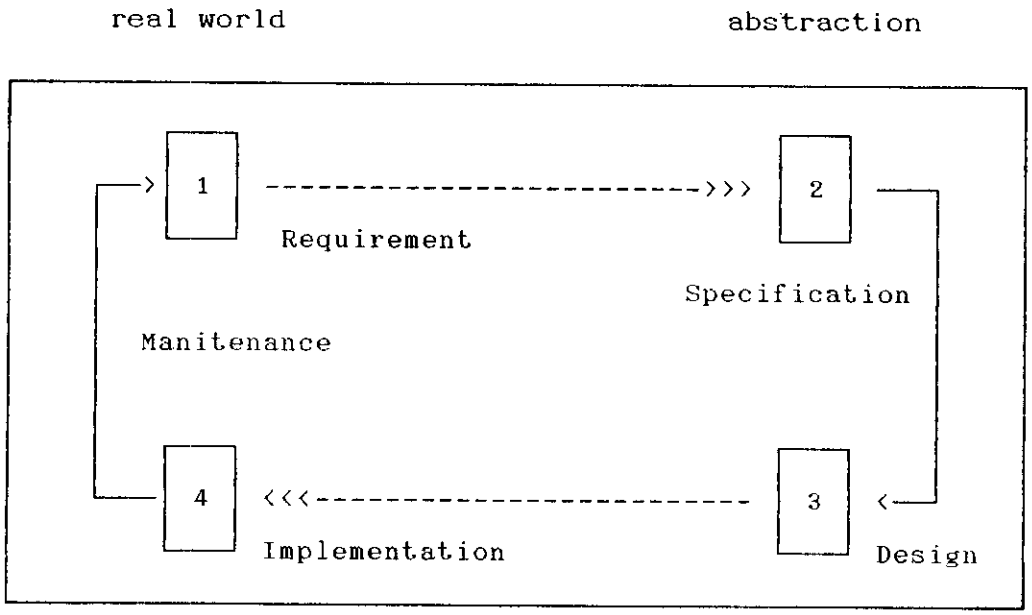
# บทที่ 2
# ขั้นตอนการสร้างโปรแกรม

จากรูปต่อไปนี้เราจะเห็นว่าชีพจักรของโปรแกรมนั้นเป็นอย่างไร




รูปที่แสดงนี้จะเริ่มจากจุดที่ 1 ซึ่งหมายถึง สภาวะแวดล้อมในสังคมที่เกิดขึ้นจริง แล้วจึงนำสภาพนั้นมากำหนดเป็นคุณลักษณะของงานที่จะดำเนินการ จากนั้นจึงนำมาออกแบบแล้วจึงนำแบบที่ได้ไปสร้างเป็นโปรแกรมเพื่อปฏิบัติงานต่อไป ภายหลังเมื่อสร้างเป็นโปรแกรมแสร็จแล้วจะต้องมีการคอยดูแลรักษาโปรแกรมจนกว่าจะพบปัญหาใหม่ที่เกิดขึ้นในสังคมโดยที่โปรแกรมนั้นไม่อาจจะปรับใช้ได้อีกแล้ว ก็จะมีการสร้างโปรแกรมขึ้นมาใหม่ทดแทนโปรแกรมที่ต้องทิ้งไป ดังนั้นเพื่อเป็นการลดภาระในการต้องสร้างโปรแกรมใหม่ขึ้นทดแทนโปรแกรมบ่อยๆ นักออกแบบโปรแกรมควรจะคำนึงถึงความต้องการ(Requirement) ในขั้นที่ 1 ของผู้ใช้ด้วยว่าในอนาคตจะมีการเปลี่ยนแปลงอย่างไรบ้าง เราจะออกแบบโปรแกรมโดยจำกัดความต้องการสภาวะใน ปัจจุบันไม่ได้แต่ต้องวางแผนในอนาคตด้วยทั้งนี้เพื่อจะช่วยให้ "ชีพจักร" ของโปรแกรมนั้นยาวนานต่อไป

**ขั้นตอนของการเขียนโปรแกรมประกอบด้วย**

**1.การแยกแยะปัญหาความต้องการของบรรดาผู้ใช้โปรแกรม**

(Defining the Problem)

ดังได้กล่าวมาแล้วข้างต้นว่าการจะพิจารณาแยกแยะปัญหานั้นอย่าได้มองเฉพาะสภาวะปัจจุบันเท่านั้น ต้องมองเผื่อไปในอนาคตอาจจะทำแค่ระยะสั้น (Short - term) คือประมาณ 1 - 3 ปี ตัวอย่างของการวางแผนไปในอนาคตเช่น กำหนดขนาดของข้อมูลของลูกค้าในอนาคตซึ่งอาจจะมีขนาดใหญ่ขึ้น หรือการกำหนดทางเลือกของการดำเนินกิจกรรมบางอย่าง เช่น การคิดภาษีที่มีเงื่อนไขมาก ขึ้นกว่าเท่าที่ปรากฏในระบบปัจจุบัน

**ขั้นตอนการแยกแยะปัญหานั้นจะต้องคำนึงถึงปัจจัยต่อไปนี้**

1.1 **Input Specification** ซึ่งในส่วนของ Input Specification จะต้องดำเนินการในเรื่องต่อไปนี้

(a.) มีการรับข้อมูลจากแหล่งใดใน 2 แหล่งคือ

- Internal data หมายถึง การกำหนด (assign) ค่าให้กับตัวแปรภายในโปรแกรมเลย เช่น X:= 7.5 เป็นต้น

- External data เป็นการรับข้อมูลจากภายนอก ซึ่งโดยกระบวนการนี้เราจะจำแนกอุปกรณ์ได้ 2 แบบ คือ

