การเขียนโปรแกรมโดยใช้ภาษาสัญลักษณ์

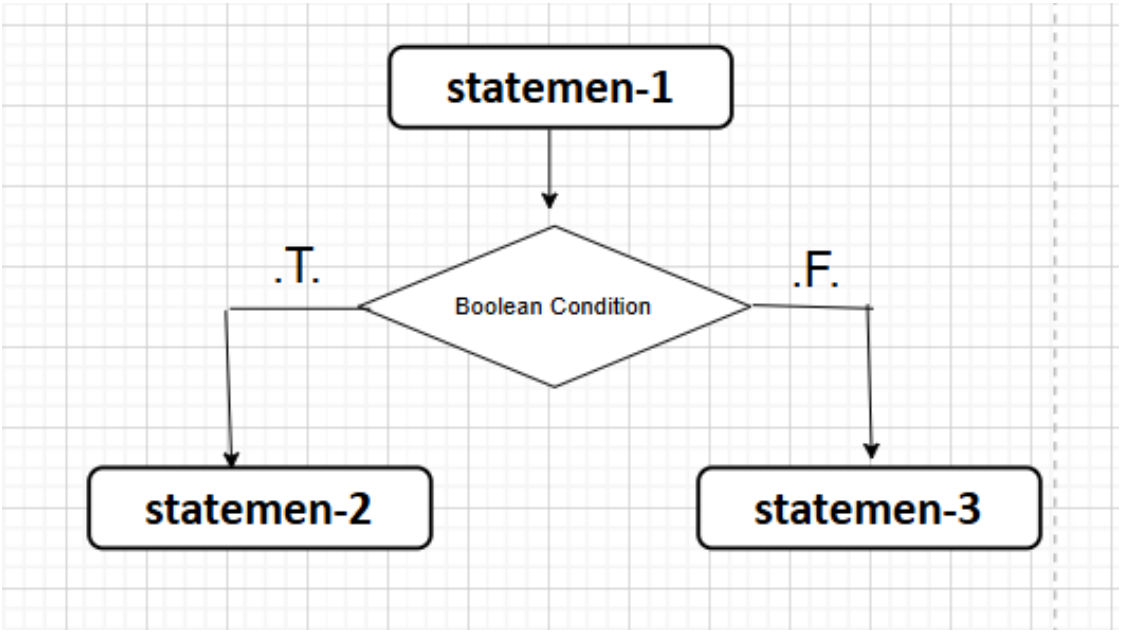
Basic Repetition flowchart

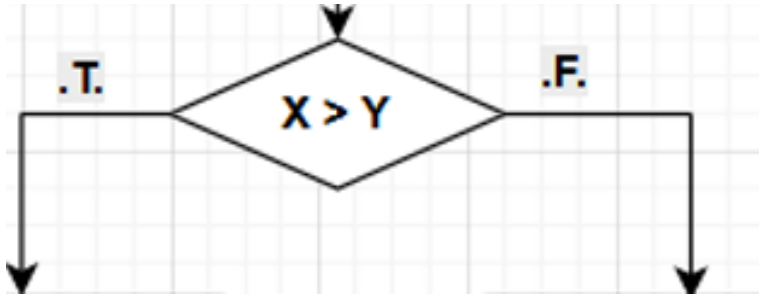
Flowchart

- เครื่องมือในการบรรยายการทำงานของ statement ของโปรแกรม

- Symbol

Symbol	Name	Function
	Start/end	An oval represents a start or end point
	Arrows	A line is a connector that shows relationships between the representative shapes
	Process	A rectangle represents a process
	Decision	A diamond indicates a decision





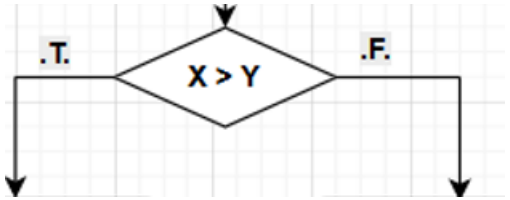
Boolean Datatype: TRUE FALSE

main.pas

```
1  program CalculateNet;
2  var
3    flag: boolean;
4  begin
5    flag :=  true;
6    writeln( 'flag = ', flag);
7    flag :=  false;
8    writeln( 'flag = ', flag);
9
10   // readln( flag ); { compile error }
11
12 end.
13
```

