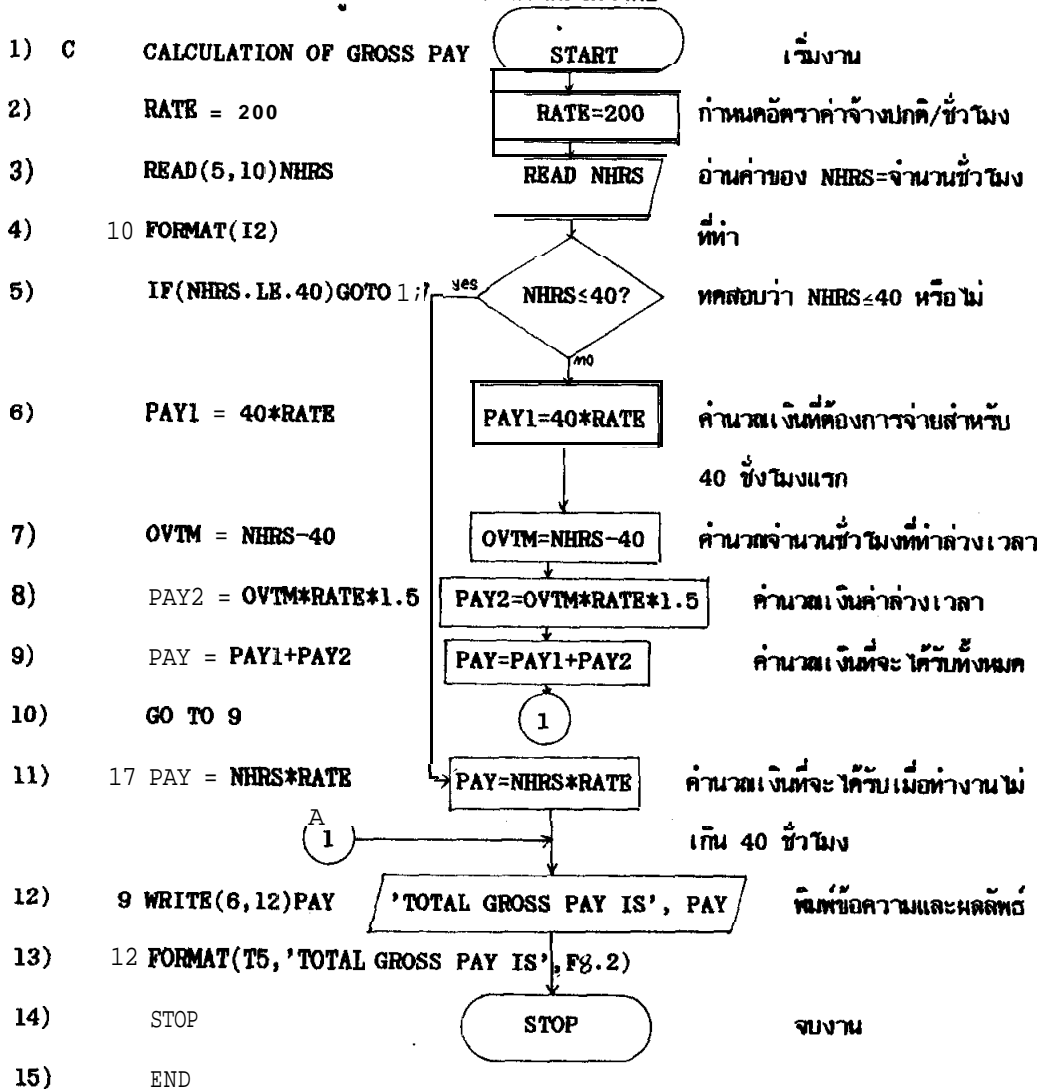


## บทที่ 2 ภาษาฟอร์แทรนเบื้องต้น (ตอนที่ 2)

### 2.1 ปัญหาตัวอย่าง

นายสุเทพได้รับค่าจ้างในอัตราปกติเป็นเงิน 200 บาท/ชั่วโมงสำหรับ 40 ชั่วโมงแรก  
อัตราค่าล่วงเวลา (สำหรับจำนวนชั่วโมงที่เกิน 40 ชั่วโมง) เป็น 1.5 เท่าของอัตราปกติ  
จงเขียนโปรแกรมภาษาฟอร์แทรนเพื่ออ่านจำนวนชั่วโมงที่ทำงาน แล้วคำนวณเงินที่นายสุเทพจะได้  
แล้วพิมพ์ค่าที่คำนวณได้ด้วย

โปรแกรมและแผนผังสายงานเพื่อทำสิ่งที่ต้องการคือ



โปรแกรมจะคำนวณค่าจ้างในบรรทัดที่ 6)-9) อาจเขียนแทนได้ด้วย

$$\text{PAY} = (40 * \text{RATE}) + (\text{NHRS} - 40) * \text{RATE} * 1.5$$

และเพื่อให้โปรแกรมยืดหยุ่นต่อการใช้งาน เราอาจอ่านค่าของ RATE และอัตราที่จะจ่ายค่า ชั่วโมงเมื่อทำล่วงเวลา

ดังนั้นเมื่อทำค่างล่วงข้างต้นแล้วโปรแกรมจะกลายเป็น

```
C      CALCULATION OF GROSS PAY,  
  
      READ(5,10)NHRS, RATE, FACTOR  
  
10     FORMAT(I2, F3.0, F3.1)  
  
      IF(NHRS. LE.40)GO TO 17  
  
      PAY=40*RATE+(NHRS-40)*RATE* FACTOR  
  
      GOT09  
  
17     PAY=NHRS*RATE  
  
      9     WRITE(6,12)PAY  
  
12     FORMAT(T5, 'TOTAL GROSS PAY IS', F8.2)  
  
      STOP  
  
      END
```

คำสั่งใหม่ที่เราพบในโปรแกรมข้างต้นคือ

- 1) คำสั่ง READ ซึ่งใช้ในการอ่านข้อมูลเข้าจากตัวกลาง (input medium)
- 2) คำสั่ง GO TO ซึ่งใช้ในการเปลี่ยนลำดับของการทำงานไปจากลำดับการทำงานที่ทำ

ต่อเนื่องกันอยู่

- 3) คำสั่ง IF ซึ่งใช้ในการควบคุมลำดับการทำงานโดยมีเงื่อนไข
- 4) Comment line ใช้อธิบายส่วนของโปรแกรม

## 2.2 คำสั่งในภาษาฟอร์แทรน

### 2.2.1 คำสั่ง READ

เท่าที่ผ่านมาเรากำหนดค่าให้แก่ตัวแปรโดยใช้คำสั่งในการกำหนดค่า ดังในตัวอย่างแรกเรากำหนดค่าให้ RATE มีค่า 200 โดยใช้คำสั่งกำหนดค่าเลขคณิต RATE=200 วิธีนี้ทำให้เราใช้ RATE ที่มีค่า 200 ได้เท่านั้น ถ้าเราต้องการใช้โปรแกรมเดียวกันแต่สำหรับค่าของ RATE ที่ต่าง ๆ กัน เราจะใช้วิธีบันทึกค่าของ RATE ไว้ในค้ำกลางแล้วใช้คำสั่ง READ ให้อ่านค่าเหล่านั้นเข้าไปเก็บใน RATE เมื่อเราต้องการใช้ค่าใหม่ของ RATE

รูปทั่วไป READ(เลขประจำเครื่อง, เลขประจำคำสั่ง FORMAT) รายชื่อตัวแปร

โดยที่ - เลขประจำเครื่องคือเลขจำนวนเต็มซึ่งใช้แทนเครื่องนำข้อมูลเข้า เช่น เครื่องคอมพิวเตอร์

แม่เหล็ก เครื่องอ่านบัตร เป็นต้น การใช้เลขจำนวนใดแทนเครื่องใดนั้นขึ้นอยู่กับศูนย์

คอมพิวเตอร์ ในที่นี้จะใช้ 5 แทนเครื่องอ่านบัตร

- เลขประจำคำสั่ง FORMAT ซึ่งคือคำสั่งที่ใช้อธิบายตำแหน่งของรายการข้อมูล (data item) ต่าง ๆ ในระเบียบ (Record) เลขประจำคำสั่งนี้โปรแกรมเมอร์เป็นผู้ตั้งขึ้นเอง

- รายชื่อตัวแปร ใช้ระบุชื่อของรายการข้อมูลที่ค่าของมันจะถูกอ่านจากค้ำกลาง ชื่อแต่ละชื่อจะเป็นชื่อของที่อยู่ในหน่วยความจำหลักนั่นเอง

ตัวอย่าง READ(5, 10) HRS, RATE, BONUS

เลขประจำเครื่อง เลขประจำคำสั่ง รายชื่อตัวแปร

คำสั่งนี้จะอ่านค่า 3 ค่าจากบัตร 1 บัตรหรือมากกว่าหนึ่งบัตร และเก็บในหน่วยความจำหลักในชื่อ HRS, RATE และ BONUS ตามลำดับ

### 2.2.2 คำสั่ง FORMAT

เราได้เห็นแล้วว่าคำสั่ง READ เพียงคำสั่งเดียวนั้นไม่อาจทำให้คอมพิวเตอร์รู้จักข้อมูลไปอย่างเพียงพอ คือ

- 1) จำนวนรายการข้อมูลคือบัตร 1 บัตร
- 2) แต่ละรายการข้อมูลเริ่มจากสนามใด
- 3) แต่ละรายการข้อมูลบันทึกอยู่ในที่สนามใด (ความยาวของฟิลด์)
- 4) ชนิดของข้อมูลเป็น integer หรือ real

คำสั่ง FORMAT จะใช้กับคำสั่ง READ เพื่อบอกคอมพิวเตอร์ว่าการอ่านจะอ่านอย่างไร จะอ่านรายการข้อมูลต่าง ๆ กันจากระเบียนอย่างไร

คำสั่ง READ จะระบุเลขประจำคำสั่ง FORMAT คำสั่ง FORMAT แต่ละคำสั่งต้องมีเลขประจำคำสั่งเพื่อใช้เมื่อต้องการอ้างอิง คำสั่ง READ เป็นคำสั่งปฏิบัติการในขณะที่คำสั่ง FORMAT ไม่เป็นคำสั่งปฏิบัติการ การอ่านจะเป็นไปตามสิ่งที่ระบุไว้ในคำสั่ง FORMAT ไปรอกแอมเมอร์ บางท่านนิยมใส่คำสั่ง FORMAT ทั้งหมดรวมกันไว้ตอนท้ายของโปรแกรมก่อนคำสั่ง END และมีหลายคนที่วางคำสั่ง READ

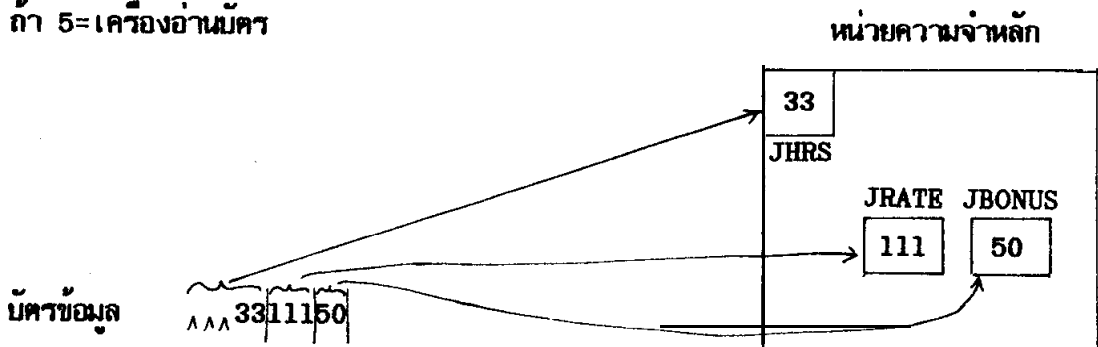
ตัวอย่าง  

```

      READ(5,10) JHRS, JRATE, JBONUS
                |   |   |
      10 FORMAT(   I5 ,   I3 ,   I2 )

