิธีการทำงานของคำสั่งบางคำสั่งเดิมเปลี่ยนแปลงไปดังนั้นต้องมีการเปลี่ยนแปลงคำสั่งนั้นให้เหมาะสม คำสั่งเหล่านั้นได้แก่ คำสั่ง GOSUB, RETURN, NEW, END, STOP, SWAP, ERASE.

5) แก้วิธีการเข้าถึงค่าของตัวแปรเสียใหม่

ข. <u>เขียนโปรแกรมของแต่ละคำสั่ง</u>

แต่ละโปรแกรมย่อยแบ่งออกเป็น 2 ส่วน คือ

1) ส่วนที่ดำเนินงานตามวัตถุประสงค์ของแต่ละคำสั่ง เช่นคำสั่ง LOCAL ก็ทำหน้าที่จองเนื้อที่ในหน่วยความจำให้กับตัวแปรที่จะใช้เฉพาะส่วนใดส่วนหนึ่งของโปรแกรม

2) ส่วนตรวจสอบความผิดพลาดของคำสั่งแบ่งออกเป็น 2 ส่วน

ก) ตรวจสอบความผิดพลาดทางรูปแบบของคำสั่ง (SYNTAX ANALYSER)

ข) ตรวจสอบความผิดพลาดในขณะปฏิบัติงาน (RUNTIME ERROR ANALYSER)

2. <u>คำสั่งและรูปแบบของคำสั่ง</u>

คำสั่งต่าง ๆ ที่เพิ่มเข้าไปมีดังนี้ คือ

ก. IFS-THENDO-ELSEDO-ENDIF

ข. LOOP- EXIT WHEN-ENDLOOP

ค. LOCAL

ง. GLOBAL

จ. กลุ่มคำสั่ง Labeled Subroutine Call

หัวข้อต่อไปนี้จะเป็นรายละเอียดของคำสั่งต่าง ๆ เหล่านี้

ก. คำสั่ง IF-THEN-ELSE-ENDIF

รูปแบบของคำสั่ง

IFS (logical expr)

THENDO

THEN BLOCK

ENDDO

ELSEDO

ELSE BLOCK

ENDDO

ENDIF

จุดมุ่งหมายของคำสั่ง

เป็นการเลือกทางเดินการทำงานของโปรแกรมว่าจะไปทำงานตามกลุ่มของคำสั่งใด จะทำงานตามกลุ่มของคำสั่งใดขึ้นอยู่กับว่าเงื่อนไขที่กำหนดไว้เป็นจริงหรือไม่จริง ถ้าเป็นจริงก็ทำตามกลุ่มของคำสั่ง THENDO ถ้าไม่จริงก็ก็ทำตามกลุ่มของคำสั่ง ELSEDO และชุดคำสั่งของ IFS-THENDO-ELSEDO-ENDDO นี้ สามารถใช้ซ้อนกันได้ไม่จำกัดจำนวนครั้งแต่ถูกจำกัดด้วยขนาดของหน่วยความจำที่ใช้เก็บค่าของสแตค (STACK) ต่าง ๆ

ตัวอย่าง การใช้ชุดของคำสั่ง IFS ได้แสดงไว้ในรูป 3.1 และผลของการใช้

โปรแกรมในรูป 3.1 ได้แสดงไว้ในรูป 3.2

```
10    PRINT "DEMONSTRATE IFS-THENDO-ELSEDO-ENDIF COMMAND"
20    A = 10
30    IFS A = 10
40        THENDO
50            PRINT "1111"
60        ENDDO
70        ELSEDO
80            PRINT "AAAA"
90        ENDDO
100   ENDIF
110   PRINT "END PROGRAM"
```

รูปที่ 3.1 แสดงตัวอย่างโปรแกรมการใช้คำสั่ง IFS

```
DEMONSTRATE IFS-THENDO-ELSEDO-ENDIF COMMAND

111

END PROGRAM
```