**แบบที่** 1 อุปกรณ์คือ แป้นพิมพ์ ซึ่งโดยปกติแล้วในบางภาษามักจะถูกกำหนดโดยปริยาย(set default) ไว้ว่า ถ้าเป็นการ read ข้อมูลจากข้างนอกแล้วจะเป็นแป้นพิมพ์อยู่แล้วการรับ ข้อมูลโดยวิธีนี้ไม่ค่อยจะมีขั้นตอนยุ่งยากมากนัก เพียงแต่ขอให้รับรู้ว่าโครงสร้างของคำสั่ง read ในการรับข้อมูลมีลักษณะใดตัวอย่างการใช้คำสั่งรับข้อมูลจากแป้นพิมพ์มีดังนี้

**ตัวอย่างในภาษาเบสิค**

```
10  CLEAR : CLS
20  INPUT A,B
30  ...
40  ...
..  ...
999 END
```

คำสั่ง INPUT ของภาษาเบสิคก็คือการรับข้อมูลมาจากข้างนอก คือแป้นพิมพ์

**ตัวอย่างในภาษาฟอร์แทรน**

```
     READ(*,10) X,Y,I
10   FORMAT (F5.2, F3.1, I2)
     .............
     .............
     STOP
     END
```

โดยที่ default ของอุปกรณ์ "*" บน "Fortran 77" คือ แป้นพิมพ์

**ตัวอย่างในการใช้ภาษาปาสคาล**

```
PROGRAM  EXAM (INPUT,OUTPUT);
CONST    PI = 3.1416;
VAR      CIR, DIAMETER : REAL;
BEGIN
     READ(DIAMETER);
     CIR := PI*DIAMETER;
     WRITELN('CIRCUMSTANT IS ',CIR)
END.
```

โดยที่ default ของ write หรือ writeln ใน Turbo Pascal คือแป้นพิมพ์

**แบบที่ 2** อุปกรณ์คือ ตัวกลางบันทึกข้อมูลต่างๆ เช่น diskette , disk, tape ในกรณีของข้อมูลที่อยู่ในตัวกลางเหล่านี้เราจำเป็นที่จะต้องมีการใช้คำสั่งที่จัดการเรื่องแฟ้มข้อมูลเข้ามา **เกี่ยวข้องว่า** จะปฏิบัติการเช่นเดียวกับการรับข้อมูลจากแป้นพิมพ์ที่ไม่ได้ กรรมวิธีของการจัดการแฟ้มข้อมูล ลักษณะเช่นนี้จะต้องมีคำสั่งลักษณะนี้คือ

```
open file
   read data from file
   check end of file
close file.
```

(b) การรับข้อมูลเข้าไปปฏิบัติการภายในโปรแกรมนอกเหนือจากคำนึงถึงข้อ "a."แล้วยังต้องคำนึงถึงรายละเอียดในโครงสร้างของข้อมูลคือ "format" ว่าค่าที่จะรับเข้าไปนั้นมี รายละเอียดดังต่อไปนี้ คือ sequence, spacing, accuracy, unit เป็นเช่นใด

(c) ขอบเขตของข้อมูลเข้าไปเป็นอย่างไร(valid range of values)

(d) คุณลักษณะหรือข้อจำกัดของข้อมูลเป็นเช่นใด การใช้ข้อมูลดังกล่าวสามารถจะไปปรับแก้ไขในอนาคตเมื่อมีการปรับแก้โปรแกรมได้อย่างไร เราจะละเลยข้อมูลดังกล่าวได้หรือไม่ ในกรณีของการปรับแก้โปรแกรมต่อไปในอนาคต

(e) เราจะสามารถรับรู้กรณีของการที่เราประสบปัญหาที่ข้อมูลสูญหายได้หรือไม่

1.2 Output Specification

เราจะต้องทราบถึงลักษณะของสารสนเทศหรือ "output data" ในรายละเอียดต่อไปนี้

(a) ความต้องการของ output ที่จะผลิตออกมาว่าคืออะไร

(b) อธิบาย format ของค่าที่จะแสดงออกมาประกอบด้วย significant digits, decimal accuracy, units, และ location on the page

(c) รูปแบบของรายงานต้องมีหัวเรื่อง รายการย่อ และรายละเอียดอย่างอื่นในรายงานหรือไม่

(d) output reportนั้นสมควรจะปรับบางรายการให้การพิมพ์ย่อลงแต่ยังคงความหมายได้เช่นเดิมทั้งนี้เพื่อประหยัดเวลาในการพิมพ์

**1.3 การประมวลผลกณีย์พิเศษ**

(Spacial Processing)

จะต้องมีการแจกแจง special condition ทั้งหลายที่อาจจะเกิดขึ้นในการปฏิบัติงาน

**2. การกำหนดนิยามในแนวทางแก้ไข**

(Outlining the Solution)

ในขั้นตอนนี้จะช่วยนำทางแก้ไขปัญหาที่เกิดขึ้นในข้อ 1 ขั้นตอนนี้มีเจตนาเพื่อที่จะกำหนดแนวทางที่จะดำเนินงาน

**3.การเลือกขั้นตอนดำเนินการที่ดีและมีประสิทธิภาพ**

(Seleting and representing algorithm)

เราต้องยอมรับว่าการเลือกวิธีการดำเนินงานถ้าเป็นงานในเรื่องต่างๆ ไม่เฉพาะแต่ในเรื่องของโปรแกรมเท่านั้นแม้กระทั่งในเรื่องอื่นๆ ก็ตามล้วนแต่มีหลายหนทางในวิธีการดำเนินงานโดยที่แต่ละหนทางสามารถจะพบจุดมุ่งหมายเดียวกันได้ ดังนั้นการเลือกกรรมวิธีถ้าเป็นงานที่มีประสิทธิภาพจะส่งผลต่อคุณภาพของโปรแกรมในหลายแง่ เช่นแง่ของการ execute time หรือ storage efficiency เป็นต้น