```

ถ้า 5 = เครื่องอ่านบัตร



คำสั่งข้างต้นจะทำให้อ่านเลข 3 จำนวนจากบัตร (ถ้า 5 = เครื่องอ่านบัตร) นำไปเก็บในหน่วยความจำที่ตั้งไว้ การอ่านจะอ่านโดยยึดตำแหน่งที่ระบุไว้ในคำสั่ง FORMAT หมายเลข 10 คำสั่ง FORMAT ที่มีเลขประจำคำสั่ง 10 มีความหมายดังนี้ เลขจำนวนแรก (ค่าของ JHRS) จะเป็นเลขจำนวนเต็ม (ระบุโดยใช้ I-format code) ซึ่งอยู่ในสนามที่ 1-5 เลขจำนวนถัดไป (ค่าของ JRATE) จะเป็นเลขจำนวนเต็ม ซึ่งอยู่ในสนามที่ 6-8 และเลขจำนวนสุดท้าย (ค่าของ JBONUS) จะเป็นเลขจำนวนเต็ม ซึ่งอยู่ในสนามที่ 9-10

โค้ดกล่าวถึงคำสั่ง FORMAT มาแล้วใน 1.2.11 ในที่นี้จะกล่าวถึง format code I

และ F เมื่อใช้กับคำสั่ง READ

I-format code

รูปทั่วไป



โดยที่ I ใช้ระบุจำนวนเต็มจะถูกอ่าน

w ใช้ระบุความกว้าง (จำนวนสกรีน) ของฟิลด์

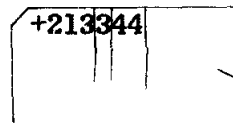
เราจะต้องสำรอง 1 สกรีนไว้สำหรับเครื่องหมายของจำนวนเต็มลบ ช่องว่าง (blank)

ในฟิลด์ของข้อมูลเข้านั้น ไม่ว่าจะอยู่ส่วนใดของฟิลด์ เครื่องจะถือว่าเป็นศูนย์ ดังนั้นเลขจำนวนเต็ม จึงต้องบันทึกทิศทางของฟิลด์ ข้อมูลที่เป็นเลขจำนวนเต็มต้องประกอบด้วยตัวเลข 0-9 เครื่องหมาย บวก (+) หรือลบ (-) ตัวอักษรอื่น ๆ ถ้าปรากฏในฟิลด์ที่ระบุด้วย Iw จะทำให้เกิดข้อผิดพลาด ในการดำเนินการ (execution time READ error) ตัวแปรในรายชื่อตัวแปรในคำสั่ง READ ที่คู่กับ Iw ต้องเป็นตัวแปรจำนวนเต็ม (integer variable)

ตัวอย่าง READ(5,15)I,J,K

15 FORMAT(I4,I1,I2)

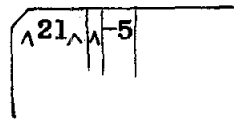
บัตรข้อมูล



ค่าในหน่วยความจำหลัก

I=213  
J=3  
K=44

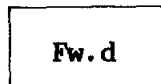
บัตรข้อมูล



I=210  
J=0  
K=-5

F-format code

รูปทั่วไป



โดยที่ F ใช้ระบุจำนวนจริงจะถูกอ่าน

w ใช้ระบุความกว้างของฟิลด์

d ใช้ระบุตำแหน่งของจุดทศนิยมให้แก่เลขจำนวนที่อ่าน ในกรณีที่เลขจำนวนในฟิลด์ไม่ได้  
 จะจุดไว้ ค่าที่เก็บในหน่วยความจำจะมีตัวเลขหลังจุดเท่ากับ d ตัว  
 ในกรณีที่ฟิลด์นั้นมีจุดทศนิยมอยู่แล้ว ค่าที่อ่านเข้าไปเก็บในหน่วยความจำจะเป็นเลขจำนวน  
 ค้างเขตที่ปรากฏในฟิลด์ นั่นคือ d จะไม่มีความหมายในกรณีนี้ ดังนั้น d อาจเป็นเลขใด ๆ ระหว่าง  
 0 ถึง w

ช่องว่างในฟิลด์จะถูกอ่านเป็นศูนย์ ถ้าเลขจำนวนเป็นลบจะต้องใช้ 1 สัญลักษณ์สำหรับเครื่องหมายลบด้วย และถ้าเลขจำนวนมีจุดทศนิยมอยู่ จะต้องใช้ 1 สัญลักษณ์สำหรับจุดด้วย

ตัวแปรในคำสั่ง READ ซึ่งใช้ F-format code นั้นจะต้องเป็นตัวแปรชนิด real

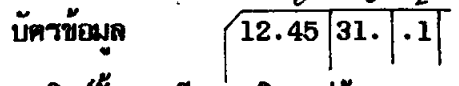
ตัวอย่าง                 READ(5,5)A, B,     C  
                                        ↓          ↓          ↓  
                5 FORMAT(F5.5,F3.2,F2.0)



ในฟิลด์ทั้ง 3 เป็นเลขจำนวนเต็ม ดังนั้น .d ใน F-format code จะเป็นตัวกำหนดตำแหน่ง  
 ของจุดทศนิยมของค่าที่จะเก็บในหน่วยความจำหลักดังนี้

A จะมีค่า =  $12345 \times 10^{-5} = .12345$   
 B จะมีค่า =  $789 \times 10^{-2} = 7.89$   
 C จะมีค่า =  $31 \times 10^{-0} = 31.$

ตัวอย่าง                 READ(5,1)X,Y,Z  
                                        ↓          ↓          ↓  
                1 FORMAT(F5.5,F3.2,F2.0)



ในฟิลด์ทั้ง 3 มีจุดทศนิยมอยู่ด้วย .d ใน F-format code จะใส่ไว้เพื่อให้การเขียนถูกต้อง  
 ตามกฎเกณฑ์เท่านั้น ค่าที่อ่านเข้าไปเก็บจะเป็นตามที่ปรากฏในฟิลด์ดังนี้

X จะมีค่า = 12.45  
 Y จะมีค่า = 31.  
 Z จะมีค่า = .1

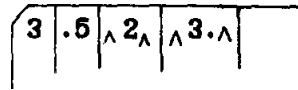
ตัวอย่าง READ(5,8)X,Y,Z,W

X จะเก็บค่า = .3

8 FORMAT(F1.1,F2.0,F3.2,F4.3)

Y จะเก็บค่า = .5

บัตรข้อมูล



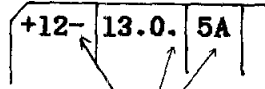
Z จะเก็บค่า = .20

W จะเก็บค่า = 3.0

ตัวอย่าง READ(5,9)X,Y,Z

9 FORMAT(F4.2,F5.1,F2.0)

บัตรข้อมูล

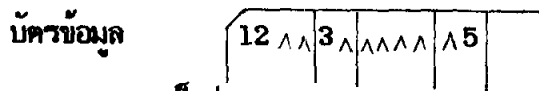


ตัวอักษรเหล่านี้ใช้ไม่ได้

จากตัวอย่างนี้จะเกิดข้อผิดพลาดในการอ่านตามคำสั่ง READ เนื่องจากมีตัวอักษรที่ใช้ไม่ได้ในฟิลด์ตัวเลข (Numeric field)

ในการเขียนคำสั่ง READ โปรแกรมเมอร์ต้องตรวจสอบเสมอว่าชนิดของตัวแปรในรายชื่อตัวแปรนั้นสอดคล้องกับชนิดของ format code ที่ใช้หรือไม่ นั่นคือถ้าตัวแปรมีชนิด real ก็ต้องใช้ F-format code ถ้าตัวแปรมีชนิด integer ก็ต้องใช้ I-format code เป็นต้น

ตัวอย่าง READ(5,10)A,K,B,J  
10 FORMAT(F4.2,I2,F4.0,I2)



A จะเก็บค่า 12.00

K จะเก็บค่า 30

B จะเก็บค่า 0.

J จะเก็บค่า 5

นอกจากการใช้ I และ F ในคำสั่ง FORMAT แล้ว ในการกำหนด FORMAT ในการนำข้อมูลเข้ายังมีบางสิ่งที่ควรกล่าวถึง สิ่งเหล่านี้เป็น Edit codes ซึ่งมันจะช่วยให้โปรแกรมเมอร์สามารถทำการเว้นข้ามสัณขที่ไม่ต้องการอ่าน โดยการใช้ X-format code

X-format code

รูปทั่วไป

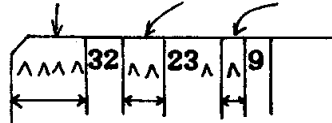


โดยที่ p เป็นเลขจำนวนเต็มที่จะระบุจำนวนตำแหน่งที่จะถูกข้ามไป  
 X เป็นตัวระบุว่าจะมีการข้ามสมการระหว่างฟิลด์

ตัวอย่าง READ(5,6)X,I,J

6 FORMAT(4X,F2.1,2X,I3,1X,I1)

บัตรข้อมูล



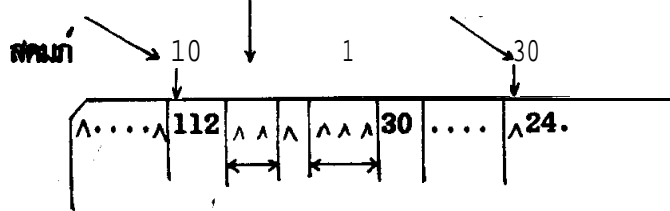
ผลการอ่าน X จะเก็บค่า 3.2  
 I จะเก็บค่า 230  
 J จะเก็บค่า 9

ตัวอย่าง การใช้ทั้ง X และ T-format code

READ(5,7)X,I,J,Z

5 FORMAT(T10,F3.2,2X,I1,3X,I2,T30,F4.1)

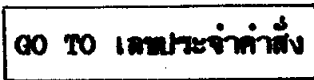
บัตรข้อมูล



ผลการอ่าน x จะเก็บค่า 1.12  
 I จะเก็บค่า 0  
 J จะเก็บค่า 30  
 z จะเก็บค่า 24.