รูปที่ 3.2 แสดงผลของโปรแกรมการใช้คำสั่ง IFS

ข. คำสั่ง LOOP : ENDLOOP

รูปแบบของคำสั่ง

LOOP

BLOCK OF STATEMENT

EXIT WHEN (logical expre)

BLOCK OF STATEMENT

ENDLOOP

จุดมุ่งหมายของคำสั่ง

เป็นการวนซ้ำการทำงานกลุ่มคำสั่งใด ๆ จนกว่าเงื่อนไขที่กำหนดไว้ในคำสั่ง EXIT WHEN เป็นจริง จึงออกจากการทำงานวนซ้ำไปทำงานตามคำสั่งที่อยู่ต่อจากคำสั่ง ENDLOOP โดยคำสั่ง EXIT WHEN จะอยู่ส่วนใดในกลุ่มของคำสั่งที่ทำงานเป็นวนซ้ำก็ได้ และคำสั่ง LOOP นี้สามารถใช้ซ้อนกันกี่ครั้งก็ได้ไม่จำกัด แต่ถูกจำกัดด้วยจำนวนหน่วยความจำที่ใช้ในการเก็บค่าลงในสแตค (STACK)

ตัวอย่าง การใช้คำสั่ง LOOP-ENDLOOP แสดงในรูป 3.3 และผลจากการใช้โปรแกรมแบบนี้แสดงในรูป 3.4

```
10    PRINT "DEMONSTRATE LOOP COMMAND"
20    A = 0
30    LOOP
40        PRINT "IN LOOP"
50        A = A + 1
60        EXIT WHEN A = 3
70    ENDLOOP
80    PRINT "END PROGRAM"
```

รูปที่ 3.3 แสดงตัวอย่างโปรแกรมการใช้คำสั่ง LOOP

```
DEMONSTRATE LOOP COMMAND
IN LOOP
IN LOOP
IN LOOP
END PROGRAM
```

รูปที่ 3.4 แสดงผลของการใช้คำสั่ง LOOP จากโปรแกรมรูป 3.3

ค. คำสั่ง LOCAL

รูปแบบของคำสั่ง

LOCAL Varisble 1, Variable 2, ...

จุดหมายของคำสั่ง

เพื่อเป็นการจองเนื้อที่ในหน่วยความจำสำหรับเก็บค่าตัวแปรนั้น ๆ และเป็นการระบุว่าตัวแปรที่อยู่ในคำสั่งนี้ใช้เฉพาะคำสั่งต่าง ๆ ที่อยู่ต่อจากคำสั่งนี้ไปจนถึงคำสั่ง RETURN หรือคำสั่ง RETURNL เปรียบเสมือนกับตัวแปรเหล่านี้ใช้เฉพาะในโปรแกรมย่อยนั้น ๆ เท่านั้น ส่วนข้อกำหนดเกี่ยวกับตัวแปรต่าง ๆ เหมือนข้อกำหนดเดิมของอินเตอร์พริเตอร์ตัวเดิมทุกประการ

ตัวอย่าง การใช้คำสั่ง LOCAL แสดงในโปรแกรมรูปที่ 3.5 และผลของการใช้โปรแกรมนี้แสดงในรูปที่ 3.6

```
10   PRINT "DEMONSTRATE LOCAL COMMAND"
20   A = 100
30   PRINT "BEFORE CALL SUBROUTINE A = ", A
40   GOSUB 100
50   PRINT "AFTER RETURN A= ", A
60   END
100  LOCAL A
110  A = 1
120  PRINT "IN SUBROUTINE A = ", A
130  RETURN
```

รูปที่ 3.5 แสดงตัวอย่างโปรแกรมการใช้คำสั่ง

```
DEMONSTRATE LOCAL COMMAND
BEFORE CALL SUBROUTINE  A = 100
IN  SUBROUTINE  A = 1
AFTER RETURN  A  =  100
```

