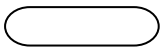
Algorithm for Programming

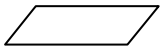
งานพัฒนาโปรแกรม คือการวางแผนแก้ปัญหา เครื่องมือที่ใช้จะมีให้เลือกอยู่สองชนิดคือ *Flowchart และ Pseudocode* การเขียน flowchart จะมีข้อได้เปรียบ Pseudocode คือ flowchart เขียนเป็นรูปภาพทำให้ง่ายต่อการเข้าใจมากกว่า Pseudocode ซึ่งเขียนเป็นตัวหนังสือ การเขียน flowchart หรือ Pseudocode ก็คือการลำดับขั้นตอนการทำงานนั่นเอง

1. Flowchart

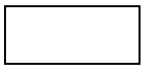
การเขียน Flowchart เบื้องต้นเราจะใช้สัญลักษณ์ดังต่อไปนี้คือ (จริงๆ แล้วมีมากกว่านี้ แต่ที่ใช้บ่อยๆ มีเท่านี้)



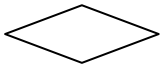
จุดเริ่มต้น/สิ้นสุด (Terminator)



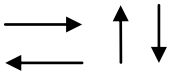
ข้อมูล นำเข้า/ส่งออก (Input / Output)



ประมวลผล (Process)



ตัดสินใจ (Decision)



ทิศทาง (Direction)

จุดต่อเชื่อม (Connector)

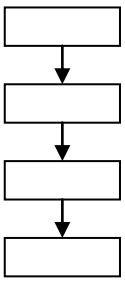
การเขียน Flowchart แบบโครงสร้าง

การเขียน Flowchart แบบโครงสร้างมีประโยชน์คือทำให้การไล่ขั้นตอนการทำงานทำได้ง่ายและเป็นระเบียบ ซึ่งมีหลักการเขียนอยู่ สามข้อ คือ

- Sequence
- Selection
- Iteration

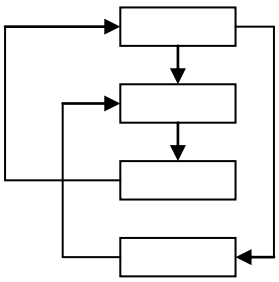
SEQUENCE

คือการเขียนให้เป็นลำดับ ดังรูปที่ 1.



รูปที่ 1

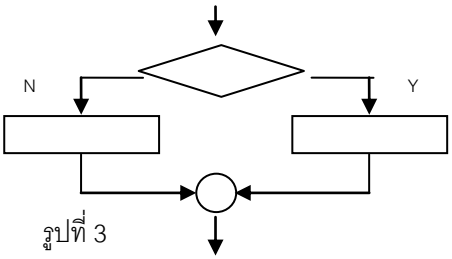
ไม่ใช่เขียนข้ามไปข้ามมาดังรูปที่ 2.



รูปที่ 2

SELECTION

เป็นทางเลือกของโปรแกรมซึ่ง **จะต้องมีเพียงสองทางเลือกเท่านั้น** และ **หลังจากนั้นทางเลือกทั้งสองต้องมาพบกัน** และทำงานในขั้นตอนต่อไป

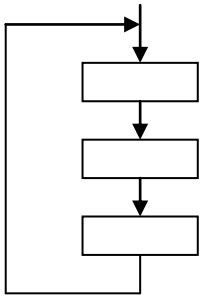


รูปที่ 3

```
if    <condition>
        statements_1
 else
        statements_2
end
```

ITERATION

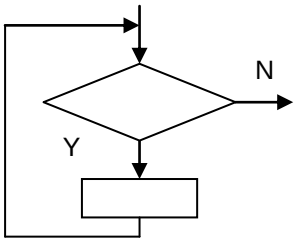
คือการทำซ้ำ เป็นการเขียน flowchart ให้กลับมาทำงานในขั้นตอนอย่างเก่า จะเห็นว่า flowchart มีลักษณะวน ซึ่งเรียกว่า loop และจะสังเกตว่า การวน loop ดัง รูปที่ 4 จะไม่มีทางออกไปทำงานในขั้นตอนต่อไปได้เลย เพื่อที่จะทำให้ออกจาก loop ได้จะต้องมีการ เช็คเพื่อออกจาก loop ดังจะได้กล่าวต่อไป



รูปที่ 4.