2.2.3 คำสั่ง GO TO แบบไม่มีเงื่อนไข (Unconditional GO TO statement)

รูปทั่วไป





เลขประจำคำสั่ง ต้องเป็นของคำสั่งปฏิบัติการ  
โปรแกรมภาษาฟอร์แทรนประกอบด้วยชุดของคำสั่ง คำสั่งจะถูกปฏิบัติตามไปตามลำดับ  
ของคำสั่ง เมื่อพบคำสั่ง GO TO มันจะโยกย้ายไปทำตามคำสั่งที่ระบุเลขประจำคำสั่งไว้ในคำสั่ง  
GO TO คำสั่งนี้จะทำให้โปรแกรมเมอร์สามารถที่จะข้ามคำสั่งบางส่วนในโปรแกรมได้ และทำให้  
สามารถสั่งให้ทำซ้ำชุดของคำสั่งตามจำนวนครั้งที่ต้องการได้ ซึ่งเราเรียกว่าการลูป (looping)

ตัวอย่าง GO TO 5  
 } คำสั่งส่วนนี้ถูกข้ามไป  
 5 READ(5,10)X

ตัวอย่าง 6 READ(5,10)GRADE  
 } คำสั่งชุดนี้จะถูกปฏิบัติซ้ำ  
 GO TO 6

#### 2.2.4 คำสั่ง IF ตรรกะ (Logical IF statement)

หน่วยคำนวณและตรรกะ ในหน่วยประมวลผลกลางสามารถทำงานตามตรรกะได้  
หน่วยนี้สามารถทำการเปรียบเทียบเลขจำนวนได้ และใช้ผลจากการเปรียบเทียบในการตัดสินใจ

คำสั่ง IF จะทำให้คอมพิวเตอร์โยกย้ายการทำงานไปยังที่ ๆ ต้องการโดยขึ้นอยู่กับว่าเงื่อนไขที่ตั้งไว้จริงหรือไม่จริง ด้วยวิธีนี้โปรแกรมจึงอาจประกอบด้วยทางเลือกหลาย ๆ  
ทางซึ่งแต่ละทางเลือกนั้นขึ้นอยู่กับข้อมูล ชุดของคำสั่งอาจถูกข้ามไปโดยไม่ถูกปฏิบัติตามทั้งนี้  
เนื่องมาจากผลของคำสั่ง IF ตรรกะ

จากโปรแกรมของปัญหาตัวอย่างข้อ 2.1 ถ้า NHRS มีค่าน้อยกว่าหรือเท่ากับ 40  
ชุดของคำสั่งตั้งแต่บรรทัดที่ 6) ถึงบรรทัดที่ 10) จะถูกข้ามไป การปฏิบัติงานจะเริ่มอีกครั้ง  
ที่คำสั่งที่มีเลขประจำคำสั่งคือ 17 (บรรทัดที่ 11) แต่ถ้า NHRS มีค่ามากกว่า 40 คำสั่งถัดไป  
จากคำสั่ง IF คือคำสั่งในบรรทัด 6) จะถูกปฏิบัติตาม รูปทั่วไปของคำสั่ง IF ตรรกะคือ

IF(นิพจน์ตรรกะ)GO TO เลขประจำคำสั่ง

โดยที่ นิพจน์ตรรกะ (logical expression) คือนิพจน์ที่ประกอบด้วยนิพจน์เลขคณิต 2 นิพจน์  
ซึ่งถูกเชื่อมกันด้วยตัวดำเนินการเปรียบเทียบ (relational operator : rop)

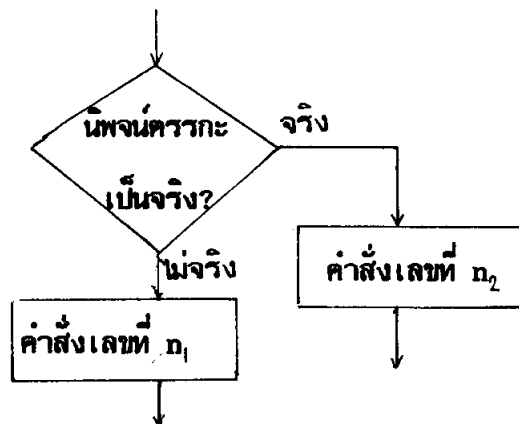
ตั้งแสดงในตาราง ค่าของนิพจน์ตรรกะจะมีค่าเป็นจริง (true) หรือเป็นเท็จ (false) เท่านั้น (จะได้กล่าวถึงนิพจน์ตรรกะและตัวดำเนินการตรรกะ (logical operator) ในตอนหลัง)

เลขประจำคำสั่งคือเลขประจำคำสั่งของคำสั่งที่ต้องการให้ถูกระงับเมื่อนิพจน์ตรรกะเป็นจริง ในกรณีนิพจน์ตรรกะเป็นเท็จ คำสั่งที่อยู่ถัดจากคำสั่ง IF ถูกระงับ

ตารางแสดงตัวดำเนินการเปรียบเทียบ (rop)

FORTRAN rop	เครื่องหมายทางคณิตศาสตร์	ความหมาย
.EQ.	=	เท่ากับ
.LT.	<	น้อยกว่า
.GT.	>	มากกว่า
.NE.	≠	ไม่เท่ากับ
.LE.	≤	น้อยกว่าหรือเท่ากับ
.GE.	≥	มากกว่าหรือเท่ากับ

ภาพแสดงการทำงานของคำสั่ง IF ตรรกะ

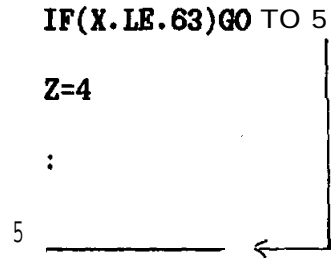
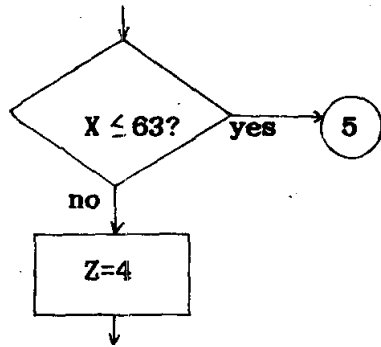


```

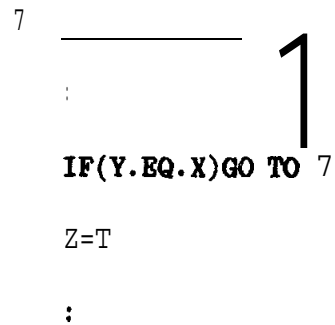
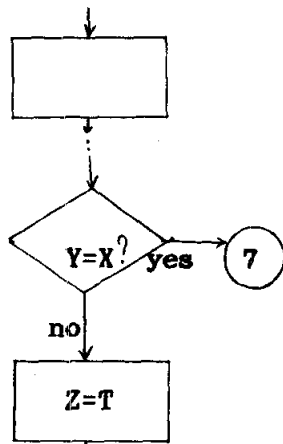
;
IF(นิพจน์ตรรกะ)GO TO n2
n1 _____
n2 _____
  
```

ไปคำสั่งเกินกว่าคำสั่งเลขที่  $n_1$  นี้จะต้องเขียนติดไว้กับคำสั่ง IF ในขณะที่คำสั่งเลขที่  $n_2$  อาจอยู่ก่อนหรือหลังคำสั่ง IF หรืออยู่ที่ใด ๆ ก็ได้ตามหลังคำสั่งเลขที่  $n_1$

ตัวอย่าง



ตัวอย่าง



คำสั่ง IF ตรรกะและความหมายของคำสั่ง

คำสั่ง IF ตรรกะ	ความหมาย
ก) 1 IF(X.EQ.0.)GO TO 5 3 A=4	ถ้า $X=0$ จะไปทำคำสั่งที่ 5 ต่อไป แต่ถ้า $X < 0$ หรือ $X > 0$ คำสั่งถัดไปคือคำสั่งที่ 3 จะถูกกระทำการ
ข) 4 IF((X-Y)**2.LT.Z)GO TO 40 8 IF(Z.GT.2)GO TO 60	ถ้า $(X-Y)^2 < Z$ จะทำคำสั่งที่ 40 แต่ถ้า $(X-Y)^2 \geq Z$ จะทำคำสั่งที่ 8 ต่อไป
ค) 5 IF(SQRT(X).GE.2.)GO TO 50 7 WRITE(6,11)X	ถ้า $\sqrt{X} \geq 2$ จะทำคำสั่งที่ 50 ถ้า $\sqrt{X} < 2$ จะทำคำสั่งที่ 7 ต่อไป

ง) 2 IF(X+Y.NE. (J-K))GO TO 70

6 READ(5,5)A

ถ้า  $X+Y \neq J-K$  จะหาค่าสั่งที่ 70

แต่ถ้า  $X+Y=J-K$  จะหาค่าสั่งที่ 6 ค่อยไป

ตัวอย่างโปรแกรม อ่านเลขจำนวนเต็ม I จากบัตรและตรวจดูว่า I มีค่าเป็นเลขคู่หรือเลขคี่ เนื่องจากกาหาร integer ด้วย integer นั้น ผลลัพธ์จะถูกตัดเศษที่เป็นทศนิยมทั้งหมด (truncation)

ถ้าค่าของ I เป็นเลขคู่แล้ว  $2*(I/2)=I$  ตัวอย่างเช่น ถ้า  $I=4$  แล้ว

$$2*(4/2)=2*2=4$$

ถ้าค่าของ I เป็นเลขคี่แล้ว  $2*(I/2) \neq I$  ตัวอย่างเช่น ถ้า  $I=5$  แล้ว

$$2*(5/2)=2*2=4 \neq 5$$

ผังโปรแกรมและโปรแกรมสำหรับปัญหาข้างต้น (ใช้คำสั่ง IF ตาราง)

อ่านค่าของ I จากบัตร

ค่าของ I เป็นเลขคู่ใช่ไหม?

ถ้าค่าของ I ไม่ใช่เลขคู่ พิมพ์ค่า

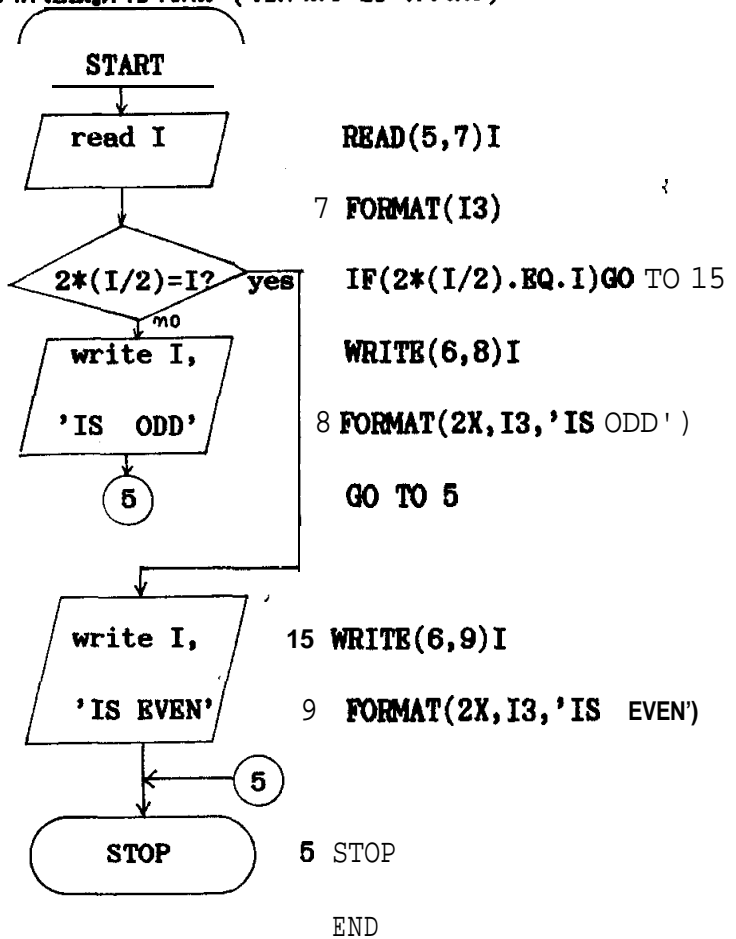
I และข้อความ 'IS ODD'

จบงาน

ถ้าค่าของ I เป็นเลขคู่ พิมพ์ค่า

I และข้อความ 'IS EVEN'