รูปที่ 3.6 แสดงผลของการใช้คำสั่ง LOCAL

ง. <u>คำสั่ง GLOBAL</u>

<u>รูปแบบของคำสั่ง</u>

GLOBAL Variable, Variable 2,

<u>จุดหมายของคำสั่ง</u>

เพื่อเป็นการจองเนื้อที่ในหน่วยความจำสำหรับเก็บค่าของตัวแปรนั้น ๆ และเป็นการระบุว่าตัวแปรนั้น ๆ ใช้ได้ทั่วไปข้อกำหนดต่าง ๆ สำหรับตัวแปรแบบนี้เหมือนข้อกำหนดของตัวแปรทั่วไปของอินเตอร์พริเตอร์

<u>ตัวอย่าง</u> การใช้คำสั่ง GLOBAL แสดงในรูป 3.7 และผลของการใช้โปรแกรมนี้แสดงในรูป 3.8

```
10    GLOBAL  A,  B,  C
20    A  =  1
30    B  =  2
40    C  =  3
50    PRINT "A=",  A,  "B=" B,  "C=",  C
```

รูปที่ 3.7 แสดงตัวอย่างการใช้คำสั่ง GLOBAL

```
A = 1 B = 2 C = 3
```

รูปที่ 3.8 แสดงผลของการใช้คำสั่ง global จากโปรแกรมในรูป 3.7

จ. กลุ่มคำสั่ง LABELED SUBROUTINE CALL เป็นการเรียกใช้โปรแกรมย่อย โดยเรียกชื่อแทนที่จะเป็นหมายเลขประจำบรรทัดการใช้นี้ประกอบด้วย 3 คำสั่งคือ

GOSUBL

RETURNL

LABEL

1) คำสั่ง GOSUBL

รูปแบบของคำสั่ง

```
GOSUBL  Label name
```

จุดมุ่งหมายของคำสั่ง

label name จะขึ้นต้นและประกอบด้วยตัวอักษรอะไรก็ได้ยกเว้นช่องว่าง และ COLON (:) ความยาวของ label name ไม่จำกัดแต่ต้องไม่เกินขนาดความจำของแป้นพิมพ์

คำสั่งนี้เป็นการส่งการควบคุมไปยังโปรแกรมย่อยใดโปรแกรมย่อยหนึ่ง โดยไม่มีเงื่อนไขใด ๆ (UNCONDITIONAL JUMP) และเมื่ออินเตอร์พรีเตอร์ปฏิบัติงานตามโปรแกรมย่อยนั้นจนถึงคำสั่ง RETURNL การควบคุมก็จะถูกส่งกลับมายังคำสั่งที่อยู่ต่อจากคำสั่ง GOSUBL คำสั่ง GOSUBL 1 คำสั่งสามารถมีคำสั่ง RETURNL ได้หลายคำสั่ง และคำสั่ง GOSUBL สามารถใช้ในลักษณะซ้อนได้ (Nested) ไม่จำกัดจำนวนครั้ง จะถูกจำกัดอยู่ที่ขนาดของหน่วยความจำที่ใช้เก็บตัวแปรและหน่วยความจำที่ใช้เก็บสแตค (stack)

2) คำสั่ง RETURNL

รูปแบบของคำสั่ง

```
RETURNL
```

จุดมุ่งหมายของคำสั่ง

ใช้คู่กับคำสั่ง GOSUBL โดยคำสั่ง RETURNL เป็นคำสั่งส่งการควบคุมกลับไปยังคำสั่งที่อยู่ต่อจากคำสั่ง GOSUBL ที่เรียกโปรแกรมย่อยนี้

3) คำสั่ง LABEL

รูปแบบของคำสั่ง

```
LABEL label name.:
```

จุดมุ่งหมายของคำสั่ง

เพื่อเป็นการกำหนดชื่อของโปรแกรมย่อยหรือส่วนต่าง ๆ ของโปรแกรม เพื่อใช้อ้างอิงในการส่งการควบคุมตามคำสั่ง