ในการเขียน flowchart จะมี loop ให้เลือกใช้ได้สองประเภทคือ **WHILE LOOP** และ**FOR LOOP**

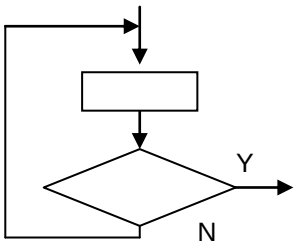
- **WHILE LOOP จะทำการเช็คเพื่อที่จะออกจาก loop ก่อนที่จะทำงานตามคำสั่งใน loop และ เงื่อนไขเพื่อที่จะออกจาก loop จะต้อง<u>เป็นเท็จ</u> ดังรูปที่ 5.**



รูปที่ 5.

```
while    <expression>
          statements
end
```

- **FOR LOOP จะทำการวน loop ตามจำนวนรอบที่เราต้องการ (รู้จำนวนรอบ) ดังรูปที่ 6.**



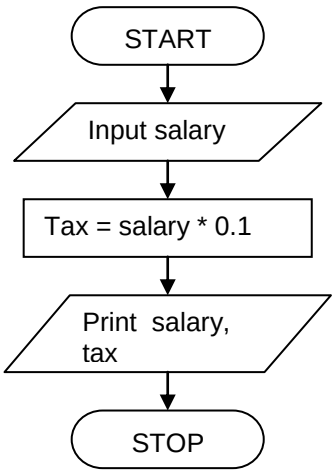
รูปที่ 6.

```
for (variable=expression)
     statements
end
```

ตัวอย่างการเขียน Flowchart

<u>ตัวอย่าง 1</u> การเขียน Flowchart เพื่อคำนวณภาษีที่พนักงานต้องชำระ อัตราภาษี 10%

Flowchart ดังกล่าวกำหนดให้ผู้ใช้ป้อนค่าเงินเดือน (salary) แล้วเครื่องจะทำการคำนวณ ภาษี (tax) 10% ให้โดยอัตโนมัติ และจะพิมพ์ค่า salary กับ tax



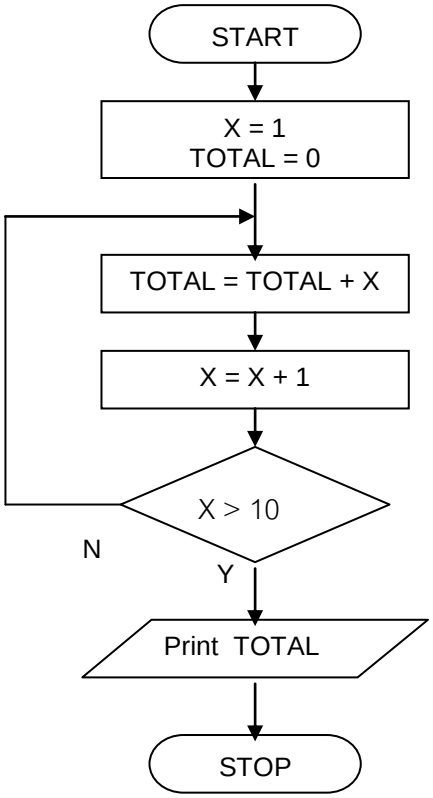
รูปที่ 7

Output ที่เราต้องการก็คือ salary และ tax (การสั่งพิมพ์ขึ้นอยู่ที่เราว่าเราต้องการให้พิมพ์อะไร ไม่จำเป็นต้องพิมพ์ salary, tax ตามตัวอย่างก็ได้ เราอาจสั่งพิมพ์ tax อย่างเดียวก็ได้

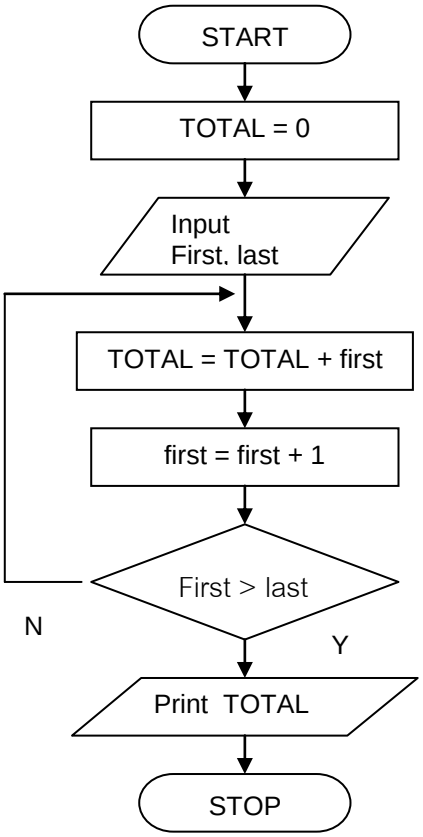
Input **คือสิ่งที่ผู้ใช้ต้องป้อนให้ระบบ** จากตัวอย่างก็คือ salary เพราะหากผู้ใช้ไม่ป้อน salary ระบบจะคำนวณ tax ไม่ได้ ส่วนอัตราภาษี 10% ผู้ใช้ไม่ต้องป้อนเพราะมีการกำหนดมาอยู่แล้วว่าภาษีคือ 10% ระบบไม่จำเป็นต้องถามผู้ใช้ เพราะฉะนั้นอัตราภาษี 10% จึงไม่ใช่ input หากจะสรุปง่ายๆ input ก็คือสิ่งที่เราต้องถามผู้ใช้ ส่วนอัตราภาษีคือ ค่าคงที่ ซึ่งเราจะกำหนดไว้ในโปรแกรมเลย โดยผู้ใช้ไม่ต้องป้อน

หากเราต้องการเขียน Flowchart ให้บวกเลข 1 ถึง 10 จะพบว่า Flowchart ดังกล่าวไม่มี input เลยเพราะ flow ดังกล่าวไม่จำเป็นต้องถามผู้ใช้ ดังรูปที่ 8.