**4.การกำหนดสายงานการรับข้อมูลและการดำเนินงาน**

(Outlining the Work Flow)

คือการนำเอาขั้นตอนที่เลือกในข้อที่ 3 มาผนวกกับโครงสร้างข้อมูลที่ออกแบบขึ้นมาเขียนเป็น statement chart เพื่อแสดงการทำงานของเครื่องคอมพิวเตอร์และเขียน statement chartเราอาจจะเลือกเครื่องมือ (Tools) ดังต่อไปนี้มาเพื่อแสดงขั้นตอนที่ 4 ออกมาให้ดู

A. **โปรแกรมลำลอง (Pseudo program หรือ Pseudo code)**

B. **ผังโปรแกรม (Program flow)**

C. **ผังงานของแนชชีชไนเดอร์ (N-S chart)**

เครื่องมือที่กล่าวมานี้เป็นที่นิยมในการแสดงภาพสายงานของผังงานในการเขียนโปรแกรมเพื่อสั่งให้เครื่องคอมพิวเตอร์ปฏิบัติงาน หรือ อาจจะมีเครื่องมืออย่างอื่น Wanier Orr diagram

**A. โปรแกรมลำลองหรือโปรแกรมเทียม (Pseudo program)**

เป็นการออกแบบโปรแกรมที่เขียนขึ้นมาโดยคำสั่ง (Keyword) ในภาษาอังกฤษมาสร้างเป็นขั้นตอนเพื่ออธิบายถึงกิจกรรมที่ทำตลอดจนกรรมวิธีการรับข้อมูลเข้ามาใช้งาน

นิยามของขั้นตอนการดำเนินงาน (Algorithms) คือ "An algorithm is a fomula, a recipe, a step-by-step procedure to be followed in order to obtain the solution to a problem"

**คุณลักษณะของขั้นตอนการดำเนินงานที่ดี** (Good algorithm) ประกอบด้วย

1. แก้ปัญหาได้ถูกจุดโดยใช้ช่วงเวลาที่เหมาะสม (Arrive at correct solution within a finish time)

2. ชัดเจน ถูกต้อง และไม่กำกวม (Be clear process and unambiguous)

3. สามารถกำหนดรูปแบบ ที่จะไปเขียนเป็นภาษาโปรแกรมได้อย่างไม่ยุ่งยาก (Be in a format which lends itself to and elegant implementation progrmming language)

***การใช้ประโยชน์จากโปรแกรมลำลองมีประโยชน์หลายประการคือ***

1. การเขียนประโยคคำสั่ง จะเป็นไปอย่างอิสระเหมือนกับเขียนภาษาอังกฤษที่อาจจะย่อสรุปหรือบางครั้งอาจจะเขียนขยายความตามความต้องการของการดำเนินงานได้

2. การเขียนโปรแกรมลำลองในส่วนของคำสั่งควบคุมก็ดำเนินการตามรูปแบบของโปรแกรมแบบโครงสร้างคือรูปแบบ sequence , iterative และ selection โดยการเขียนรูปตามแบบแล้วไม่ต้องเขียนสายการควบคุมเพียงแต่เขียนให้ตรงกับเงื่อนไขเท่านั้น

3. ภาษาโปรแกรมคอมพิวเตอร์โดยทั่วไปเช่นภาษาปาสคาล ภาษาซี มักจะมีโครงสร้างที่ตรงกับลักษณะของโปรแกรมลำลองอยู่แล้ว ดังนั้นการถอดรหัสจากโปรแกรมลำลองไปสู่ภาษาคอมพิวเตอร์จึงกระทำได้ไม่ยุ่งยากนัก

ตัวอย่างของการเขียนโปรแกรมลำลอง

```
do calculate pay
```

ซึ่งอาจขยายรายละเอียดเพิ่มเติมดังนี้คือ

```
input employee's work details
    do calculate pay
        print pay employee details
```

ตัวอย่างของการเขียนขั้นตอนการทำงาน (algorithms) แบบต่างๆซึ่งจะนำไปประกอบกันเขียนขึ้นเป็นโปรแกรมลำลองต่อไป

**ตัวอย่างที่ 1**

start

make a decision that divides the problem into three possible cases

<u>case 1</u> .....

.....

<u>case 2</u> .....

.....

<u>case 3</u> .....

.....

**ตัวอย่างที่ 2**

```
start :
if a_i equals the key then
     write "found at location,"i
     stop
else
     increment i by 1
     repeat
```

ตัวอย่างของการเขียนขั้นตอนดำเนินงาน (algorithm) ต่างๆ ที่ใช้กันอยู่

- compute n + 1
- determine if s is odd
- wait 1 hours and 17 minutes
- take the square root of two to three decimal place accuracy
- preceed to room 1234

ตัวอย่างของการเขียนขั้นตอนการดำเนินงาน **ที่ไม่ดี** เช่น

- compute n/0
- determine the largest prime number
- write out the exact value for the square root of 2
- wait -1 hours
- find out if there is or is not a goal