จบงาน



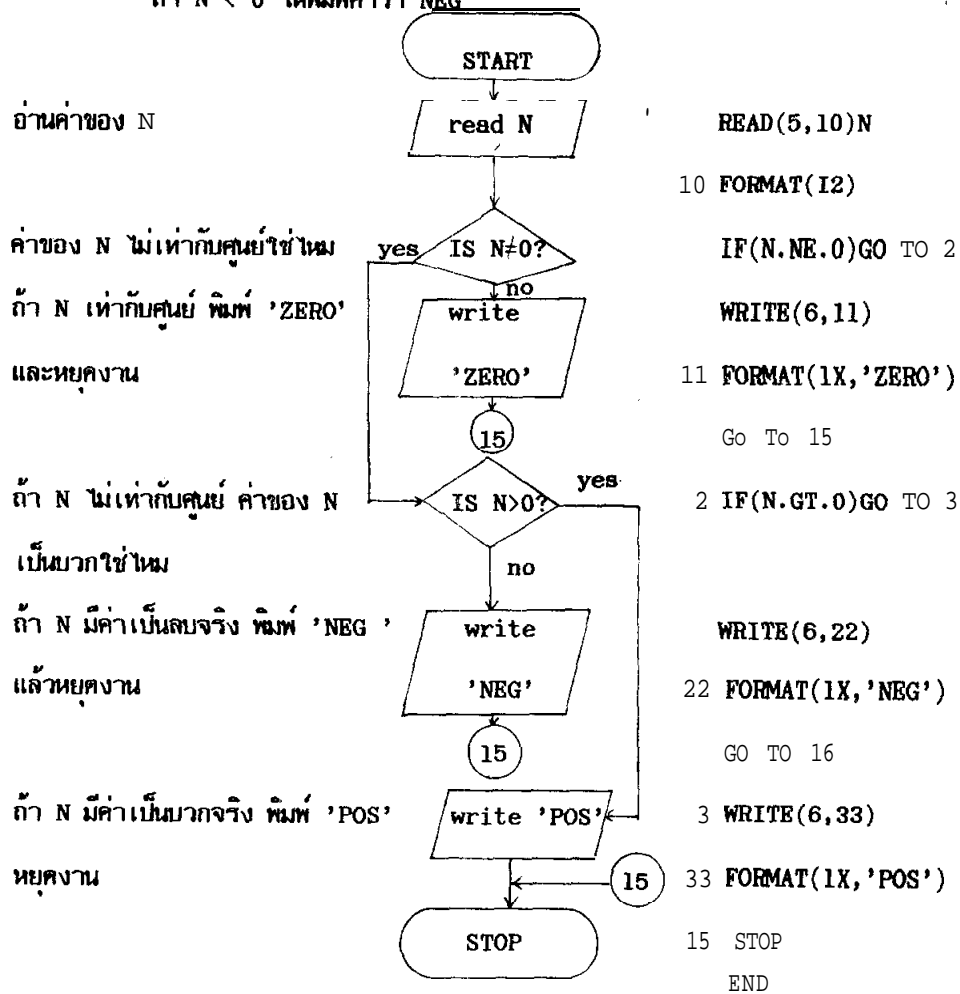
ในตัวอย่างข้างต้น เราต้องใช้คำสั่ง GO TO 5 ถ้าเราไม่มีคำสั่งนี้จะพบว่าถ้า I มีค่าเป็นเลขคี่ ค่าของ I และข้อความ 'IS ODD' และค่าของ I และข้อความ 'IS EVEN' จะถูกพิมพ์ ดังนั้นเราต้องใช้คำสั่ง GO TO 5 เพื่อข้ามคำสั่ง WRITE คำสั่งที่ 2

คำสั่ง IF ธรรมดาที่เราใช้เมื่อมีทางเลือก 2 ทางคือเลือกทางหนึ่งเมื่อเงื่อนไขเป็นจริง และเลือกอีกทางหนึ่งเมื่อเงื่อนไขไม่เป็นจริง ถ้าการกระทำเกี่ยวข้องกับการกระทำ 3 อย่าง เราอาจต้องใช้คำสั่ง IF ธรรมดา 2 คำสั่ง (ดูตัวอย่างต่อไป) ในกรณีดังกล่าว การใช้คำสั่ง IF เลขคณิต (ซึ่งจะกล่าวในหัวข้อ 2.2.5) จะสะดวกกว่า

ตัวอย่าง จงเขียนโปรแกรมเพื่ออ่านค่าของ N และทำสิ่งต่อไปนี้

ถ้า  $N > 0$  ให้พิมพ์คำว่า POS ถ้า  $N=0$  ให้พิมพ์คำว่า ZERO

ถ้า  $N < 0$  ให้พิมพ์คำว่า NEG



เราต้องใช้คำสั่ง GO TO 15 หลังคำสั่ง WRITE(6,11) และ WRITE(6,22) ส่วนหลังคำสั่งที่ 3 ไม่ต้องใช้คำสั่ง GO TO 15 เพราะคำสั่งที่ 15 อยู่ถัดไปจากคำสั่ง WRITE(6,33)ทันที  
คำสั่ง IF ตรวจจับอีกาหนึ่ง

รูปทั่วไป

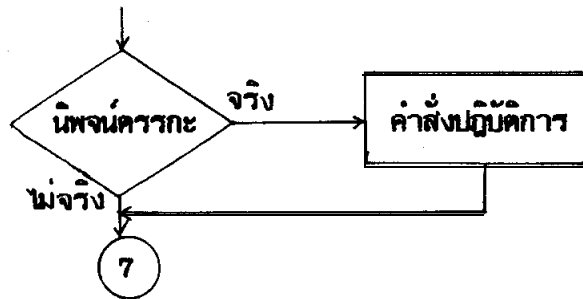
IF (นิพจน์ตรรกะ) คำสั่ง

โดยที่คำสั่งที่ตามหลังวงเล็บเป็นคำสั่งปฏิบัติการใด ๆ ยกเว้นคำสั่ง IF หรือคำสั่ง DO

ถ้าเงื่อนไขที่เขียนขึ้นด้วยนิพจน์ตรรกะเป็นจริงแล้วคำสั่งที่อยู่ในคำสั่ง IF จะถูกกระทำ แต่ถ้าเงื่อนไขไม่เป็นไปตามต้องการคือเงื่อนไขหรือนิพจน์ตรรกะเป็นเท็จ คำสั่งที่อยู่หลังคำสั่ง IF จะไม่ถูกกระทำ ไม่ว่าจะเกิดกรณีใด ๆ ใน 2 กรณี หลังจากนั้นคอมพิวเตอร์จะปฏิบัติตามคำสั่งถัดไปจากคำสั่ง IF การทำงานของคำสั่งนี้ แสดงเป็นผังภาพได้ดังนี้

IF(นิพจน์ตรรกะ)คำสั่ง

7 \_\_\_\_\_



ตัวอย่าง

5 IF(A.LT.B)K=K+1

10 IF(A.EQ.4)GO TO 1

8 IF(CODE.NE.0)WRITE(6,1)A

6 STOP

{ ถ้า  $A < B$  แล้วคำสั่ง  $K=K+1$  จะถูกกระทำ  
แล้วคำสั่งเลขที่ 10 จะถูกกระทำต่อไป  
แต่ถ้า  $A \geq B$  แล้วคำสั่งเลขที่ 10 จะถูกทำทันที

{ ถ้า  $CODE \neq 0$  แล้วจะพิมพ์ค่าของ A ก่อนจบงาน  
แต่ถ้า  $CODE=0$  เป็นอันจบงานได้ทันที

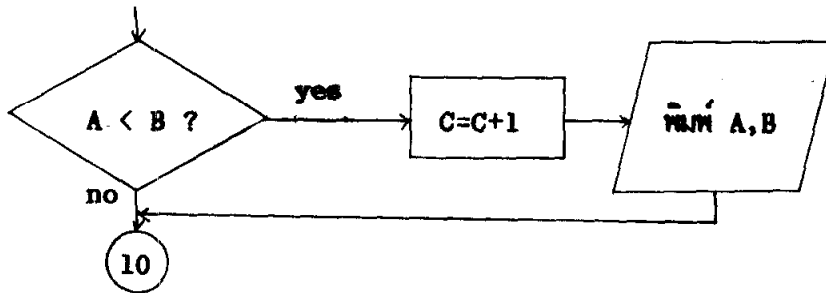
ตัวอย่างที่ไม่ถูกต้อง

7 IF(A.GT.3.1)IF(A.LT.1.)K=K+1

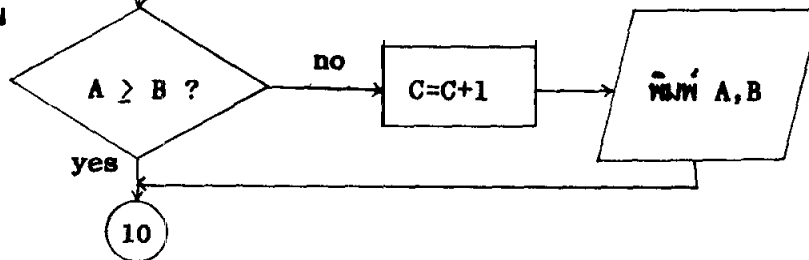
8 Y=0

คำสั่งที่ตามหลังวงเล็บเป็นคำสั่ง IF ตรวจจับเป็นคำสั่ง IF อีกไม่ได้

ในคำสั่ง IF ควบคุมจะมีคำสั่งตามมาในบรรทัดเดียวกันได้เพียง 1 คำสั่งเท่านั้น (นั่นคือคำสั่งที่จะถูกกระทำเมื่อนิพจน์ควบคุมเป็นจริง) ดังนั้นจากแผนภูมิสายงานต่อไปนี้ เราต้องทำการเปลี่ยนนิพจน์ควบคุมก่อนไขก่อนเขียนคำสั่ง IF



เปลี่ยนเป็น



ซึ่งจะเห็นได้ว่าไม่ทำให้ผลการกระทำเปลี่ยนแปลงแต่อย่างใด จากแผนภูมิสายงานใหม่เราเขียนคำสั่งได้ดังนี้

```
IF(A.GE.B)GO TO 10
```

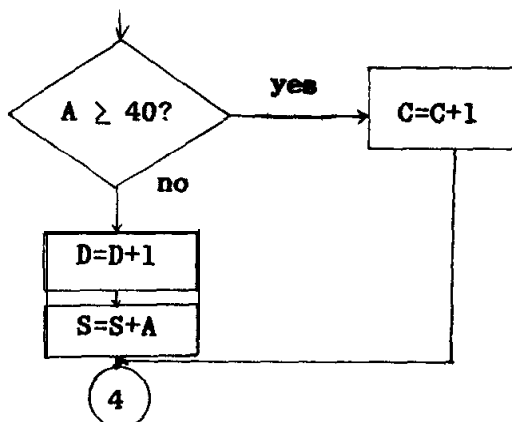
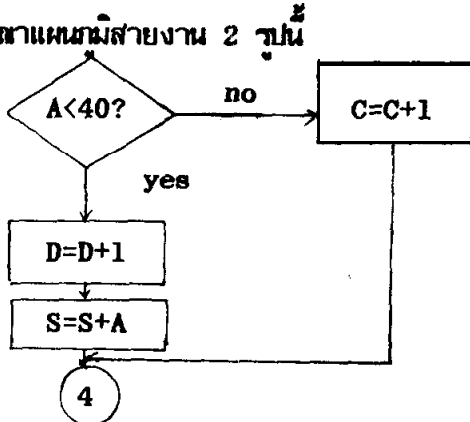
```
C=C+1
```

```
WRITE(6,1)A,B
```

```
1 FORMAT(2F10.2)
```

```
10 _____
```

พิจารณาแผนภูมิสายงาน 2 รูปนี้



ทั้งสองแผนภูมิสายงานเหมือนกัน การเขียนคำสั่ง IF ควบคุมจะเขียนจากรูปใดก็ได้

จากรูปซ้ายเขียนคำสั่งได้ดังนี้

IF(A.LT.40)GO TO 5  
C=C+1  
GO TO 4  
5 D=D+1  
S=S+A  
4 \_\_\_\_\_

จากรูปขวาเขียนคำสั่งได้ดังนี้

IF(A.GE.40)GO TO 6  
D=D+1  
S=S+A  
GO TO 4  
6 C=C+1

จากรูปขวาด้านบนเขียนคำสั่งดังต่อไปนี้

IF(A.GE.40)C=C+1  
D=D+1  
S=S+A  
4 \_\_\_\_\_

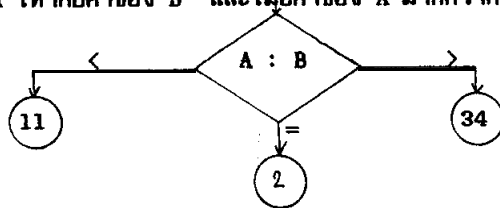
จะเห็นว่าถ้า  $A \geq 40$  จริง หลังจากทำคำสั่ง  $C=C+1$  แล้วคำสั่ง  $D=D+1$  และ  $S=S+A$  จะถูกทำ ซึ่งไม่ถูกต้องตามแผนภูมิสายงาน คำสั่งต่อไปนี้คือ