Pascal ไม่สามารถใช้คำสั่ง read() / readln() กับ data boolean ได้

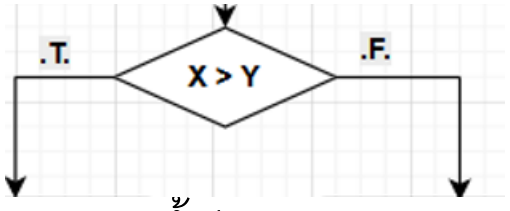
Relation Expression



- **Relation expression)** ใช้เปรียบเทียบค่าระหว่างตัวแปรหรือตัวดำเนินการ 2 ตัวขึ้นไป
- ผลลัพธ์เป็น ค่าความจริง (**Boolean)** คือ **true** หรือ **false**

สัญลักษณ์	ความหมาย	ตัวอย่าง	ผลลัพธ์
=	เท่ากันหรือไม่	5 = 5	true
<>	ไม่เท่ากันหรือไม่	5 <> 3	true
>	มากกว่า	7 > 2	True
<	น้อยกว่า	2 < 4	True
>=	มากกว่าหรือเท่ากับ	5 >= 5	True
<=	น้อยกว่าหรือเท่ากับ	3 <= 2	false

Relation Expression



- Relation expression) ใช้เปรียบเทียบค่าระหว่างตัวแปรหรือตัวดำเนินการ 2 ตัวขึ้นไป
- ผลลัพธ์เป็น ค่าความจริง (Boolean) คือ true หรือ false
- การเขียน relation expression ในภาษาโปรแกรมจะสามารถเขียนในลักษณะนี้ได้:

X > Y > Z ต้องเขียนเป็น X > Y and Y > Z

โดย and ทำหน้าที่เชื่อม relational expression X>Y กับ Y > Z

โอเปอร์เรเตอร์ AND เรียกว่า Boolean expression

Boolean Expression

x = 6
y = 10

- ทำหน้าที่เชื่อม relation expression
- ผลลัพธ์เป็น ค่าความจริง (Boolean) คือ true หรือ false

สัญลักษณ์	ความหมาย	ตัวอย่าง	ผลลัพธ์
and	และ (and,&&)	x > 5 and y < 10	false
or	หรือ (or,||)	x == 0 or y == 0	false
not	ไม่ (not,!)	not(x == 5)	True

x	y	x && y
T (1)	T (1)	T (1)
T (1)	F (0)	F (0)
F (0)	T (1)	F (0)
F (0)	F (0)	F (0)

x	y	x || y
T (1)	T (1)	T (1)
T (1)	F (0)	T (1)
F (0)	T (1)	T (1)
F (0)	F (0)	F (0)

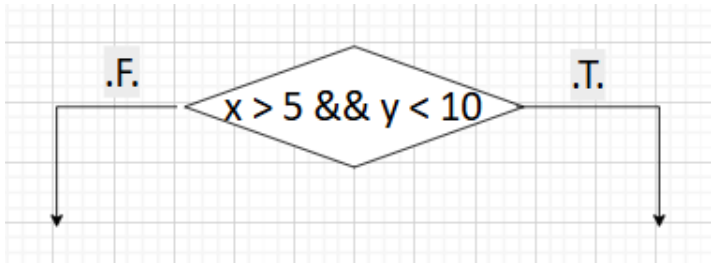
x	! x
T (1)	F (0)
F (0)	T (1)

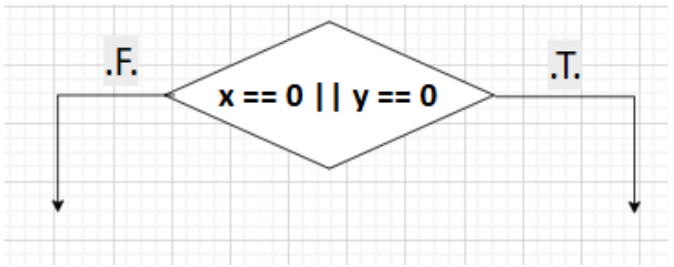
Boolean Expression

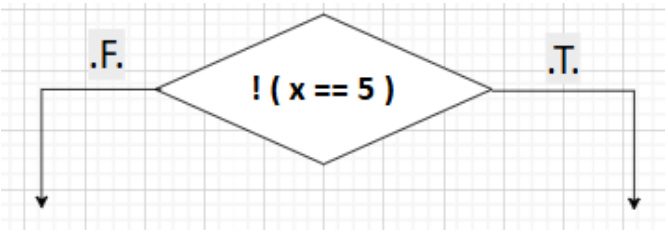
x = 6
y = 10

- ทำหน้าที่เชื่อม **relation expression**
- ผลลัพธ์เป็น ค่าความจริง (**Boolean**) คือ **true** หรือ **false**