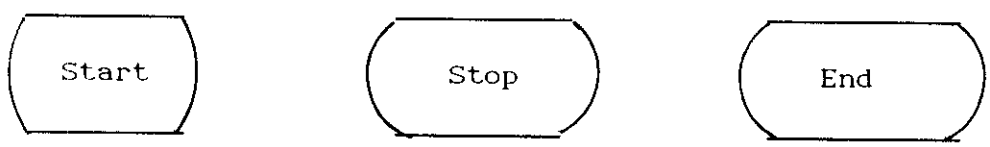
**B. ผังโปรแกรม (Program flow)**

จะเป็นรูปภาพซึ่งใช้สัญญลักษณ์ที่ตกลงกันในหมู่ผู้ใช้คอมพิวเตอร์ว่า สัญญลักษณ์แต่ละรูปจะหมายถึงสิ่งใดที่จะปฏิบัติการ เพื่อการทบทวนผู้ที่เคยใช้หรืออาจจะไม่เคยใช้สัญญลักษณ์ดังกล่าว ให้พอเข้าใจสัญญลักษณ์ที่ใช้กัน จะประกอบด้วย

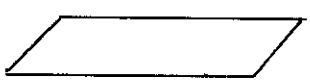
1. Termination block

จะเป็นสัญญลักษณ์เป็นจุดเริ่มต้นและจุดสิ้นสุดของงานโปรแกรม

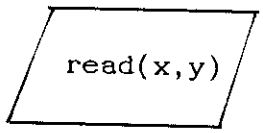
ภายในบล็อคดังกล่าวจะมีข้อความอธิบายอยู่ภายใน เช่น

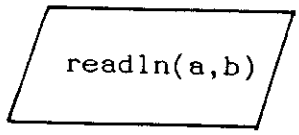


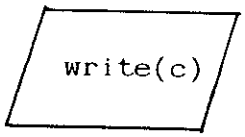

2. Input/Output

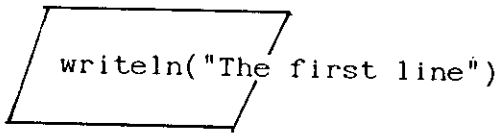


จะเป็นสัญญลักษณ์แสดงถึงการรับข้อมูลหรือแสดงข้อมูลที่ติดต่อกับอุปกรณ์ ภายนอกเช่น Keyboard, Diskette Drive, Tape Drive ,CRT , Printer และเพื่อเป็นการให้ความกระจ่างกับผู้ใช้งานภายใน บล็อคก็จะมีการ อธิบายถึงตัวแปรที่เก็บข้อมูลภายในเครื่องคอมพิวเตอร์ ตัวอย่างเช่น



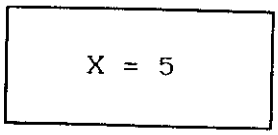



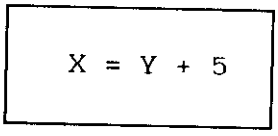




3. Process Block

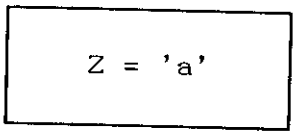
จะเป็นสัญลักษณ์แสดงถึงปฏิบัติการภายในเครื่องคอมพิวเตอร์ตัวอย่างของการใช้สัญลักษณ์ดังกล่าว เช่น




หมายถึงการกำหนด (assign) ค่า 5 ให้กับตัวแปร X ภายในสมองของคอมพิวเตอร์

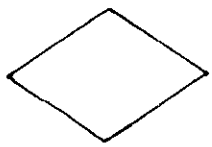



หมายถึงการส่งค่านิพจน์คณิตศาสตร์จากการนำค่าในตัวแปร Y มาบวกกับค่าคงที่ 5 แล้วส่งค่า(Transfer) ดังกล่าวเก็บไว้ที่ตัวแปร X

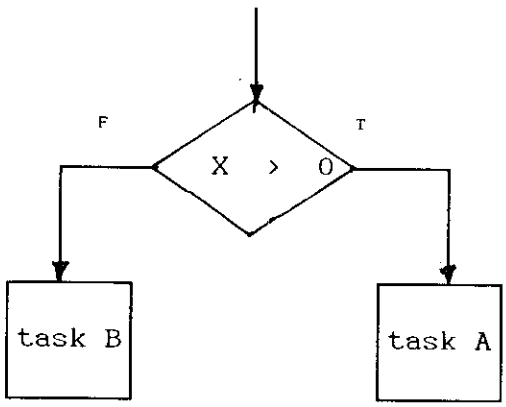



หมายถึงการกำหนดตัวอักขระ(Character) a ให้เก็บไว้ในตัวแปร Z

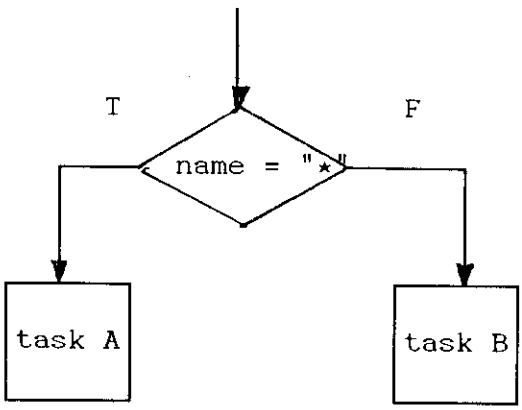
4. Decision Block



สัญญลักษณ์นี้หมายถึงการตัดสินใจเชิง ตรรก (logical expression)ซึ่งจะมีการเลือกสายงานที่จะดำเนินการตามเงื่อนไขของบล็อคดังกล่าว เช่น