แต่หากเราต้องการเขียน Flowchart ให้**บวก*เลขจำนวนที่หนึ่ง* ถึง *เลขจำนวนที่สอง*** เราจะพบว่าผู้ใช้จำเป็นต้องบอกเราว่า จำนวนที่หนึ่ง คือเลขอะไร และ จำนวนที่สอง คือเลขอะไร เพราะฉะนั้น input คือ first (เลขจำนวนที่หนึ่ง) และ last (เลขจำนวนที่สอง) ดังรูปที่ 9



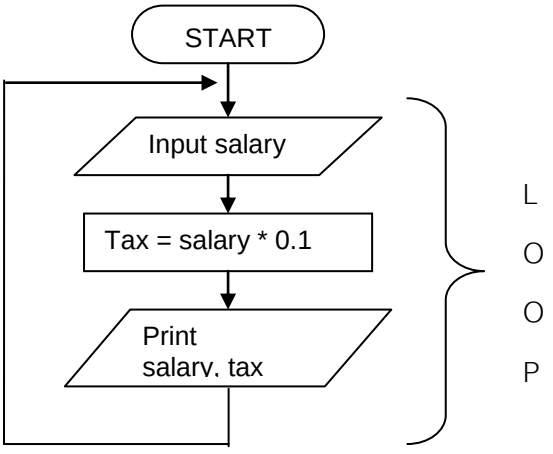
รูปที่ 8



รูปที่ 9

Iteration
(การทำซ้ำ)

Flowchart ดังแสดงในรูปที่ 7จะคำนวณภาษีสำหรับพนักงานหนึ่งคน หากเราต้องการให้คำนวณคนที่สอง สาม สี่ ... เราจะต้องสั่งให้กลับมาทำงานดังแสงในรูปที่ 10

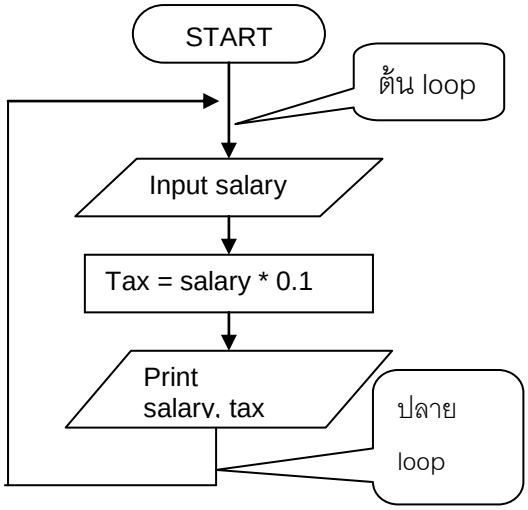


รูปที่ 10

ให้สังเกตว่า flowchart ดังกล่าวไม่มีทางออกจาก loop ได้เลย นั่นหมายถึงหลังจากคำนวณภาษีเสร็จเครื่องจะรอรับค่า salary คนต่อไปไม่มีวันสิ้นสุด

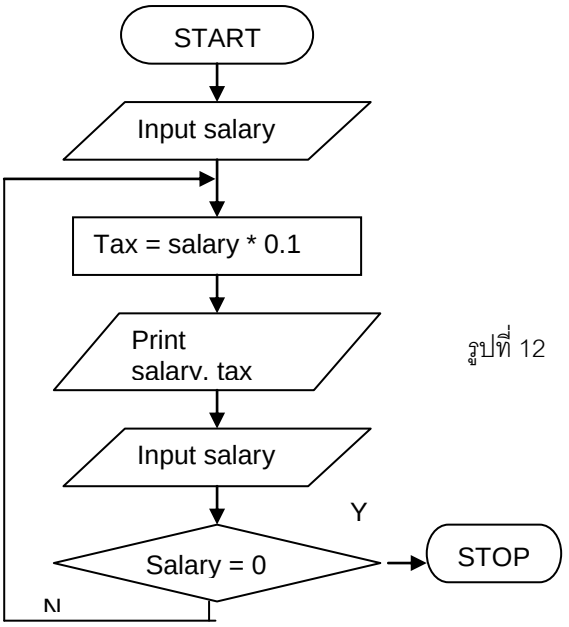
คำถามที่เกิดขึ้นก็คือ เมื่อไหร่ที่เราต้องการออกจาก loop คำตอบก็คือ เมื่อคำนวณภาษีให้พนักงานทุกคนครบแล้ว วิธีการที่เราจะบอกระบบว่าพนักงานหมดแล้วเราสามารถบอกได้โดย “ถ้าเรา input ค่า salary เป็น 0 หมายถึงพนักงานหมดแล้ว นั่นคือให้ออกจาก loop” (ที่ใช้เป็น 0 เพราะไม่มีพนักงานคนใดที่มีเงินเดือนเท่ากับ 0 บาท) ซึ่งเราเรียกค่าดังกล่าวว่าค่า dummy