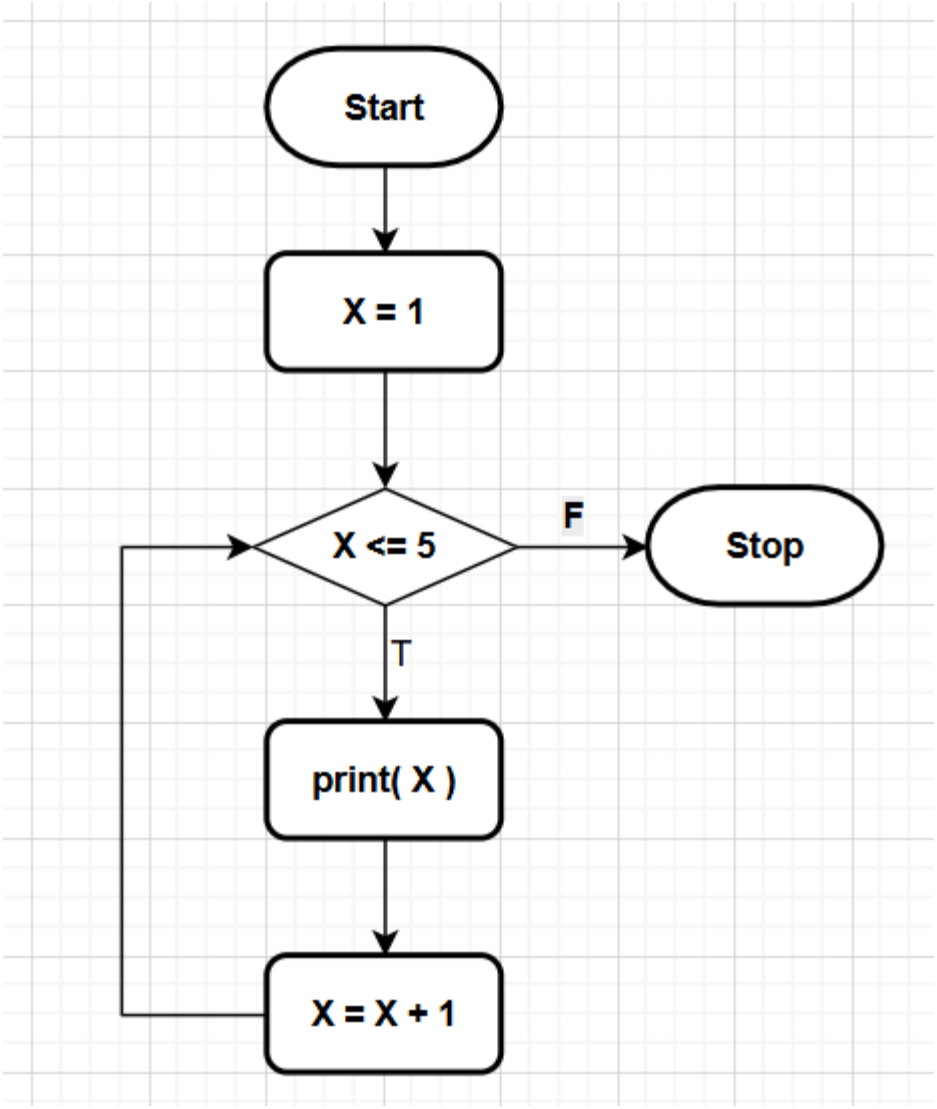
สัญลักษณ์	ความหมาย	ตัวอย่าง	ผลลัพธ์
and	และ (&&)	x > 5 and y < 10	false
or	หรือ (||)	x = 0 or y = 0	false
not	ไม่ (!)	not(x == 5)	True