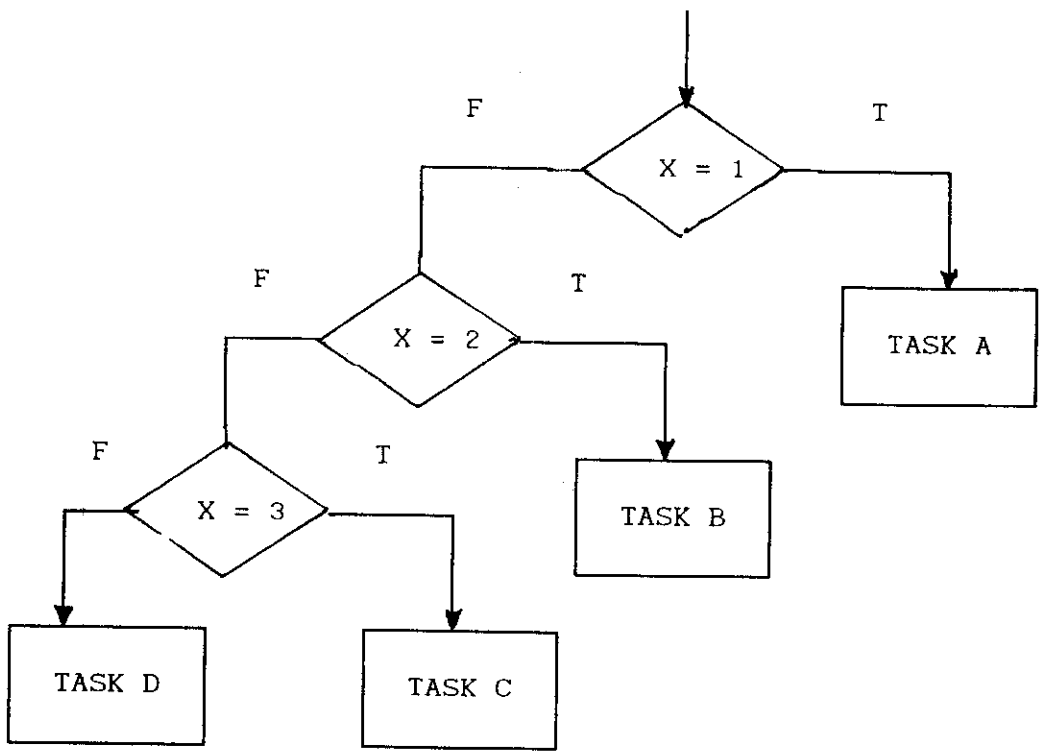



เป็นรูปการตัดสินใจว่าถ้า X มีค่ามากกว่า 0 จริงก็ให้ไปดำเนินงานตาม task A แต่ถ้า X ไม่มากกว่า 0 ก็ให้ไปดำเนินงานตาม task B

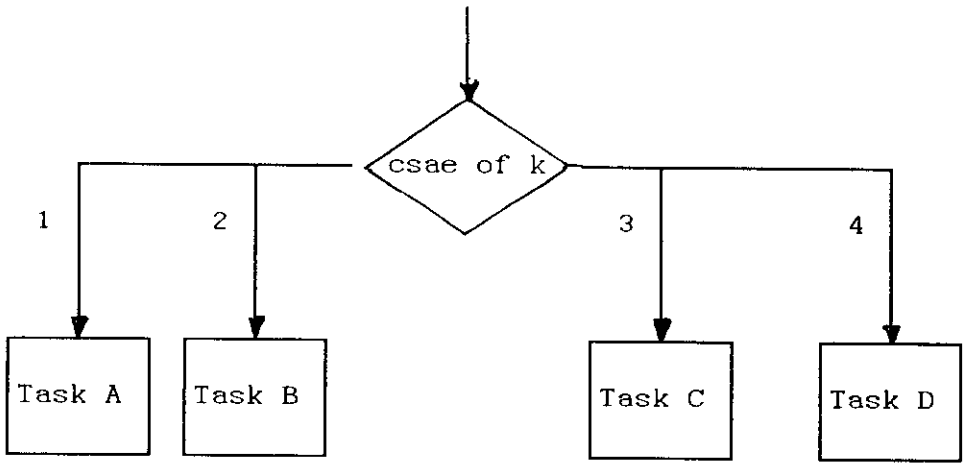



เป็นรูปการตัดสินใจว่าข้อมูลที่ปรากฏในตัวแปร nameนั้นคือ "*" หรือไม่ ถ้าเป็น "*" จริง นิพจน์ตรรกดังกล่าวจะให้ค่าเป็นจริงและไปดำเนินงานที่ task A ต่อไปแต่ถ้าข้อมูลที่เก็บใน nameไม่ใช่"*" นิพจน์ตรรกดังกล่าวจะให้ผลเป็นเท็จ และคอมพิวเตอร์จะไปทำงานที่ task B ต่อไป

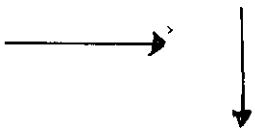
บางครั้งเราอาจจะใช้สัญลักษณ์ดังกล่าวในลักษณะพิเศษที่แตกต่างออกไปคือ ใช้แยกแยะกรณีของการดำเนินงานซึ่งมีมากกว่า 2 กรณีย์ ซึ่งในลักษณะดังกล่าวนี้การแยกแยะภายในเงื่อนไขของบล็อคจะมีลักษณะการเก็บ case ที่ใช้ในบางภาษา แต่ในบางภาษาก็ไม่สามารถดำเนินงานในลักษณะเช่นนี้ได้ จึงต้องแยกแยะโดยการใช้สัญลักษณ์แบบแรกมาต่อเนื่องมากลั่นกรองต่อไป ดังตัวอย่างเช่น