ดังได้กล่าวไว้ต้นแล้วว่าการเลือกใช้ loop มีให้เลือกใช้สองชนิดคือ DO WHILE และ DO UNTIL ซึ่ง DO WHILE จะทำการเช็คเพื่อออกจาก loop ที่ต้น loop ในขณะที่ DO UNTIL เช็คปลาย loop



รูปที่ 11

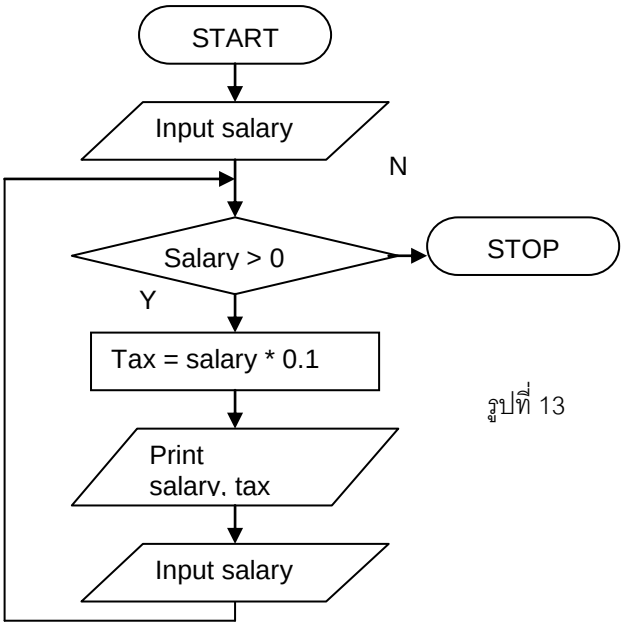
ให้สังเกตว่า การเช็ค ณ ต้น loop คือ **การเช็คก่อนมี process ใดๆทั้งสิ้น (DO WHILE)** ในขณะที่การเช็ค ณ ปลาย loop คือ**ให้มี process ทุกอย่างก่อนแล้วค่อยเช็ค (DO UNTIL)** โดยปกติแล้วเราจะใช้ DO WHILE หรือ DO UNTIL ก็ได้ (แต่มีบางกรณีที่จำเป็นต้องใช้ DO WHILE หรือ DO UNTIL) จาก flowchart รูปที่ 11 หากเราใช้ DO UNTIL จะได้ flowchart ดังรูปที่ 12



รูปที่ 12

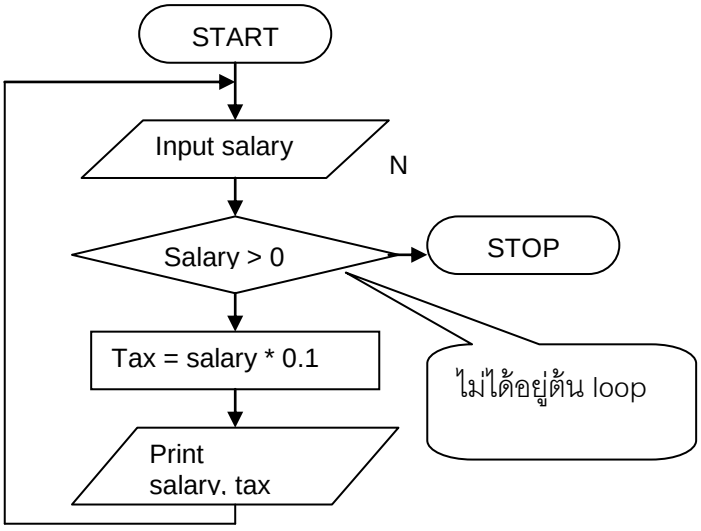
จะเห็นว่าเงื่อนไขออกจาก loop จะต้องเป็นจริง และการเช็คออกจาก loop จะอยู่ ณ ตำแหน่งสุดท้ายของ loop นอกจากนี้การที่ต้องมี input salary เพิ่มขึ้นมาอีกหนึ่ง process และไว้อยู่หน้าการเช็คเพื่อออกจาก loop เพราะว่า เมื่อผู้ใช้ใส่ค่า 0 มา ระบบจะทำการออกจาก loop ทันที เพราะหากไว้ตำแหน่งอื่นระบบอาจจะมีการ print หรือคำนวณ tax ซึ่งเราไม่ต้องการให้ทำ

Flowchart รูปที่ 13 แสดงการใช้ **WHILE loop** ให้สังเกตว่า การเช็คเพื่อที่จะออกจาก loop อยู่ต้น loop และเงื่อนไขเพื่อที่จะออกจาก loop จะเป็นเท็จ (เพราะฉะนั้นเงื่อนไขจึงต้องเป็น salary>0)