ตัวอย่าง การใช้ชุดของคำสั่ง Label subroutine call อยู่ในโปรแกรมดังแสดงในรูป 3.9 ส่วนผลของการใช้โปรแกรมนี้แสดงในรูป 3.10

```
10   PRINT "DEMONSTRATE GOSUBL  COMMAND"
20   GOTO  60
30   LABEL  AAA:
40   PRINT  "IN GOSUB  LABEL"
50   RETURNL
60   PRINT  "BEFORE  GOSUBL  LABEL"
70   COSUBL   AAA
80   PRINT  "AFTER  GOSUB  LABEL"
90   PRINT  "END  PROGRAM"
100  END
```

รูปที่ 3.9 แสดงการใช้กลุ่มคำสั่ง LABELED SUBROUTINE CALL

```
DEMONSTRATE  GOSUBL  COMMAND
BEFORE GOSUB  LABEL
IN  GOSUB  LABEL
AFTER  GOSUB  LABEL
END  PROGRAM .
```

รูปที่ 3.10 แสดงผลของการใช้กลุ่มของคำสั่ง LABELED SUBROUTINE CALL

3. นโยบายหลักของการปฏิบัติตามคำสั่งที่เพิ่มขึ้น

ในหัวข้อต่อไปนี้เป็นนโยบายหลักของคำสั่งต่าง ๆ ที่เพิ่มขึ้น ซึ่งอธิบายถึงขั้นตอนการทำงานของแต่ละคำสั่งมีรายละเอียดดังต่อไปนี้

1) คำสั่ง IFS

เนื่องจากคำสั่ง IFS เป็นคำสั่งที่ใช้ในการตัดสินใจเลือกทางเดินของโปรแกรมว่าควรไปทำงานตามกลุ่มของคำสั่งใด ขึ้นอยู่กับเงื่อนไขที่กำหนดให้มาว่าเป็นจริงหรือเท็จ ดังนั้นหน้าที่หลักของคำสั่ง IFS คือ

ก. ประเมินผลว่าเงื่อนไขที่กำหนดมานั้นเป็นจริงหรือเท็จ

ข. หาตำแหน่งเริ่มต้นของคำสั่งที่จะต้องปฏิบัติการต่อไปถ้าเงื่อนไขที่กำหนดมาเป็นจริง (คือหาตำแหน่งของคำสั่ง THENDO ที่คู่กัน)

ค. หาตำแหน่งที่ต้องปฏิบัติการต่อไปถ้าเงื่อนไขที่กำหนดไม่เป็นจริง (คือหาตำแหน่งของคำสั่ง ELSEDO ที่คู่กัน)

ง. หาตำแหน่งของคำสั่งที่จะปฏิบัติการต่อไป หลังจากไปปฏิบัติการตามกลุ่มของคำสั่งตามเงื่อนไขที่กำหนดมาแล้ว (คือหาตำแหน่งของคำสั่ง ENDIF ที่คู่กัน)

เมื่อทำการประมวลผลตามเงื่อนไขที่กำหนด อินเตอร์พรีเตอร์จะส่งการควบคุมไปยังกลุ่มของคำสั่งนั้น ๆ และส่งข้อมูลบางอย่างไปยังคำสั่ง THENDO, ELSEDO, ENDDO โดยข้อมูลเหล่านั้นถูกเก็บภายในสแตกโดยไม่รบกวนสภาพเดิมของสแตก

2) <u>คำสั่ง THENDO และ ELSEDO</u>

คำสั่งทั้งสองนี้ เป็นคำสั่งที่ใช้คู่กับคำสั่ง IFS โดยคำสั่งเหล่านี้เป็นคำสั่งที่จะต้องปฏิบัติการหลังการตัดสินใจของคำสั่ง IFS ทำหน้าที่ดังนี้คือ

ก. เป็นตัวบอกจุดเริ่มต้นของชุดของคำสั่ง

ข. ทำการตรวจสอบว่าก่อนปฏิบัติการตามคำสั่งนี้ต้องผ่าน คำสั่ง IFS มาแล้ว

3) <u>คำสั่ง ENDDO</u>

คำสั่งนี้เป็นคำสั่งสุดท้ายในชุดของคำสั่ง THENDO และ ELSEDO ก่อนส่งการควบคุมไปยังคำสั่ง ENDIF หน้าที่ของคำสั่งนี้คือ