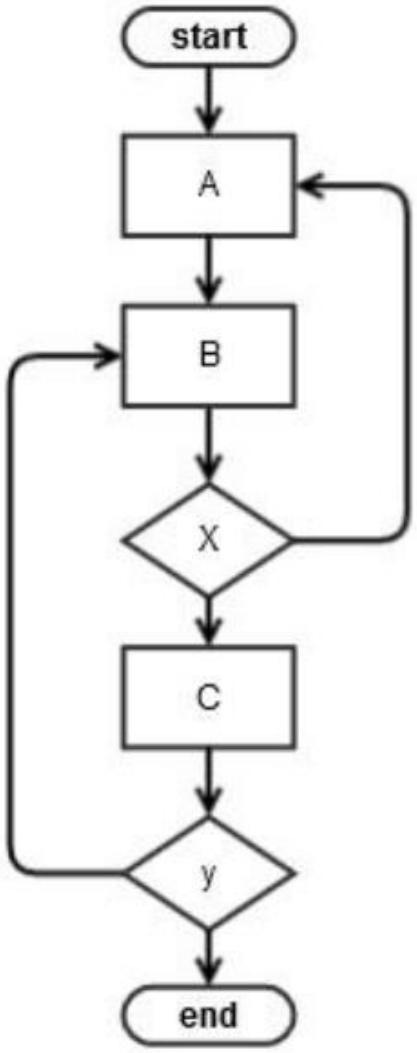


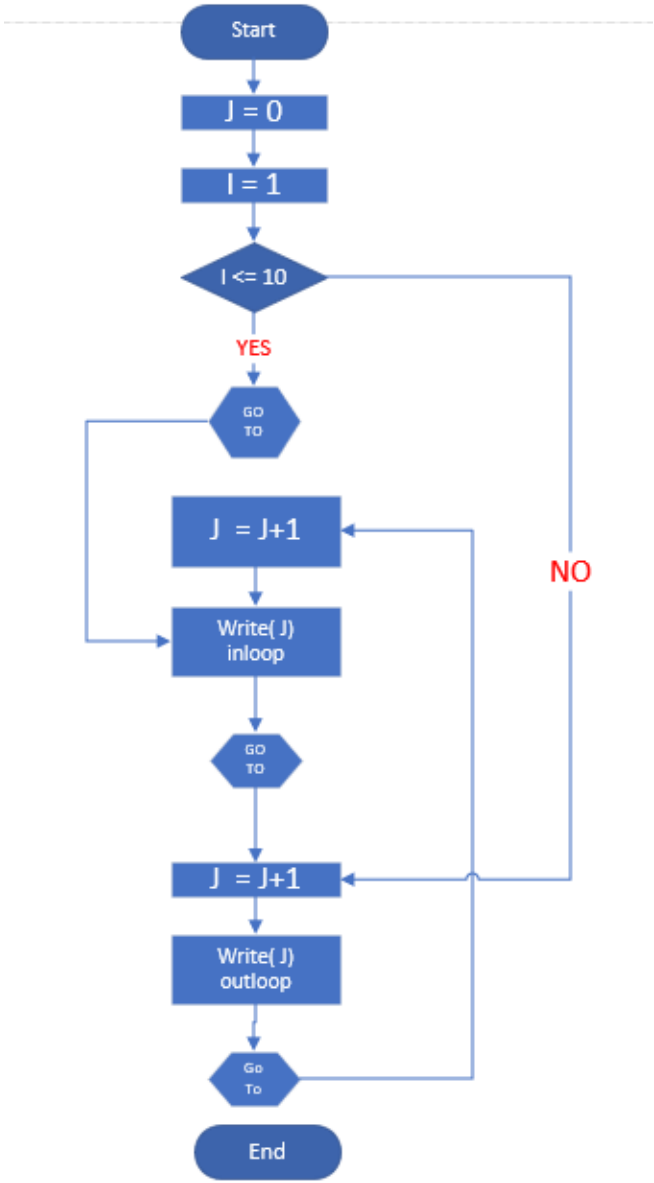




ลักษณะของ repetition flow