รูปที่ 13

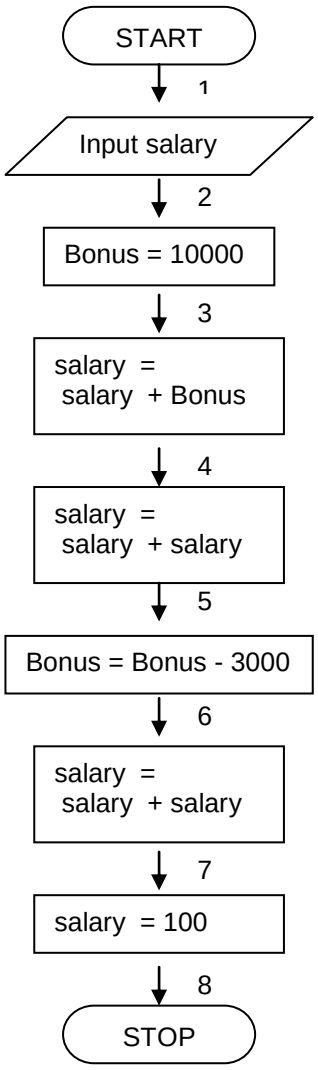
ใน Flowchart จะมีการ input Salary อยู่สองตำแหน่งคือบนสุด และใน loop ณ ตำแหน่งล่างสุด input Salary ซึ่งอยู่บนสุดมีไว้เพื่อ input ค่า salary คนแรก เท่านั้น สำหรับค่า salary คนต่อๆมา จะถูก input จาก input salary ที่อยู่ใน loop สาเหตุที่เราไม่สามารถเขียน flowchart ให้วนกลับไป input salary คนต่อๆมาดังรูปที่ 14 แม้ว่าจะสามารถทำงานได้ถูกต้อง เนื่องจากจะผิดกฎ WHILE ซึ่งกำหนดไว้ว่า การเช็คเพื่อออกจาก loop จะต้องอยู่ต้น loop



รูปที่ 14

ค่าตัวแปรใน Flowchart

ในการเขียน Flowchart เป็นสิ่งสำคัญอย่างยิ่งที่เราจะต้องรู้ว่าคอมพิวเตอร์มีวิธีการอย่างไรในการเก็บค่าของตัวแปร เพราะไม่เช่นนั้นแล้วเราจะไม่สามารถเขียน flowchart ที่ถูกต้องได้เลย ในการเก็บค่าตัวแปรของคอมพิวเตอร์นั้น ตัวแปรแต่ละตัวจะมีค่าเพียงค่าเดียว ดังตัวอย่างข้างล่าง



รูปที่ 15

สมมติ input salary 5000

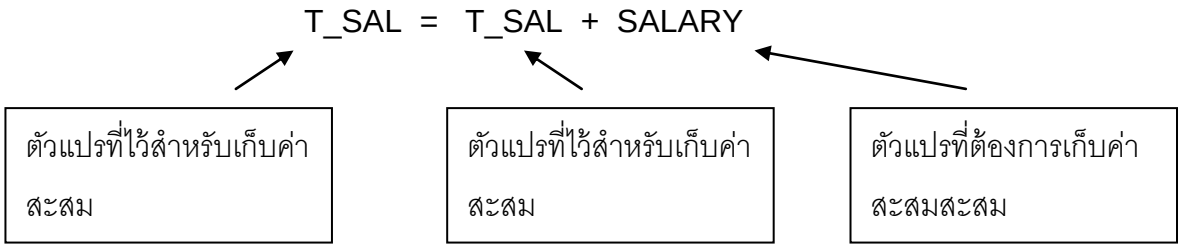
ตำแหน่ง	SALARY	BONUS
1	0	0
2	**5,000**	0
3	5,000	10,000
4	15,000 (5,000+10,000)	10,000
5	30,000 (15,000+15.000)	10,000
6	30,000	7,000 (10,000-3,000)
7	60,000 (30,000+30,000)	7,000
8	100	7,000

จะเห็นว่า ณ เวลาหนึ่งๆ ตัวแปรจะถูกแทนที่ด้วยค่าใหม่ โดยไม่มีการเก็บค่าเก่า เช่น salary จาก 0 $\Rightarrow$ 5,000 $\Rightarrow$ 15,000 $\Rightarrow$ 30,000 $\Rightarrow$ 60,000$\Rightarrow$ 100

ACCUMULATION (การสะสมค่า)

ในการเขียน flowchart ทางธุรกิจส่วนใหญ่จะต้องการให้หายอดรวม เช่นหายอดรวมเงินเดือนของพนักงานทุกคน หายอดรวมของเงินภาษี ฯลฯ ในการเขียน flowchart เราจะใช้วิธีสะสมค่า โดยกำหนด ตัวแปรที่ไว้สำหรับเก็บค่าสะสม = ตัวแปรที่ไว้สำหรับเก็บค่าสะสม + ตัวแปรที่ต้องการสะสม