IF(A.GE.40)C=C+1,GO TO 4

ใช้ไม่ได้เพราะหลังเงื่อนไขในคำสั่ง IF ครอบคลุมให้มีคำสั่งเพียง 1 คำสั่งเท่านั้น

### 2.2.5 คำสั่ง IF เลขคณิต (Arithmetic IF statement)

คำสั่ง IF เลขคณิตนั้นทำให้เราสามารถเลือกทางปฏิบัติได้ถึง 3 ทาง ในขณะที่คำสั่ง IF ธรรมดาให้เลือกเพียง 2 ทาง ตัวอย่างเช่นในการเปรียบเทียบค่าของตัวแปร 2 ตัวคือ A และ B ทางเลือก 3 ทางขึ้นอยู่กับเงื่อนไขดังนี้คือเมื่อค่าของ A น้อยกว่าค่าของ B เมื่อค่าของ A เท่ากับค่าของ B และเมื่อค่าของ A มากกว่าค่าของ B ซึ่งแสดงได้ดังรูปต่อไปนี้





เครื่องหมาย ':' หมายความว่า 'เปรียบเทียบกับ'

จากแผนภูมิสายงานมีความหมายว่า

ถ้า  $A < B$  ให้ทำตามคำสั่งเลขที่ 11

ถ้า  $A = B$  ให้ทำตามคำสั่งเลขที่ 2

ถ้า  $A > B$  ให้ทำตามคำสั่งเลขที่ 34

แผนภูมิข้างต้น	หาคำสั่งเลขที่ 11	หาคำสั่งเลขที่ 12	หาคำสั่งเลขที่ 34
อาจเขียนดังนี้	A : B	ถ้า A < B	ถ้า A = B
เขียนเป็นภาษา			
พอร์แกรม	IF(A-B)	11	2

รูปทั่วไปของคำสั่ง IF เลขคณิตคือ

IF(นิพจน์เลขคณิต)  $n_1, n_2, n_3$

โดยที่ IF เป็นคีย์เวิร์ด

$n_1, n_2, n_3$  เป็นเลขประจำคำสั่งของคำสั่งปฏิบัติการในโปรแกรมนั้น ๆ

การทำงานตามคำสั่ง IF เลขคณิตเป็นไปตามขั้นตอนดังนี้

- 1) ทำการคำนวณเพื่อหาค่าของนิพจน์เลขคณิต
- 2) ถ้าผลลัพธ์ของนิพจน์เลขคณิตมีค่าเป็นลบ คอมพิวเตอร์จะทำตามคำสั่งเลขที่  $n_1$   
 ถ้าผลลัพธ์ของนิพจน์เลขคณิตมีค่าเป็นศูนย์ คอมพิวเตอร์จะทำตามคำสั่งเลขที่  $n_2$   
 ถ้าผลลัพธ์ของนิพจน์เลขคณิตมีค่าเป็นบวก คอมพิวเตอร์จะทำตามคำสั่งเลขที่  $n_3$

ในการเปรียบเทียบค่าของ A และ B นิพจน์ที่ใช้ในคำสั่ง IF คือ  $A - B$

ดังนั้นคำสั่ง

IF(A-B) 11, 2, 34

หมายความว่า ถ้า  $A - B < 0$  หรือ  $A < B$  แล้วให้ทำตามคำสั่งเลขที่ 11

ถ้า  $A - B = 0$  หรือ  $A = B$  แล้วให้ทำตามคำสั่งเลขที่ 2

ถ้า  $A - B > 0$  หรือ  $A > B$  แล้วให้ทำตามคำสั่งเลขที่ 34

การใช้คำสั่ง IF เลขคณิตเมื่ออาจใช้เมื่อมีทางเลือกเพียง 2 ทางเท่านั้นก็ได้

ตัวอย่างเช่น

IF(A-B)1,5,1

หมายความว่า ถ้า A=B ให้ทำตามคำสั่งเลขที่ 5

แต่ถ้า  $A \neq B$  ( $A < B$  หรือ  $A > B$ ) ให้ทำตามคำสั่งเลขที่ 1

IF(A-B)11,11,4

หมายความว่า ถ้า  $A \leq B$  ให้ทำตามคำสั่งหมายเลขที่ 11 แต่ถ้า  $A > B$  ให้ทำตามคำสั่งเลขที่ 4

ตัวอย่างคำสั่ง IF เลขคณิตที่ใช้ได้และความหมายของมัน

คำสั่ง IF เลขคณิต	ความหมาย
IF(X)1,3,5	ถ้า $X < 0$ ให้ทำตามคำสั่งเลขที่ 1, ถ้า $X = 0$ ให้ทำตามคำสั่งเลขที่ 3 และถ้า $X > 0$ ให้ทำตามคำสั่งเลขที่ 5
IF(X-(-4))1,1,2	} 2 คำสั่งนี้เหมือนกัน คือการเปรียบเทียบ X กับ -4
IF(X+4)1,1,2	
IF(X+Y*Z)3,4,3	ถ้า $X+Y*Z$ มีค่าเท่ากับศูนย์ ให้ทำตามคำสั่งเลขที่ 4 นอกนั้นให้ทำตามคำสั่งเลขที่ 3
IF(2*X+(Z-4)*X**2)1,2,3	หาค่าของนิพจน์เลขคณิตแล้วให้ทำตามคำสั่งเลขที่ 1, 2 หรือ 3 เมื่อค่าที่คำนวณได้เป็น -, 0 หรือ + ตามลำดับ

ตัวอย่างคำสั่ง IF เลขคณิตที่ไม่ถูกต้องและเหตุผลที่ทำให้คำสั่งไม่ถูกต้อง

คำสั่ง IF เลขคณิต	เหตุผล
IF(A*B)1,2	เลขประจำคำสั่งขาดไป 1 ตัว
IF(X:10)1,2,3	X:10 ไม่ใช่นิพจน์เลขคณิต

คำสั่ง IF เลขคณิต	เหตุผล
IF A-B 2,3,5	ขาดวงเล็บเปิดที่นิพจน์เลขคณิต
IF(2*(X+3)1,2,3	ขาดวงเล็บปิดที่นิพจน์เลขคณิต
IF (X=10) 4, 5, 6	X=10 ไม่ใช่นิพจน์เลขคณิต

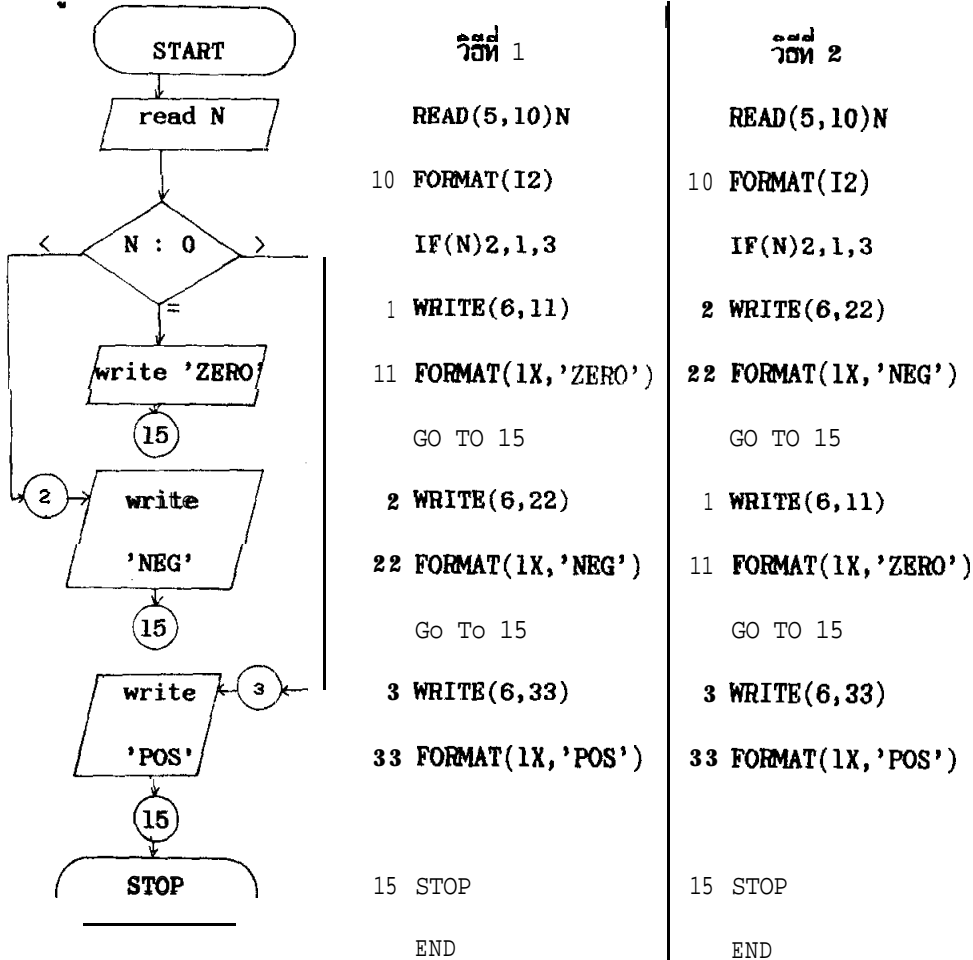
ตัวอย่างต่อไปนี้แสดงความเกี่ยวข้องของคำสั่ง IF เลขคณิตกับคำสั่งอื่น ๆ ในโปรแกรม

ตัวอย่าง อ่านเลขจำนวนเต็มเป็นค่าของตัวแปร N แล้วทำตามเงื่อนไขต่อไปนี้

จงเขียนโปรแกรมเพื่อ 1) ถ้า  $N > 0$  ให้พิมพ์ข้อความ POS, 2) ถ้า  $N = 0$  ให้พิมพ์ข้อความ ZERO