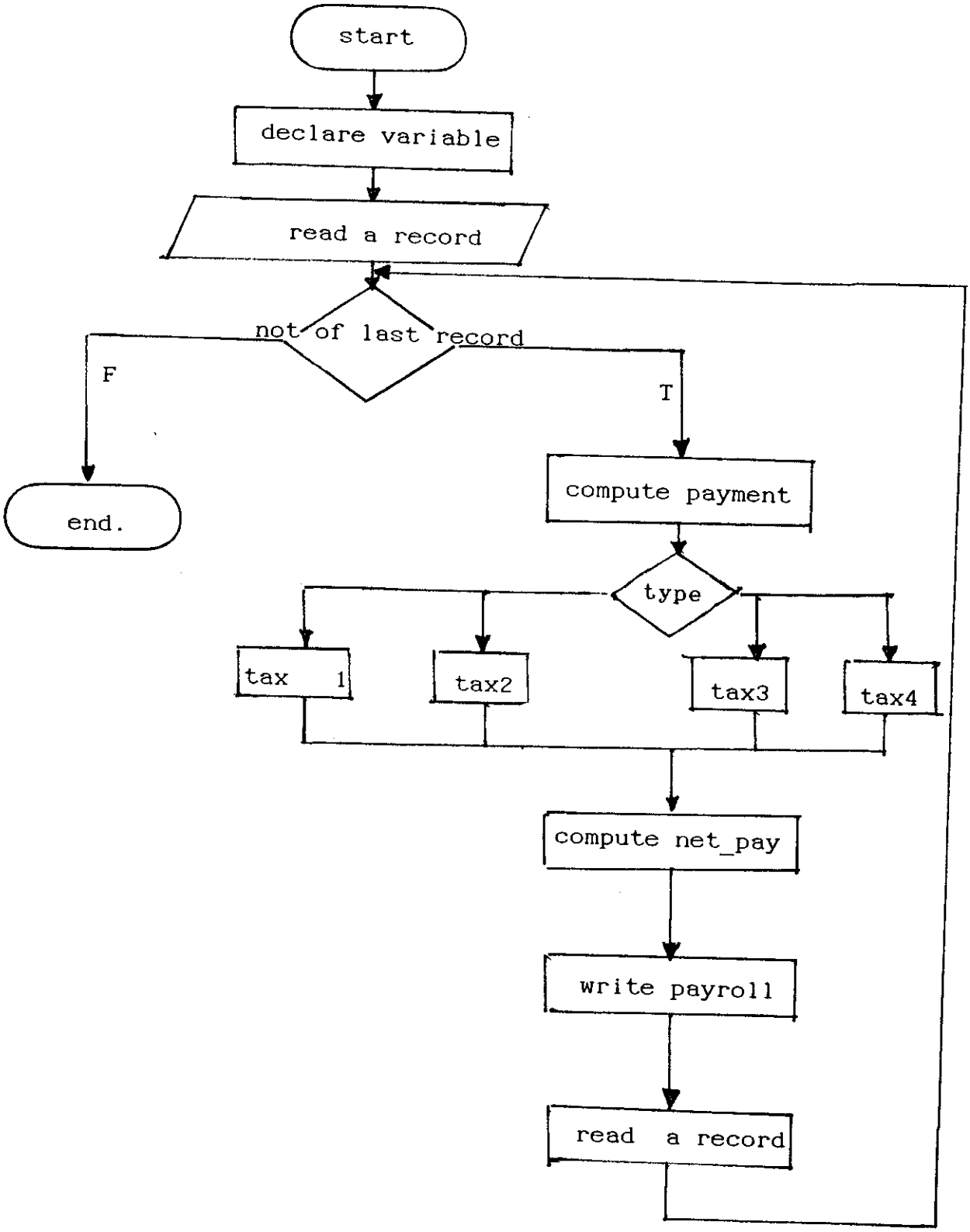

รูปดังกล่าวนี้จะใช้สัญลักษณ์ของบล็อคดังกล่าวในเงื่อนไขของนิพจน์ตรรก ซึ่งในบางภาษาอาจจะแปลงไปใช้รูปต่อไปนี้แทนได้




การใช้รูปแบบที่แสดงในกรณีวิธีนี้นั้นจะตรงกับคำสั่ง "CASE" ในภาษาปาสคาลและในภาษาซี

5.  เครื่องหมายลูกศร แสดงแทนเส้นทางของกิจกรรมที่จะดำเนินการต่อไป

ให้พิจารณาวิธีการเขียนผังโปรแกรมจากตัวอย่างต่อไปนี้

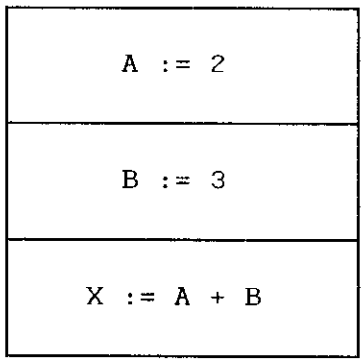

start
declare variable
read a record
not of last record
F
T
end.
compute payment
type
tax 1
tax2
tax3
tax4
compute net_pay
write payroll
read a record

**C. N-S Chart (Nassi-Schneiderman Chart : N-S Chart)**

สัญญลักษณ์ที่ใช้ใน N-S Chartจะแสดงด้วยรูปสี่เหลี่ยมผืนผ้าแต่จะมีเส้นแบ่งลักษณะของกิจกรรมที่ทำแตกต่างกัน