สมมติเราต้องการหายอดรวมของ salary เราสามารถเขียนการทำงานดังกล่าวได้ดังนี้

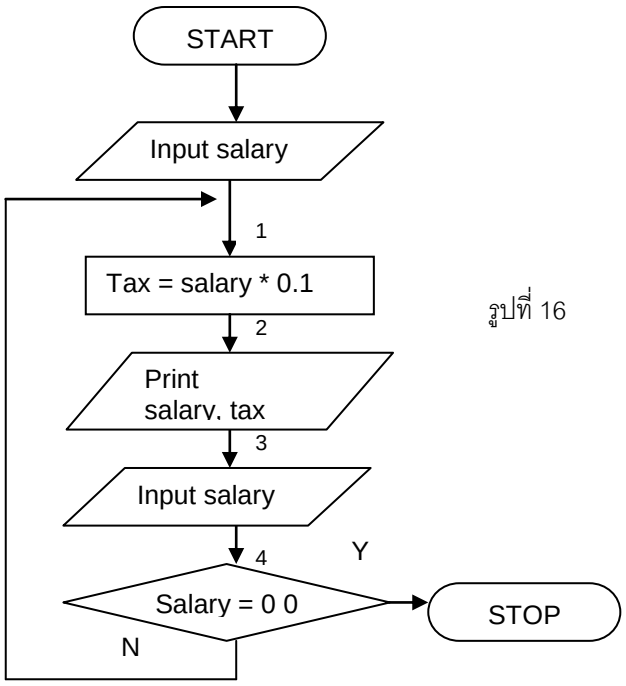


จากตัวอย่าง หากเราต้องการหายอดรวมของ salary และ tax จะได้ process ดังนี้

T_SAL = T_SAL + SALARY

T_TAX = T_TAX + TAX

การวางตำแหน่งของ process ดังกล่าวจะต้องวางในตำแหน่งที่ salary, และ tax ของทุกคนวิ่งผ่าน (เนื่องจากเราจะหาค่าสะสมของทุกคน) นั่นก็คือจะต้องวาง process ทั้งสองใน loop แต่นั่นมิได้หมายความว่าทุกตำแหน่งภายใน loop จะสามารถวางได้ เราจะมาพิจารณาแต่ละจุดดังนี้

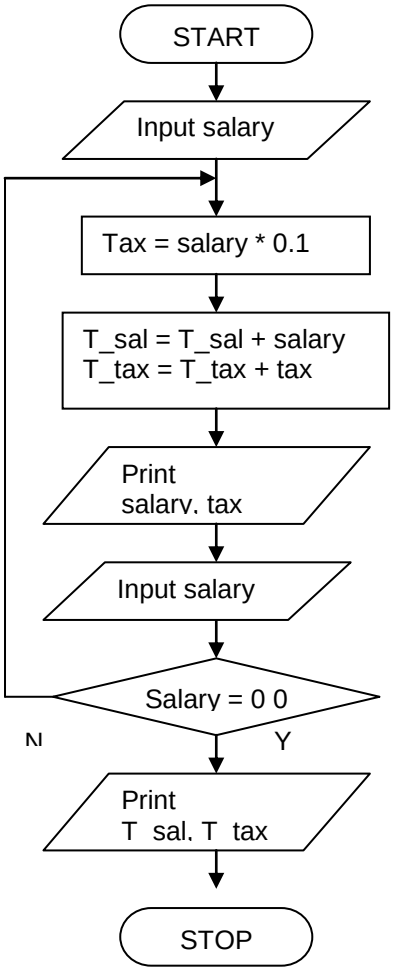


รูปที่ 16

จุดที่	T_SAL = T_SAL + SALARY	T_TAX = T_TAX + TAX
1.	ได้	ไว้ไม่ได้ (ค่าของคนสุดท้ายจะไม่ถูกสะสม)
2.	ได้	ได้
3.	ได้	ได้
4.	ไม่ได้ (input ของคนที่หนึ่งจะไม่ถูกร่วมสะสม เนื่องจากถูกแทนที่ด้วยค่าของคนที่สอง)	ได้