3) ถ้า  $N < 0$  ให้พิมพ์ข้อความ NEG ในตัวกลางแสดงผลลัพธ์

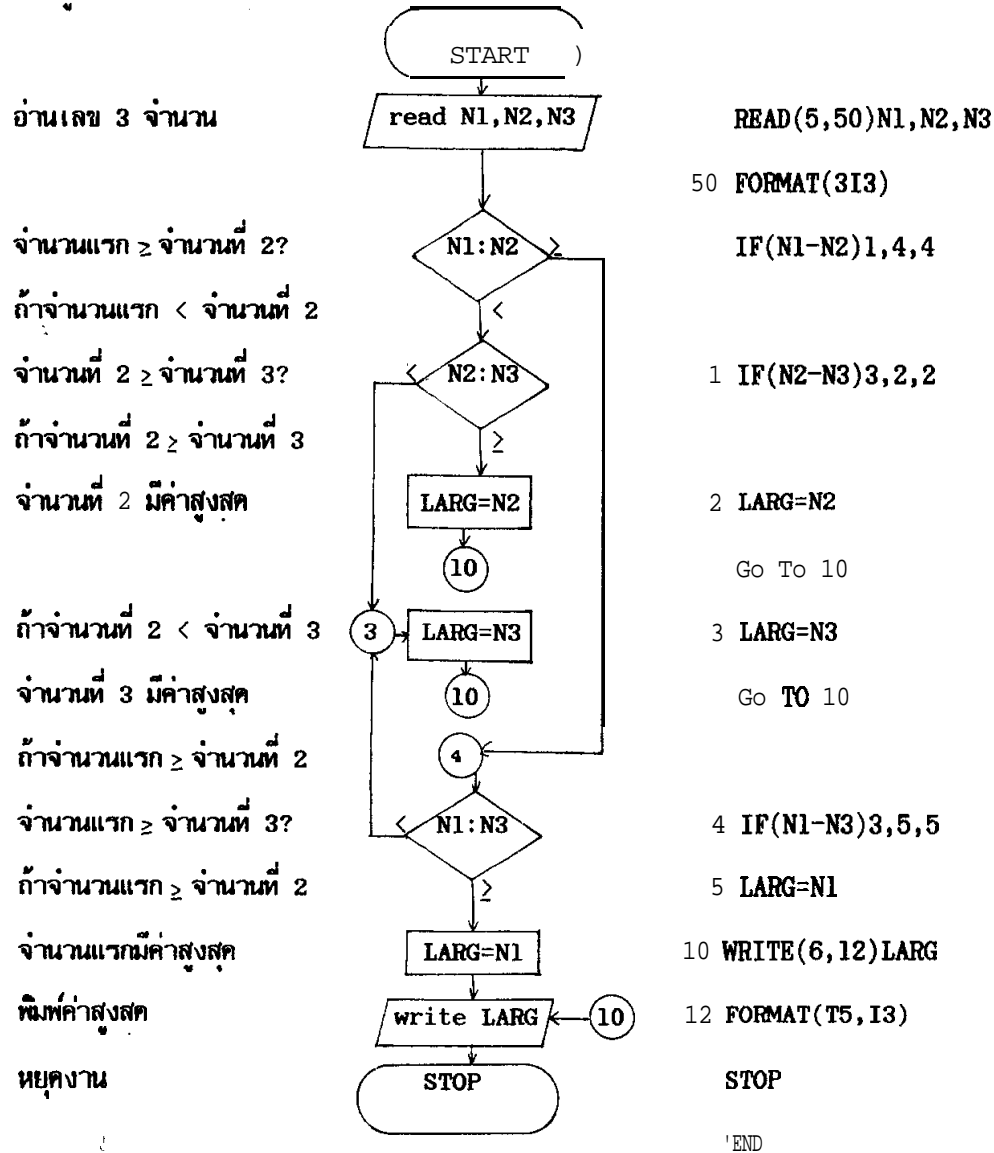
แผนภูมิสายงานและโปรแกรมคือ



ในวิธีที่ 1 ลำดับของคำสั่งเป็นไปตามแผนภูมิสายงาน วิธีนี้จะง่ายที่สุดสำหรับผู้เขียน  
 หัดเขียนโปรแกรม วิธีที่ 2 เป็นทางเลือกที่เราอาจทำได้

คำสั่ง END ใช้เป็นคำสั่งสุดท้ายในโปรแกรม จะไม่มีคำสั่งอื่นใดตามหลังมันอีก คำสั่ง  
 END เป็นคำสั่งที่บอกตัวคอมพิวเตอร์ให้หยุดแปลงคำสั่งเมื่อพบคำสั่ง END ส่วนคำสั่ง STOP นั้น  
 มีผลให้หยุดการปฏิบัติงาน ซึ่งเป็นอันจบงาน

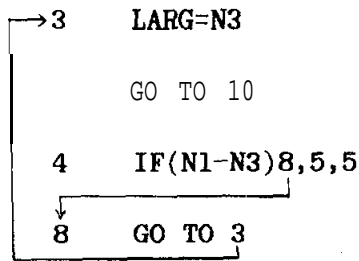
ตัวอย่าง จงเขียนโปรแกรมเพื่ออ่านเลข 3 จำนวนจากบัตรแล้วพิมพ์ค่าที่มากที่สุด  
 แผนภูมิสายงานและโปรแกรมแสดงไว้ดังนี้



จากคำสั่งเลขที่ 4 นั้นถ้า  $N1 < N3$  มันจะไปทำคำสั่งเลขที่ 3 ไปแถมเมอร์ใหม่ ๆ

อาจพยายามเขียนดังนี้

:



คำสั่ง IF นี้ควรใช้ดังนี้

4 IF(N1-N3)3,5,5

แล้วตัดคำสั่ง 8 GO TO 3 ออก

### 2.2.6 คำสั่ง BLOCK IF

ภาษาฟอร์แทรน 77 มีวิธีเขียนคำสั่ง IF วิธีใหม่ ซึ่งเรียกว่าคำสั่ง BLOCK IF (โครงสร้างรูป IF-THEN-ELSE) ซึ่งทำให้การอ่านโปรแกรมง่ายขึ้นโดยลดการใช้คำสั่ง GO TO ลง

รูปทั่วไปของโครงสร้าง IF-THEN-ELSE

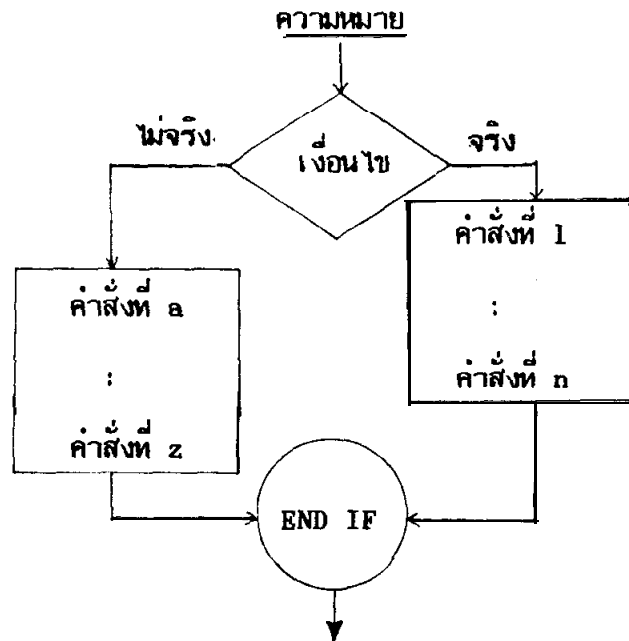
รหัสเทียม (Pseudocode)

```

IF (เงื่อนไข) THEN
  คำสั่งที่ 1
  :
  คำสั่งที่ n
ELSE
  คำสั่งที่ a
  :
  คำสั่งที่ z
END IF
  
```

block 1

block 2



โดยที่คำสั่งที่ 1,...,คำสั่งที่ n อาจเป็นคำสั่งใด ๆ ในภาษาฟอร์แทรน ยกเว้นคำสั่ง IF ครอบ  
 และ IF เลขคณิต ถ้าเงื่อนไข (condition) เป็นจริง คำสั่งที่ 1-คำสั่งที่ n (นั่นคือคำสั่ง  
 ทั้งหมดก่อนคำ ELSE) จะถูกกระทำ ถ้าเงื่อนไขเป็นเท็จ คำสั่งที่ a-คำสั่งที่ z (นั่นคือคำสั่ง  
 ระหว่างคำ ELSE และ END IF) จะถูกทำ หลังจากแต่ละกรณีดังกล่าวข้างต้นคำสั่งหลัง  
 END IF จะถูกกระทำ

ขอให้สังเกตการย่อหน้าคำสั่ง IF/ELSE และ ELSE/END IF ทั้งนี้เพื่อช่วยการอ่าน  
 คำ END IF นั้นต้องใส่ไว้ถ้าใช้ IF/THEN คีย์เวิร์ด 3 คำคือ IF, ELSE และ END IF  
 ควรเขียนให้ตรงกัน

กฎข้อหนึ่งคือคำ ELSE และ END IF จะไม่มีเลขประจำคำสั่ง ดังนั้นจะไม่มีการอ้างถึง  
 คำสั่ง ELSE และ END IF จากส่วนอื่นของโปรแกรม

ตัวอย่าง อ่านจำนวนชั่วโมงที่ทำงาน อัตราค่าจ้างต่อชั่วโมง เพื่อที่จะคำนวณเงินที่จะต้อง  
 จ่ายให้ลูกจ้าง จะพิมพ์เงินล่วงเวลาก่อนการทำล่วงเวลา

ใช้ IF-THEN-ELSE

```

READ(5,6)HRS,RATE
IF(HRS.LE.40)THEN
    REGPAY=HRS*RATE
ELSE
    OVTPAY=(HRS-40)*RATE*1.5
WRITE(6,7)OVPAY
    REGPAY=OVPAY+40*RATE
END IF
WRITE(6,8)REGPAY
    
```

ใช้ IF ครอบ

```

READ(5,6)HRS,RATE
IF(HRS.LE.40)GO TO 5
OVPAY=(HRS-40)*RATE*1.5
WRITE(6,7)OVPAY
    REGPAY=OVPAY+40*RATE
GO TO 8
5 REGPAY=HRS*RATE
8 WRITE (6,18)REGPAY
    
```

ให้สังเกตว่าไม่มีคำสั่ง GO TO เมื่อใช้ IF-THEN-ELSE ซึ่งเป็นลักษณะหนึ่งของ  
 โปรแกรมแบบโครงสร้าง

ถ้าส่วน ELSE ในคำสั่ง BLOCK IF ไม่มีก็อยู่ในรูปดังนี้

IF (เงื่อนไข) THEN

คำสั่งที่ 1

คำสั่งที่ 2

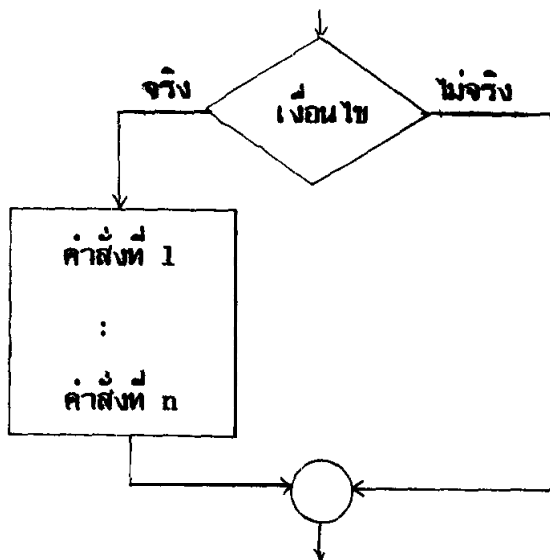
⋮

คำสั่งที่ J

คำสั่งที่ n

END IF

ภาพปฏิบัติการงานจะเป็นไปตามแผนภูมิสายงานต่อไปนี้



นั่นคือถ้าเงื่อนไขเป็นจริงคำสั่งที่ 1 ถึงคำสั่งที่ n จะถูกทำ และต่อจากนั้นจะทำตามคำสั่งที่อยู่ถัดจากคำสั่ง END IF

ตัวอย่าง ถ้ายอดขายเกิน \$10,000 เพิ่มเงินโบนัส (BONUS) ให้ 10% ของยอดขาย

(SALES) ให้นำ PAY ถ้า SALES > \$10,000 BONUS เป็นศูนย์ ทั้งนี้ไม่ว่า  
กรณีใด ๆ ให้พิมพ์ค่า PAY ด้วย > เท่ากับหรือ

แบบไม่ใช้โครงสร้าง

```

BONUS=0
IF(SALES.LE.10000)GO TO 7
BONUS=.1*SALES
WRITE(6,5)BONUS
7 PAY=HRS*RATE+BONUS
WRITE(6,6)PAY
หรือ
BONUS=0
IF(SALES.GT.10000)GO TO 6
GO TO 7
6 BONUS=.1*SALES
WRITE(6,5)BONUS
7 PAY=HRS*RATE+BONUS
WRITE(6,6)PAY

```

แบบโครงสร้าง

```

BONUS=0
IF(SALES.GT.10000)THEN
  BONUS=.1*SALES
  WRITE(6,5)BONUS
END IF
PAY=HRS*RATE+BONUS
WRITE(6,6)PAY