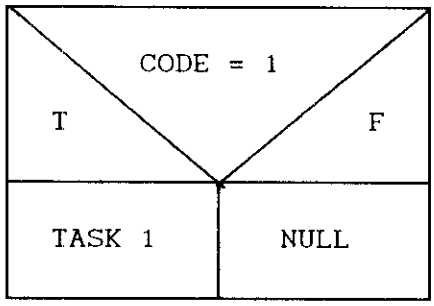
ตัวอย่างของ N-S Chart

1. Sequence structure

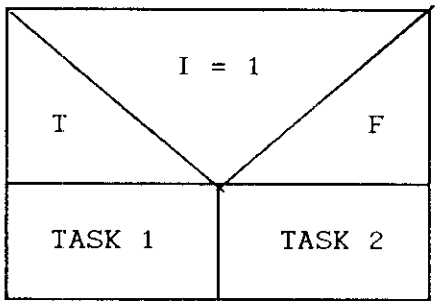



2. Decision structure

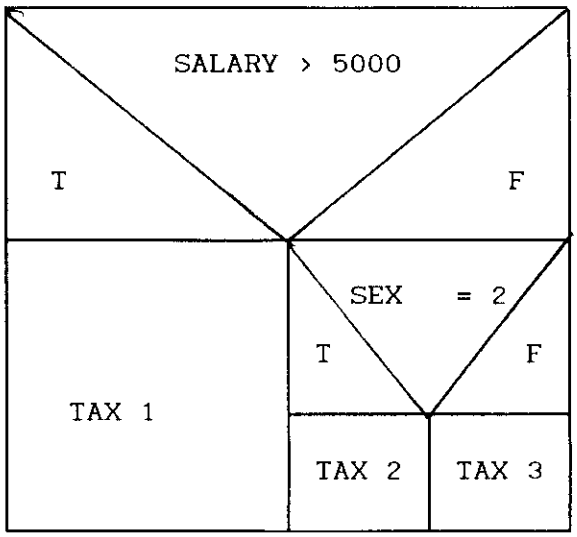
2.1 If ~ Then




2.2 If ~ Then ~ Else

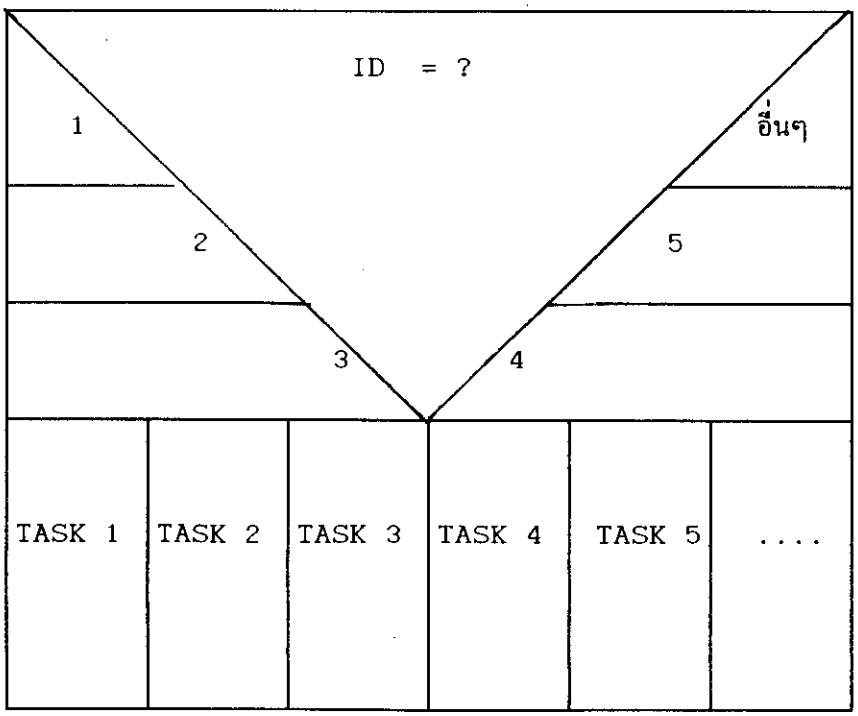


2.3 Nested If ~ Then ~ Else

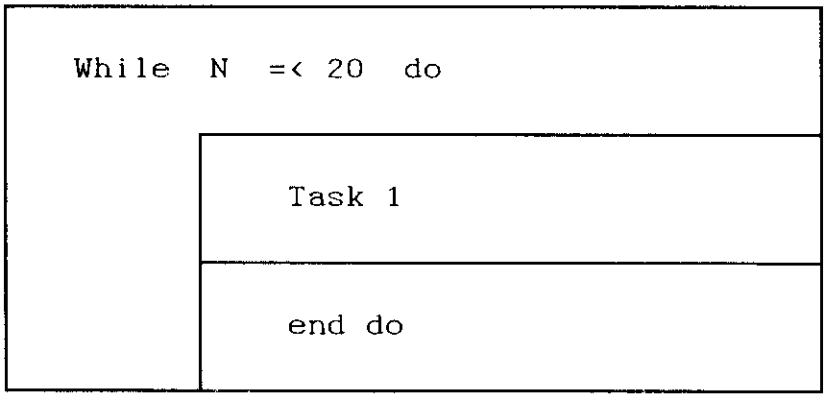



2.4 CASE

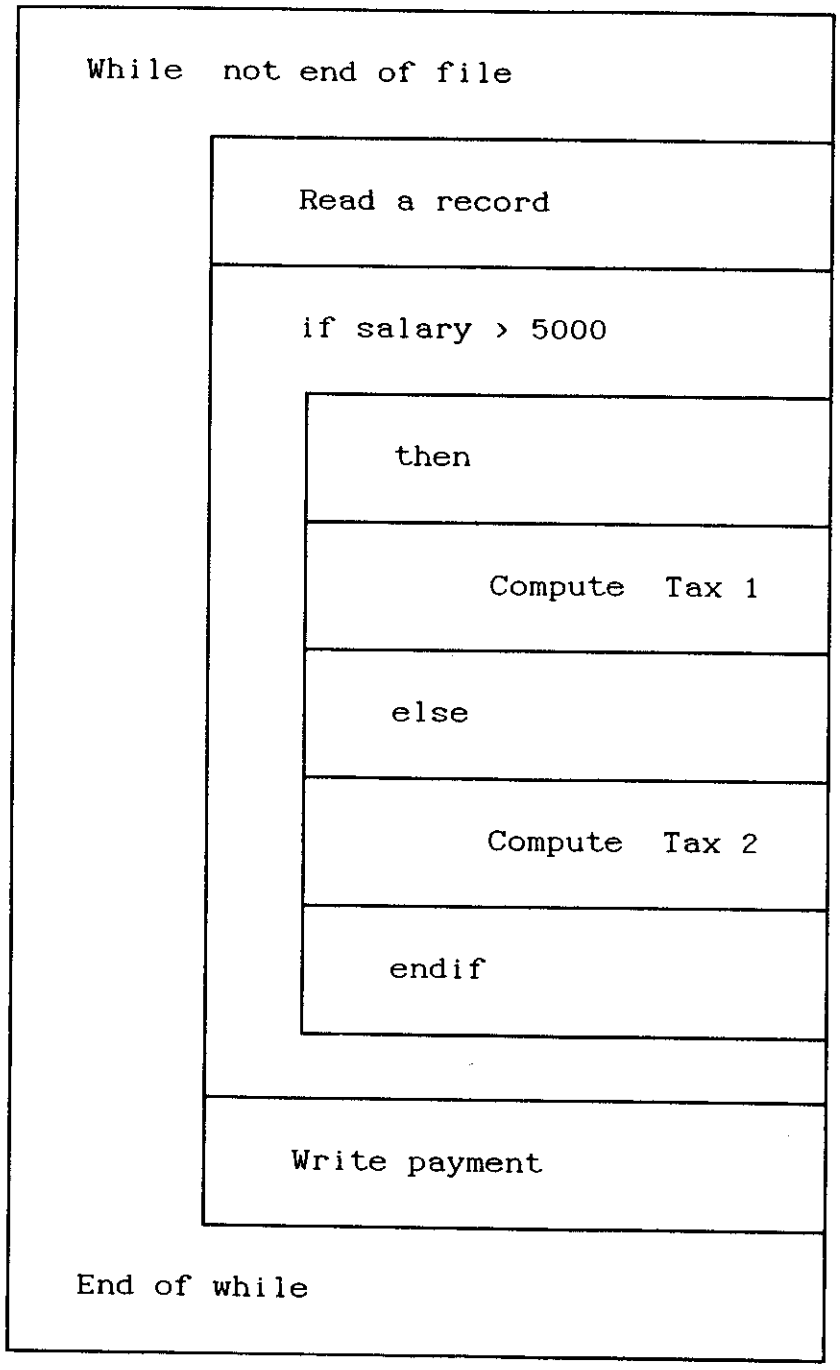
คำสั่ง CASE นี้บางภาษาอาจจะใช้ไม่ได้ดังนั้นภาษาที่ไม่สามารถใช้คำสั่ง CASE ได้จะใช้ในรูปแบบของ Nested If ~ Then ~ Else แทน



3. WHILE ~ DO



ตัวอย่างการใช้ N-S Chart เขียนสายการทำงาน



**5. การแปลงรหัส**

( Coding )

จาก Structure ในข้อ 5 ที่เลือกสร้างได้เสร็จแล้ว เราจะนำมาตัดสินใจว่าควรจะเลือกภาษาคอมพิวเตอร์ที่เหมาะสมและตรงกับลักษณะของงาน การเลือกภาษาคอมพิวเตอร์นั้นเราตัดสินใจตามปัจจัย 3 ประการ คือ

1. The nature of the problem.
2. The programming language available on your computer.