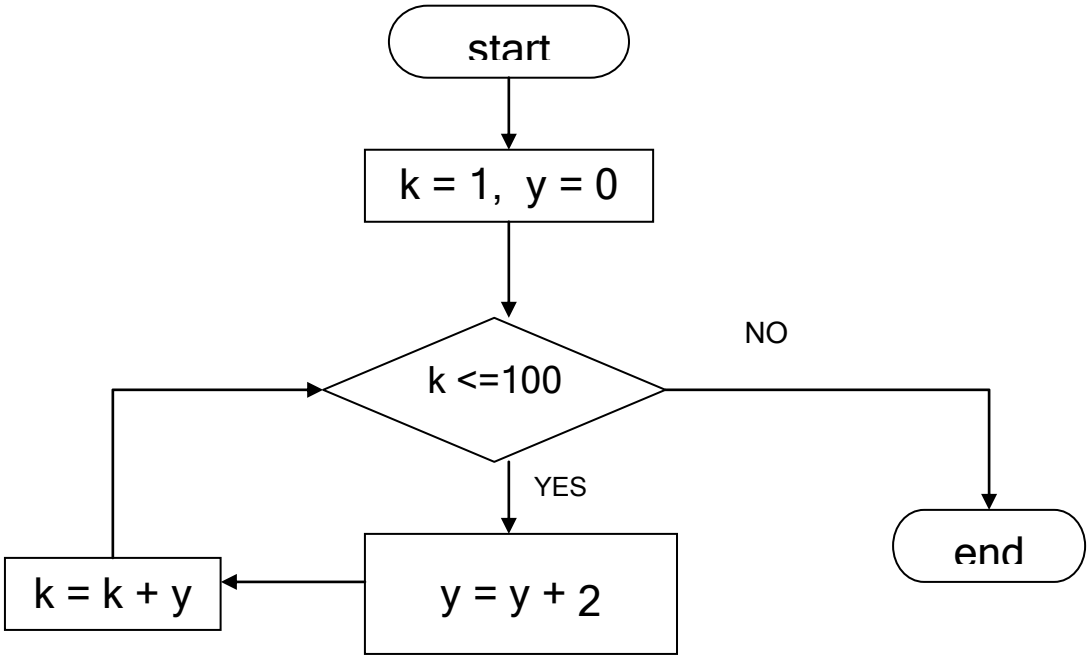
เพราะฉนั้นเราสามารถวาง T_SAL = T_SAL + SALARY ไว้ ณ ตำแหน่ง 1, 2, หรือ 3 ก็ได้ สำหรับ T_TAX = T_TAX + TAX ไว้ ณ ตำแหน่ง 2, 3 หรือ 4 ก็ได้ สำหรับตัวอย่างเลือกวางทั้งสองไว้ ณ ตำแหน่งที่ 2 (ทั้งสองไม่จำเป็นต้องวางตำแหน่งเดียวกัน)

นอกจากนี้เราจะสั่งให้พิมพ์ค่ารวมเงินเดือน (T_SAL) และ รวมภาษี (T_TAX) หลังจากที่คำนวณของทุกคนเรียบร้อยแล้ว นั่นคือต้องสั่งพิมพ์นอก loop นั่นเองดังแสดงในรูปที่ 17



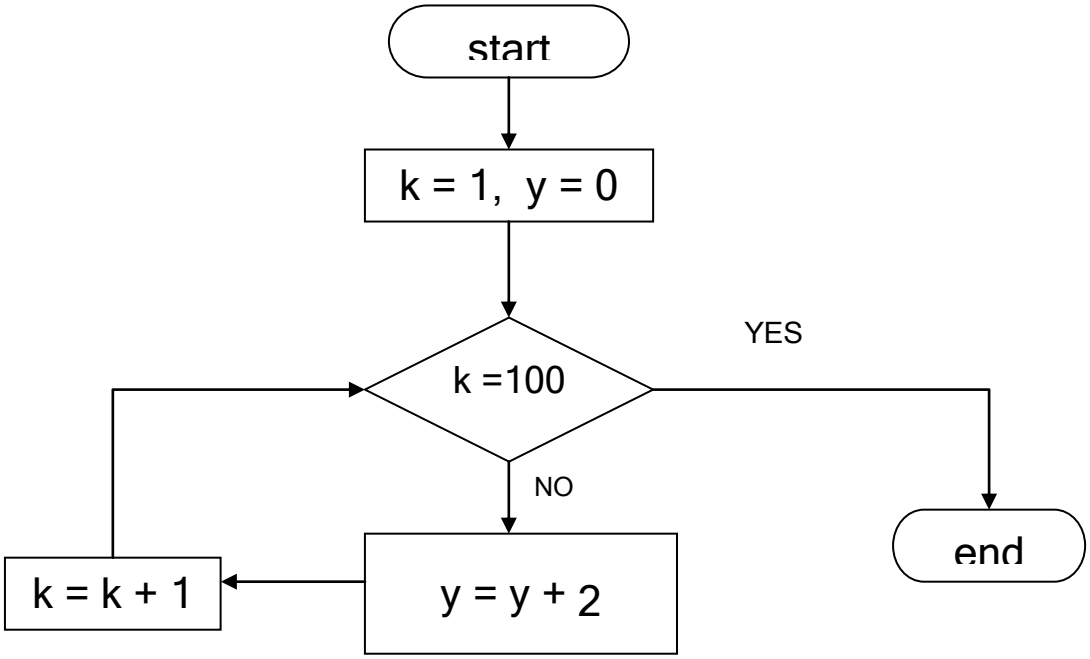
รูปที่ 17

ตัวอย่างที่ 1



จงหาค่าของ y

ตัวอย่างที่ 2



จงหาค่าของ y

2. รหัสเทียม (Pseudocode)

รหัสเทียม (Pseudocode) คือ การเขียนโปรแกรมในรูปแบบภาษาอังกฤษ(หรือภาษาไทย) ที่มีขั้นตอนและรูปแบบแน่นอนกะทัดรัด และมองดูคล้ายภาษาระดับสูงที่ใช้กับเครื่องคอมพิวเตอร์ซึ่งไม่เจาะจงภาษาใดภาษาหนึ่ง การเขียนรหัสเทียมไม่มีกฎที่แน่นอนตายตัว แต่ก็มีลักษณะคล้ายกับภาษาคอมพิวเตอร์