```

Nested IF/THEN structures

ในปัญหาหลาย ๆ ปัญหา การตัดสินใจขึ้นอยู่กับเงื่อนไขมากกว่า 1 คู่ขนานไป  
พิจารณาปัญหาคือ โบนัส พนักงานขายจะได้รับเงินตอบแทนจากการขายดังนี้

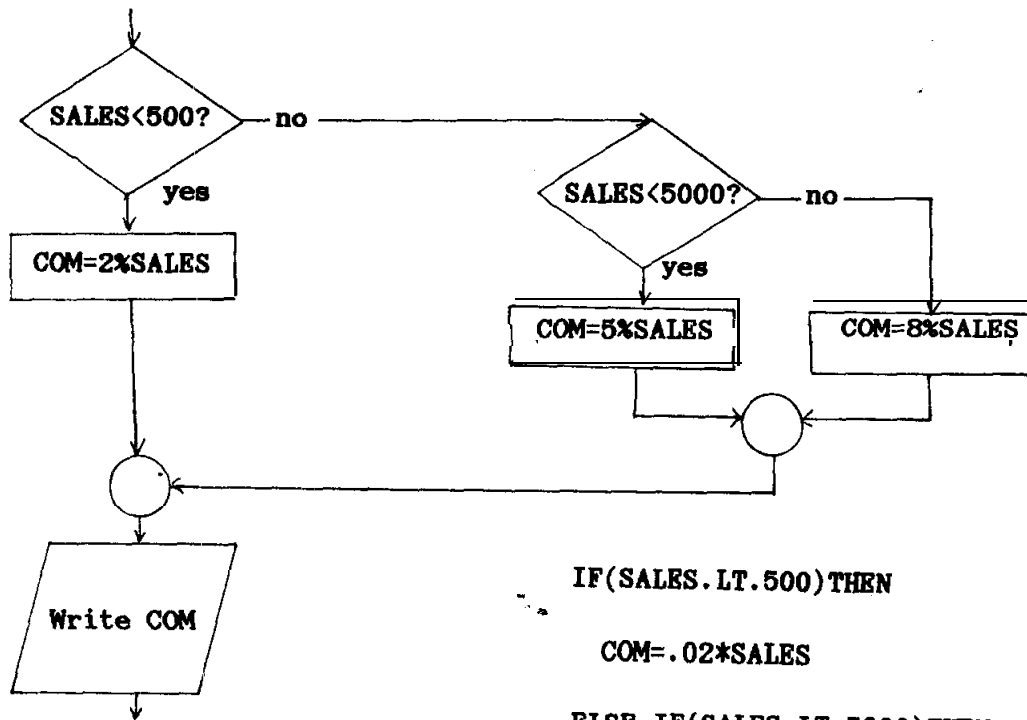
ยอดขาย	ค่าตอบแทน (% ของยอดขาย)
ต่ำกว่า \$500	2%
\$500 และต่ำกว่า \$5000	5%
\$5000 และสูงกว่านั้น	8%

วิธี 2 วิธีที่ใช้ในการแก้ปัญหานี้ ซึ่งแต่ละวิธีทำให้เกิด nested structure ต่างกัน

2 แบบคือ



กรณี 1 ใช้ IF-THEN-ELSE



IF(SALES .LT. 500) THEN

COM = .02\*SALES

ELSE IF(SALES .LT. 5000) THEN

COM = .05\*SALES

ELSE

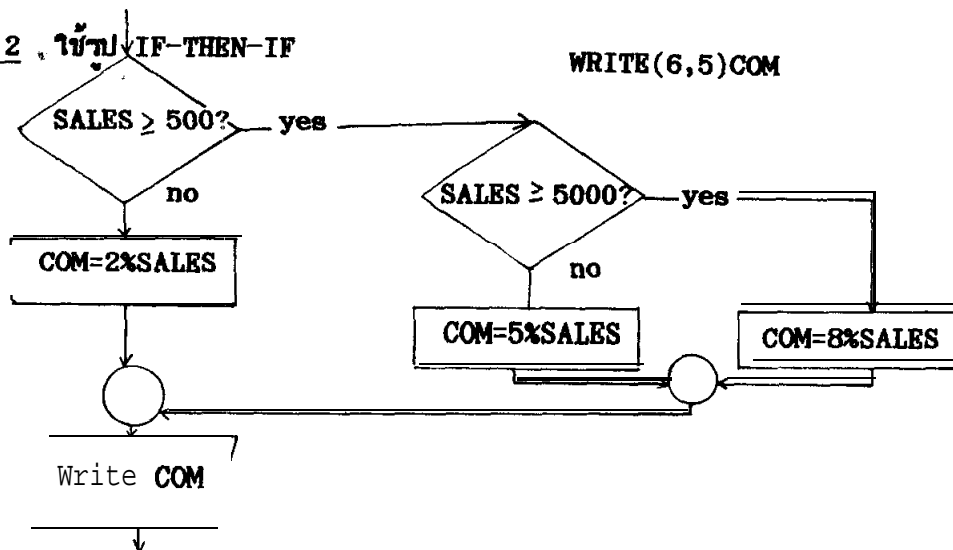
COM = .08\*SALES

END IF

END IF

WRITE(6,5)COM

กรณี 2 ใช้ IF-THEN-IF



```

IF(SALES.GE.500) THEN
    IF(SALES.GE.5000) THEN
        COM=.08*SALES
    ELSE
        COM=.05*SALES
    END IF
ELSE
    COM=.02*SALES
END IF
WRITE(6,5)COM

```

### 2.3 โปรแกรมตัวอย่าง

#### 2.3.1 รากของสมการกำลังสอง

จงเขียนโปรแกรมเพื่อหารากที่ 2 ของสมการโพลิโนเมียล  $ax^2+bx+c=0$  โดยให้อ่านค่าสัมประสิทธิ์ a, b, c เป็นข้อมูลเข้า

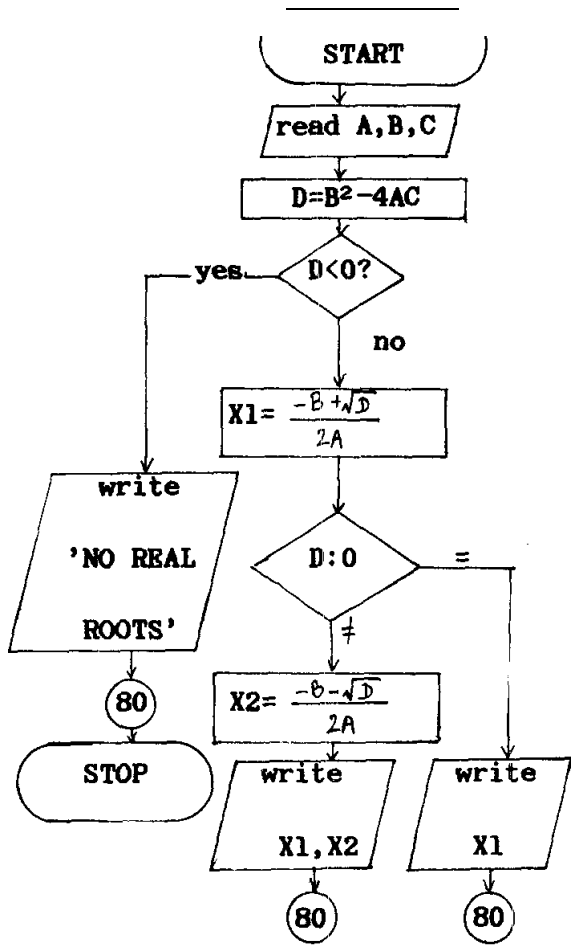
ถ้าหารากจริงไม่ได้ (เมื่อ  $b^2-4ac < 0$ ) ให้พิมพ์ข้อความ NO REAL ROOTS

ถ้ารากทั้ง 2 เท่ากัน ให้พิมพ์ข้อความว่า BOTH ROOTS ARE EQUAL TO \_\_\_\_

ถ้ารากทั้ง 2 ไม่เท่ากัน ให้พิมพ์ ROOT1= \_\_\_\_ ROOT2= \_\_\_\_

รากทั้ง 2 ของสมการ  $ax^2+bx+c=0$  คือ

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad , \quad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$



```

READ(5,10)A,B,C
10 FORMAT(F5.1,F5.2,F5.3)
D=B*B-4.*A*C
IF(D.LT.0)GO TO 75
X1=(-B+D**.5)/(2.*A)
IF(D.EQ.0)GO TO 85
X2=(-B-D**.5)/(2.*A)
WRITE(6,11)X1,X2
GO TO 80
65 WRITE(6,12)X1
GO TO 80
75 WRITE(6,13)
80 STOP
13 FORMAT(T5,'NO REAL ROOTS')
11 FORMAT(T5,'ROOT1=',F8.2,
*'ROOT2=',F8.2)
12 FORMAT(T5,'BOTH ROOTS ARE EQUAL',
*'TO',F8.2)
END

```

### 2.3.2 ปัญหาดอกเบี้ยทบต้น

จงเขียนโปรแกรมอ่านอัตราดอกเบี้ย (R) เงินต้น (P) และจำนวนปี (N) ถ้าเงินต้นและดอกเบี้ยที่ได้ไม่ถูกถอนออกเลย จงคำนวณเงินที่จะได้หลังจาก N ปีแล้ว (A)

$$\text{กำหนดสูตร } A = P(1+R)^N$$

Input data

R	P	N
.115	100000	30
F4.0	F6.2	I2

output

COMPOUND INTEREST PROBLEM	
	} เงิน 1 บาท
INTEREST RATE =	0.115
PRINCIPAL AMOUNT =	1000.00
NUMBER OF YEARS	30
	3 เงิน 1 บาท
PRINCIPAL AMOUNT INVEST AT	0.116
WILL HAVE A VALUE OF	26196.70

```

READ(5,10)R,P,N
10 FORMAT(F4.0,F6.2,I2)
WRITE(6,11)
11 FORMAT('1',T50,'COMPOUND INTEREST PROBLEM')
WRITE(6,12)R
12 FORMAT('0',T40,'INTEREST RATE    =',T69,F9.3)
WRITE(6,13)P
13 FORMAT(T40,'PRINCIPAL AMOUNT. =',T69,F8.2)
WRITE(6,14)N
14 FORMAT(T40,'NUMBER OF YEARS',T69,I5)
A=P*(1+R)**N
WRITE(6,15)R
15 FORMAT('0',T40,'PRINCIPAL AMOUNT INVEST AT',T69,F9.3)
WRITE(6,16)A
16 FORMAT(T40,'WILL RAVE A VALUE OF',T69,F8.2)
STOP
END

```

## 2.4 คำสั่งอื่น ๆ ในภาษาฟอร์แทรน

### 2.4.1 Coment line

บางครั้งโปรแกรมที่เขียนมีความยาวมาก โปรแกรมเมอร์จะพบว่า การเขียนอธิบาย ส่วนของโปรแกรมจะช่วยให้ตนเองเข้าใจเมื่ออ่านโปรแกรมนั้น ๆ ภายหลัง เพื่อการปรับปรุง โปรแกรมหรือนำโปรแกรมไปใช้ หรือช่วยให้ผู้ที่ใช้โปรแกรมเข้าใจโดยโปรแกรมเมอร์ผู้เขียน โปรแกรมนั้น ๆ ไม่ต้องคอยให้คำอธิบายเพิ่มเติม

เราอาจใส่ comment line ในที่ใด ๆ ก็ได้ในโปรแกรมโดยใส่ตัวอักษร C ที่

ส่วนนำแรกของบรรทัด ข้อความใน comment line นี้จะไม่ถูกแปลเป็นภาษาเครื่องเลย ค่อยไป  
เป็นตัวอย่างของการใช้ comment กับโปรแกรมตัวอย่างในข้อ 2.3.1