3. The dictates and limitations of your particular computer installations.

**6. การตรวจสอบค้นหาที่ผิดพลาดภายในโปรแกรม**

(Debugging)

**7. การทดสอบความถูกต้องและการปฏิบัติงานของโปรแกรม**

(Testing and Validation)

**8. การเขียนเอกสารกำกับโปรแกรม**

(Documentating)

การเขียนเอกสารกำกับโปรแกรมนั้นจะเป็นกระบวนการต่อเนื่องที่เราจะต้องดำเนินการตั้งแต่ ขั้นตอนที่ 1 ไปจนถึงขั้นตอนที่ 4 ในขั้นตอนที่ 3 จะอธิบายถึงการเลือก algorithm ที่ใช้เอกสารนี้จะใช้ในการเป็นคู่มือเพื่อการ ปรับปรุง (Modified) โปรแกรมต่อไปในอนาคต ตลอดจนเป็นคู่มือของผู้ใช้โปรแกรมดังกล่าวแล้วเกิดปัญหาขึ้นมา

**9. การบำรุงดูแลและรักษาโปรแกรม**

(Program Maintenance)

ภายหลังเมื่อโปรแกรมดังกล่าวถูกนำไปใช้ปฏิบัติงานจริงในช่วงระยะเวลาหนึ่งเราต้องคอยดูตรวจสอบว่าโปรแกรมดังกล่าวยังอยู่ในสภาพสมบูรณ์ว่าไม่มีส่วนใดส่วนหนึ่งที่ขาดหายไป และกรณีที่สภาพแวดล้อมเปลี่ยนไปมีผลกระทบต่อโปรแกรมหรือไม่ ถ้ามีเราต้องคอยดูแลและปรับปรุงโปรแกรมให้ถูกต้องและสอดคล้องกันด้วย