ก. ตรวจสอบว่าก่อนปฏิบัติการตามคำสั่งนี้ต้องผ่านคำสั่ง THENDO หรือ ELSEDO มาแล้ว

ข. เป็นตัวบอกว่าสิ้นสุดการทำงานในชุดของคำสั่ง THENDO หรือ ELSEDO

ค. ส่งการควบคุมไปยังคำสั่ง ENDIF โดยเอาตำแหน่งของคำสั่ง ENDIF มาจากสแตกซึ่งคำสั่ง IFS ส่งมา

4) คำสั่ง ENDIF

คำสั่งนี้ เป็นคำสั่งในชุดของคำสั่ง IFS ซึ่งเป็นตัวบอกให้รู้ว่าเมื่อเสร็จสิ้นการทำงานในชุดของคำสั่ง THENDO หรือ ELSEDO แล้วการควบคุมต้องถูกส่งมายังคำสั่งนี้

5) คำสั่ง LOOP

คำสั่งนี้เป็นคำสั่งอยู่ในชุดการทำงานวนซ้ำ โดยคำสั่ง LOOP จะเป็นจุดเริ่มต้นของการทำงานวนซ้ำ หน้าที่หลักของคำสั่งนี้คือ

ก. เก็บตำแหน่งของคำสั่งที่อยู่ต่อจากคำสั่ง LOOP นี้ เพื่อเป็นข้อมูลใส่ในสแตกสำหรับคำสั่ง ENDLOOP ในการส่งการควบคุมมายังคำสั่งเริ่มต้นของการทำงานเป็นวนซ้ำ (NESTED - LOOP ได้

ข. เก็บตำแหน่งของคำสั่งต่อจากคำสั่ง ENDLOOP เพื่อว่าเมื่อต้องการออกจากการทำงานวนซ้ำจะได้ส่งการควบคุมมา ยังคำสั่งนี้

6) EXIT WHEN

คำสั่งนี้เป็นคำสั่งในชุดการทำงานวนซ้ำทำหน้าที่ในการตัดสินใจว่าเมื่อไรควรจะออกจากการทำงานวนซ้ำ โดยดูตามเงื่อนไขที่กำหนดถ้าเงื่อนไขเป็นจริงก็ไปทำงานตามคำสั่งหลัง ENDLOOP ถ้าไม่จริงก็ทำงานตามคำสั่งที่ตามมา นั่นคือคำสั่งนี้ต้องทำหน้าที่

ก. ทำการประมวลผลเงื่อนไขที่กำหนด

ข. ส่งการควบคุมไปยังคำสั่งตามแต่ผลของเงื่อนไข

7) ENDLOOP

คำสั่งนี้เป็นคำสั่งสุดท้ายในชุดของคำสั่งการทำงานวนซ้ำ ทำหน้าที่ดังนี้คือ

ก. เป็นตัวบอกให้คำสั่ง LOOP รู้ว่าคำสั่งนี้เป็นคำสั่งสุดท้ายในการทำงานวนซ้ำ

ข. ตรวจสอบว่าก่อนปฏิบัติการตามคำสั่งนี้ได้ผ่านคำสั่ง LOOP มาแล้ว

ค. ส่งการควบคุมไปยังคำสั่งแรกของการทำงานวนซ้ำ

8) คำสั่ง GLOBAL / LOCAL

คำสั่งทั้งสองคำสั่งทำหน้าที่ในการกำหนดว่าตัวแปรที่ตามมาใช้ได้เฉพาะในโปรแกรมย่อยนั้นเท่านั้นหรือใช้ได้ทั่วไป คำสั่งนี้จะจองเนื้อที่ในหน่วยความจำให้กับตัวแปรในรูปแบบของตัวแปรชนิดนั้น ๆ ส่วนการเข้าถึงหรือตรวจสอบว่าตัวแปรตัวใดเป็น LOCAL ตัวแปรตัวใดเป็น GLOBAL ทำได้โดยการตรวจสอบตัวแปรที่เป็น LOCAL ในโปรแกรมย่อยนั้นก่อนถ้าไม่มีก็ไปตรวจสอบตัวแปรที่เป็น LOCAL ในโปรแกรมย่อยระดับถัดไป ถ้ายังไม่เจออีกก็ทำอย่างนี้เรื่อย ๆ จนกว่าจะถึงระดับสุดท้ายถ้ายังไม่เจออีกก็ถือว่าตัวแปรนั้นเป็น GLOBAL