C2345678901234

```
C      QUADRATIC EQUATION
C
C      READ THREE COEFFICIENTS
C
C      READ(5,10)A,B,C
C
C      COMPUTE DISCRIMINANT
C
C      D=B*B-4.*A*C
C
C      IF(D.LT.0)GO TO 75
C
C      X1=(-B+D**.5)/(2.*A)
C
C      DETERMINE IF ROOTS ARE EQUAL
C
C      IF(D.EQ.0)GO TO 65
C
C      ROOTS ARE UNEQUAL
C
C      X2=(-B-D**.5)/(2.*A)
C
C      :
C
C      ROOTS ARE EQUAL
C
C      :
C
C      STOP
C
C      END
```

#### 2.4.2 คำสั่ง COMPUTED GO TO

คำสั่งนี้จะช่วยให้สะดวกในการเปลี่ยนทิศทางไปจุดต่าง ๆ ในโปรแกรมโดยการใช้  
คำสั่งเพียง 1 คำสั่งเท่านั้นคือ คำสั่ง **COMPUTED GO TO**

รูปทั่วไป

GO TO (no<sub>1</sub>, no<sub>2</sub>, ..., no<sub>n</sub>), ตัวแปร

โดยที่ ตัวแปรต้องเป็นตัวเลขชนิด integer

no<sub>1</sub>, no<sub>2</sub>, ..., no<sub>n</sub> เป็นเลขประจำคำสั่งของคำสั่งปฏิบัติการในโปรแกรมนี้ ๆ

ถ้าค่าของตัวแปรเป็น 1 คำสั่งเลขที่ no<sub>1</sub> จะถูกทำ (GO TO no<sub>1</sub>)

ถ้าค่าของตัวแปรเป็น 2 คำสั่งเลขที่ no<sub>2</sub> จะถูกทำ (GO TO no<sub>2</sub>)

:

ถ้าค่าของตัวแปรเป็น n คำสั่งเลขที่ no<sub>n</sub> จะถูกทำ (GO TO no<sub>n</sub>)

ตัวอย่าง GO TO (3,57,100,4),N

ถ้า N=1 , GO TO 3

ถ้า N=2 , GO TO 57

ถ้า N=3 , GO TO 100

ถ้า N=4 , GO TO 4

ถ้าค่าของตัวแปรน้อยกว่า 1 หรือเกิน n คอมพิวเตอร์จะข้ามไปทำคำสั่งที่อยู่ถัดจาก

คำสั่ง GO TO

สิ่งที่ต้องระวังคือตัวแปรในคำสั่ง GO TO นั้นจะต้องเป็นตัวเลขชนิด integer จะใช้

นิพจน์ในตำแหน่งของตัวแปรไม่ได้ เช่น

GO TO (2,13,4,5),2\*I

ใช้ไม่ได้เพราะ 2\*I ไม่ใช่ตัวแปร (ในภาษาฟอร์แทรน 77 เน้นยอมให้ใช้นิพจน์เลขคณิตได้)

ตัวอย่าง ต้องการพิมพ์ความหมายของรหัสของชั้นปีที่ศึกษาอยู่ ซึ่งอ่านจากบัตร

ความหมายของรหัสคือ

<u>รหัสของชั้นปี</u>	<u>ความหมาย</u>
1	Freshman
2	Sophomore
3	Junior

<u>รหัสของคีย์</u>	<u>ความหมาย</u>
4	Senior
5	Graduate

### โปรแกรมอาจเป็นดังนี้

```

READ(5,6)KCODE
6  FORMAT(I1)
   GO TO (15,20,25,30,35),KCODE
   WRITE(6,100)KCODE
100 FORMAT(3X,'INVALID CODE',I3)
    GO TO 106
    15 WRITE(6,101)
101  FORMAT(3X,'FRESHMAN')
    GO TO 106
    20 WRITE(6,102)
102  FORMAT(3X,'SOPHOMORE')
    GO TO 106

106  STOP
    END

```

### 2.4.3 แพคเคอร์แสดงการซ้ำ

ในการอ่านค่าของตัวแปรหลาย ๆ ตัวนั้น ทุกตัวอาจใช้ format code เดียวกัน

เช่น

```

READ(5,101)X,Y,Z
101 FORMAT(F4.0,F4.0,F4.0)

```



เราอาจเขียนคำสั่ง FORMAT ใหม่ได้ดังนี้

101 FORMAT(3F4.0)

ตัวอย่าง

สพท	1	2	3	4	5	6	7	8	9	10	11
บัตรข้อมูล			1		9	2	2	3	3		

READ(5,5) I, J, K, L, M

5 FORMAT(2I2, I1, 2I2)

ผลกาอ่าน I=0

J=10

K=9

L=22

M=33

ตัวอย่าง WRITE(6,5) A, J, B, K, L, M, N

5 FORMAT(3X, 2(F3.1, I2), 3(2X, I1))

คำสั่ง FORMAT ข้างต้นจะเหมือนกับ

5 FORMAT(3X, F3.1, I2, F3.1, I2, 2X, I1, 2X, I1, 2X, I1)

#### 2.4.4 การต่อบรรทัด

คำสั่งในภาษาฟอร์แทรนโดยเฉพาะคำสั่ง FORMAT มักจะยาวเกินที่จะเขียนในสพท 7-72 ได้หมด เราสามารถเขียนต่อในบรรทัดถัดไปได้ โดยการใส่ตัวอักษรระเค ๗ (นอกจากช่องว่างและเลขศูนย์) ในสพท 6 ของบรรทัดที่เป็นบรรทัดต่อหบบรรทัด

ตัวอย่าง คำสั่งที่เหมือนกัน

101 FORMAT(3X, 'AREA LOT1 IS', 2X, F8.1, 3X, 'AREA LOT2 IS', 2X, F8.1)

หรือเขียน

```
101 | FORMAT(3X, 'AREA LOT1 IS', 2X, 2X, F8.1,
```

```
    | *3X, 'AREA LOT2 IS', 2X, F8.1)
```

ตัวอย่าง    คำสั่งที่ 100 และ 101 เหมือนกัน

```
100 | X=Y+
```

```
    | 1Z+
```

```
    | 2Q
```

```
101 | X=Y+Z+Q
```

ตัวอย่าง    แสดงการต่อบรรทัดในคำสั่ง FORMAT ซึ่งต้องระมัดระวังเกี่ยวกับ literal format

```
    | WRITE(6,11)
```

```
11 | FORMAT(...., 'I DREAM OF THINGS THAT ^^^^^
```

      ↓  
      สพมภ์ 72

```
    | INEVER WERE')
```

จะ ได้ผลการทำงานดังนี้

... I DREAM OF THINGS THAT ^^^^^ NEVER WERE

      ↓  
      สพมภ์ 72

แต่ถ้าใช้

```
12 | FORMAT(..., 'I DREAM OF TRINGS THAT NEVE
```

```
    | *R WERE')
```

จะ ได้ผลการทำงานดังนี้

... I DREAM OF THINGS TRAT NEVER WERE

หรือถ้าใช้

```
13 | FORMAT(...., 'I DREAM OF THINGS THAT',
```

```
    | *'NEVER WERE')
```

จะ ได้ผลเช่นเดียวกับการใช้คำสั่ง FORMAT ที่ 12

## แบบฝึกหัดที่ 2

1. คำสั่ง READ และคำสั่ง FORMAT ของมัน เป็นคำสั่งที่ถูกต้องหรือไม่ ถ้าไม่ถูกต้องจงบอก

เหตุผล

1.1) READ(5,10),A,B,C

1.2) READ(5,5)I,J

10 FORMAT(F1.0,F2.0,F3.0)

5 FORMAT(I1,I4)

1.3) READ(5,6)A,I,J

1.4) READ(5,7)A,B,C

6 FORMAT(F4.5,I2,I1)

7 FORMAT(F5.2,I3,F4)

1.5) READ(5,8)X,J,Y

1.6) READ(5,5)X,I

8 FORMAT(F3.1,F2.1,I4)

5 FORMAT(T5,F5.1,T2,I2)

2. จากบัตรข้อมูลคำสั่ง READ และคำสั่ง FORMAT ที่กำหนดให้ จงบอกค่าของ I, J, K เมื่อ

ใช้คำสั่ง FORMAT ต่างกันไป

สมุดที่ 1

13,5.2,,,,-467.19

I - READ(5,5)I,J,K

2.1) 5 FORMAT(T1,I1,I1,I1)

2.2) 5 FORMAT(T2,1X,I1,1X,I2,I5)

2.3) 5 FORMAT(I3,I1,T6,I1)

2.4) 5 FORMAT(I4,I3,I5)

2.5) 5 FORMAT(T8,I1,I4,I1)

2.6) 5 FORMAT(I2,T1,I1,T2,I4)

2.7) 5 FORMAT(2(1X,I1),T10,I3)

3. จากบัตรข้อมูลคำสั่ง REM และคำสั่ง FORMAT ที่กำหนดให้ จงบอกค่าของ A, I, S

สมุดที่ 1

1.3,25.1.5

READ(5,7)A,I,S

3.1) 7 FORMAT(F4.0,I2,3X,F1.1)

3.2) 7 FORMAT(T3,F4.3,1X,I1,F3.1)

3.3) 7 FORMAT(2X,F2.1,I3,F3.0)

3.4) 7 FORMAT(T2,F6.1,I1,F2.2)

3.5) 7 FORMAT(T2,F1.1,I2,F2.0)

4. จงแสดงผลการพิมพ์จากคำสั่ง WRITE ที่กำหนดให้

4.1) จากบัตรข้อมูล  $\sqrt{34312 \cdot 5621}$

READ(5,10)A,J,X,Z

10 FORMAT(1X,F3.2,T6,I3,F4.2,F3.0)

WRITE(6,11)X,A,J,Z

11 FORMAT(T3,F4.1,2X,F3.1,1X,I3,F2.0)

4.2) จากบัตรข้อมูล  $\sqrt{2.439-1.3}$

READ(5,10)A,I,J,C

10 FORMAT(T2,F4.1,2X,I2,I1,F4.4)

C=2\*C

WRITE(6,11)C,J,I,A

11 FORMAT(T10,'RES=',F4.0,1X,2I3,F4.2)

5. จงบอกผลการทำตามคำสั่ง IF ต่อไปนี้ในแต่ละชุดของค่า A, B, C

ชุดที่ 1 : A=-3., B=-2., C=2.

ชุดที่ 2 : A=-4., B=-5., C=1.

5.1) IF(B\*\*2.LE.4\*A\*C)GO TO 4

R=1

4 R=2

5.2) **IF(A.GT.B)WRITE(6,1)A**

GO TO 3

5.3) **IF(A.LE.0)A=-A**

**WRITE(6,1)A**

5.4) **IF(B.GE.C--4)STOP**

C=0

**WRITE(6,1)C**

5.5) **IF(A.NE.C\*3)WRITE(6,1)C**

**WRITE(6,2)A**

6. จงหาข้อผิดพลาดในคำสั่งต่อไปนี้

6.1) **IF(I=N)GO TO 15**

6.2) **IF(X+Y.GT.3.4)X.EQ.2**

6.3) **IF G.GT.3 GO TO 41**

6.4) **IF(X+Y)P=G\*\*(5)**

6.5) **IF(A.LE.(A+B)P=Q**

6.6) **7 IF(17.LE.B)GO TO 7**

6.7) **IF(A.GE.7),WRITE(6,1)A**

6.8) **IF 3\*A.EQ.B,,A=A+1**

6.9) **IF(7.3+A.GT.2+B)END**

6.10) **IF(A.LT.B)IF(A.EQ.B)STOP**

6.11) **IF(A.LT.B)WRITE(6,1)A,GO TO 3**