ตัวอย่าง การต้มมาม่า

1. หามาม่าไว้ 1 ซอง

2. ฉีกซองมาม่าและเทลงถ้วยเปล่า

3. ฉีกซองเครื่องปรุง แล้วเทลงถ้วยเดิม

4. ต้มน้ำให้ร้อนได้ที่ แล้วเทลงถ้วย

5. ปิดฝาไว้ 3 นาที

6. เปิดฝา แล้วรับประทาน

เกณฑ์ในการเขียนรหัสเทียม มีดังนี้

1. ประโยคคำสั่ง เขียนเป็นภาษาอังกฤษอย่างง่าย

2. ประโยคคำสั่งหนึ่ง ๆ จะเขียนต่อหนึ่งบรรทัดเท่านั้น

3. คำหลัก (key word) และการเขียนย่อหน้าใช้เพื่อ
   แยกโครงสร้างควบคุม

4. คำสั่งถูกเขียนจากบนลงล่างโดยมีทางเข้า-ออก เพียงทางเดียว

5. กลุ่มของประโยคคำสั่งอาจถูกจัดอยู่ในรูปส่วนจำเพาะ(Module) และแต่ละกลุ่มต้องมีชื่อเรียก

การเขียนรหัสเทียม (Pseudo code)

การปฏิบัติการของคอมพิวเตอร์แบ่งออกเป็น 6 แบบดังนี้

1. การกำหนดค่าให้กับตัวเก็บข้อมูล

    1.1 กำหนดค่าเริ่มต้น คำที่ใช้ *Initialize* หรือ *Set*

        เช่น Set AA = 500

    1.2 กำหนดค่าที่เกิดจากการประมวลผลไว้ที่ตัวเก็บ จะใช้เครื่องหมาย =

        เช่น AA = 500 + 1 หรือ BB = 100 หรือ CC = AA

2. การรับข้อมูล คำที่ใช้ *Read* หรือ *Get* เช่น Read AA

3. การแสดงข้อมูลออก คำที่ใช้ *Print,Write,Display,Output*

    เช่น Print “Hello Students” หรือ Print AA

4. การปฏิบัติการทางคณิตศาสตร์ + , - , * , ( )

    เช่น C = (F-32) * 5/9

5. การเปรียบเทียบและทำการเลือก คำที่ใช้ *IF, THEN, ELSE* เช่น

```
IF buffalo = “My friend” THEN
    print “Yes My friend is buffalo”
END IF
```

```
IF buffalo = “My friend” THEN
    print “Yes My friend is buffalo”
ELSE
    print “I’m buffalo”
END IF
```

```
IF AA = 100 THEN
    AA = 100 - 50
    print “Result is ”
    print AA
END IF
```

6. คอมพิวเตอร์สามารถปฏิบัติการซ้ำ คำที่ใช้คือ *DOWHILE และ END DO* เช่น

```
DOWHILE  cc < 50
    print “Please input cc :”
    Read cc
ENDDO
```

```
DOWHILE  cc < 50
    print cc
    cc = cc + 1
ENDDO
```

ตัวอย่าง 3.1 จงเขียนรหัสเทียมเพื่อเปรียบเทียบค่าข้อมูลที่เก็บในตัวแปร x โดยมีเงื่อนไขดังนี้

- ❑ ถ้า x >0 ให้แสดงคำว่า “Positive Number”
- ❑ ถ้า x <0 ให้แสดงคำว่า “Negative Number”
- ❑ ถ้า x =0 ให้แสดงคำว่า “Zero Number”

■ วิธีที่ 1

```
เริ่มต้น
    อ่านค่า x
    IF x>0 THEN
        พิมพ์ "Positive Number"
    ELSE
        IF x<0 THEN
            พิมพ์ "Negative
Number"
        ELSE
            พิมพ์ "Zero
Number"
        ENDIF
    ENDIF
จบงาน
```

■ วิธีที่ 2

```
START
   Read x
   IF x>0 THEN
      Print "Positive Number"
   ELSE
      IF x<0 THEN
         Print "Negative Number"
      ELSE
         Print "Zero  Number"
      ENDIF
   ENDIF
STOP
```

ตัวอย่าง 3.2 จงเขียนรหัสเทียมแสดงการเพิ่มของข้อมูล ตัวเลขที่เก็บอยู่ในหน่วยความจำของตัวแปร J โดยมีค่าเริ่มต้นจาก 0 ให้ทำการเพิ่มค่าขึ้นไปเรื่อยๆ ทีละ 1 จนกระทั่ง J มีค่าข้อมูลมากกว่า 100 จึงจะหยุดการทำงาน

■ วิธีการแบบ DO WHILE

```
START
    Set J to 0
    DO WHILE J<=100
        Add 1 to J
    ENDDO
STOP
```

■ วิธีการแบบ DO UNTIL

```
START
    Set J to 0
    DO
        Add 1 to J
    UNTIL J>100
STOP
```

CS.Rambhai 3-28