9) คำสั่ง, GOSUBL

คำสั่งนี้ทำหน้าที่ในการควบคุมการส่งการควบคุมไปยังคำสั่งต่าง ๆ หน้าที่ของคำสั่งนี้คือ

ก. หาตำแหน่งของคำสั่งที่ต้องการส่งการควบคุมไป

ข. เก็บตำแหน่งของคำสั่งที่ต่อจากคำสั่งนี้เพื่อเป็นข้อมูลสำหรับคำสั่ง RETURNL ในการส่งการควบคุมกลับ

ค. กำหนดจุดเริ่มต้นของหน่วยความจำที่จะใช้เก็บตัวแปรซึ่งจะใช้เฉพาะในโปรแกรมย่อยนั้นเท่านั้น

ง. ส่งการควบคุมไปยังโปรแกรมย่อยนั้น

10) คำสั่ง LABEL

คำสั่งนี้เป็นคำสั่งบอกจุดเริ่มต้นของโปรแกรมย่อย โดยทำหน้าที่บอกให้คำสั่ง GOSUBL รู้ว่าคำสั่งนี้เป็นคำสั่งเริ่มต้นของโปรแกรมย่อย คำสั่ง LABEL นี้ไม่ต้องทำอะไรเมื่อรับการควบคุมมาก็ส่งการควบคุมกลับ

11) คำสั่ง RETURNL

คำสั่งนี้เป็นคำสั่งที่จะส่งการควบคุมกลับไปยังคำสั่งที่อยู่ต่อจากคำสั่ง GOSUBL ที่เรียก

โปรแกรมย่อยนี้ หน้าที่ของคำสั่งนี้คือ

ก. ลบตัวแปรในหน่วยความจำที่ใช้สำหรับโปรแกรมย่อยนั้น

ข. ส่งการควบคุมกลับมายังคำสั่ง GOSUBL ที่เรียกโปรแกรมย่อยนี้

4. ขั้นตอน (ALGORITHM) ของแต่ละคำสั่ง

รายละเอียดของขั้นตอนของแต่ละคำสั่งจะกล่าวไว้ในภาคผนวก ข

5. สรุป

ในบทนี้เป็นการเพิ่มคำสั่งใหม่ให้กับอินเตอร์พรีเตอร์โดยแบ่งออกเป็น 2 ส่วน คือ

ก. แก้ไขอินเตอร์พรีเตอร์ของเดิมเพื่อให้รับรู้คำสั่งใหม่ที่เพิ่มเข้าไป และคำสั่งเดิมบางคำสั่งเพื่อให้สามารถทำงานร่วมกับคำสั่งใหม่นี้ได้อย่างถูกต้อง

ข. เขียนโปรแกรมเพิ่มเติมเพื่อควบคุมการทำงานของคำสั่งใหม่ให้เป็นไปตามจุดประสงค์ของคำสั่งใหม่นั้น ๆ

คำสั่งต่าง ๆ ที่เพิ่มเข้าไปได้แก่

1 IFS - THENDO - ELSEDO - ENDIF

2 LOOP - EXIT WHEN - ENDLOOP

3 LOCAL

4 GLOBAL

5 GOSUBL - RETURNL - LABEL

ส่วนขั้นตอน (ALGORITHM) โปรแกรมของแต่ละคำสั่งที่เพิ่มเข้ามาตลอดจนโปรแกรมที่ทำการทดสอบคำสั่งต่าง ๆ เหล่านี้จะแสดงไว้ในภาคผนวก ข