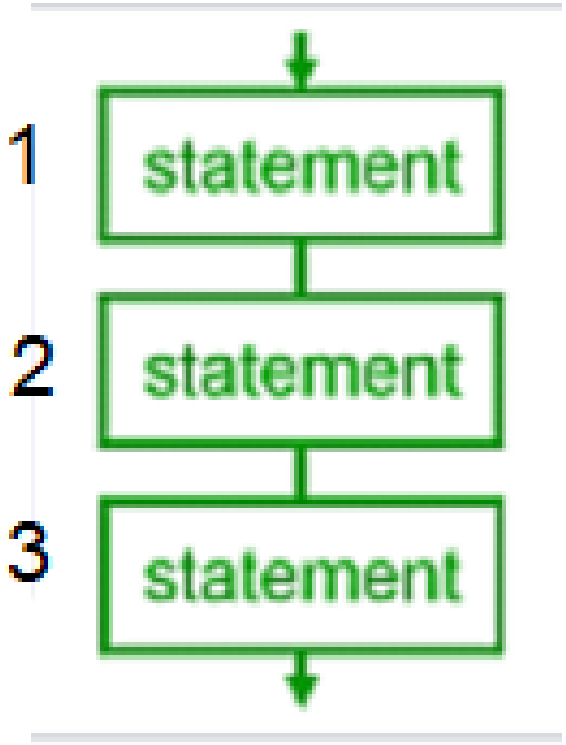


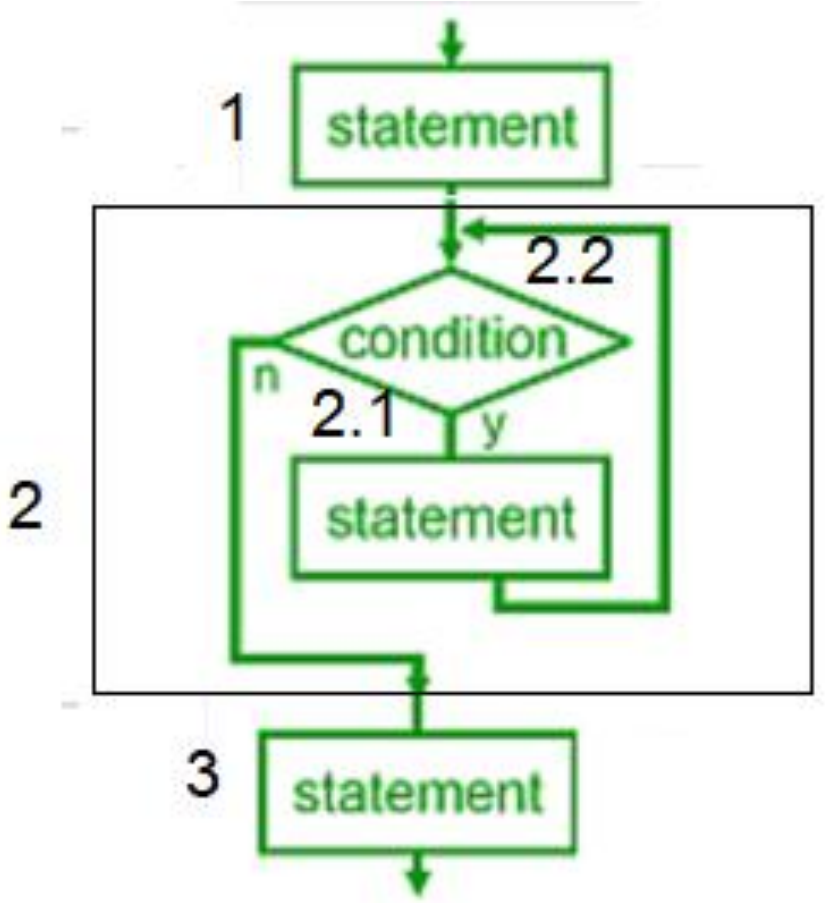


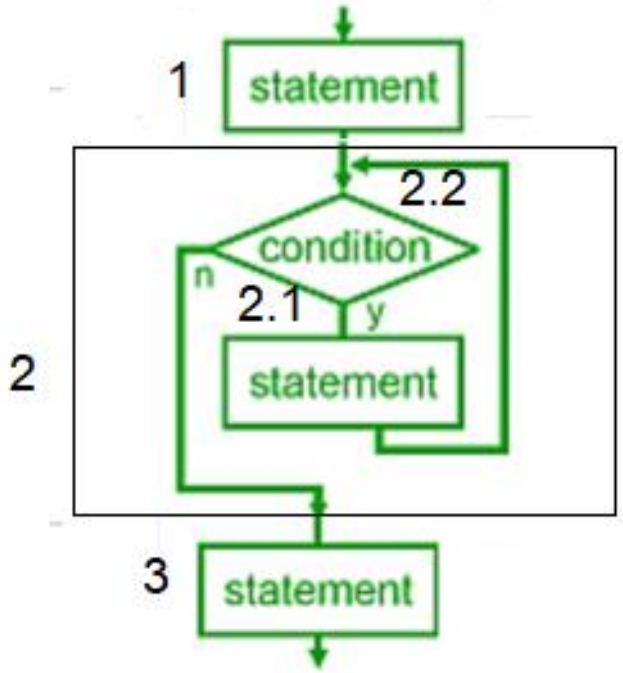


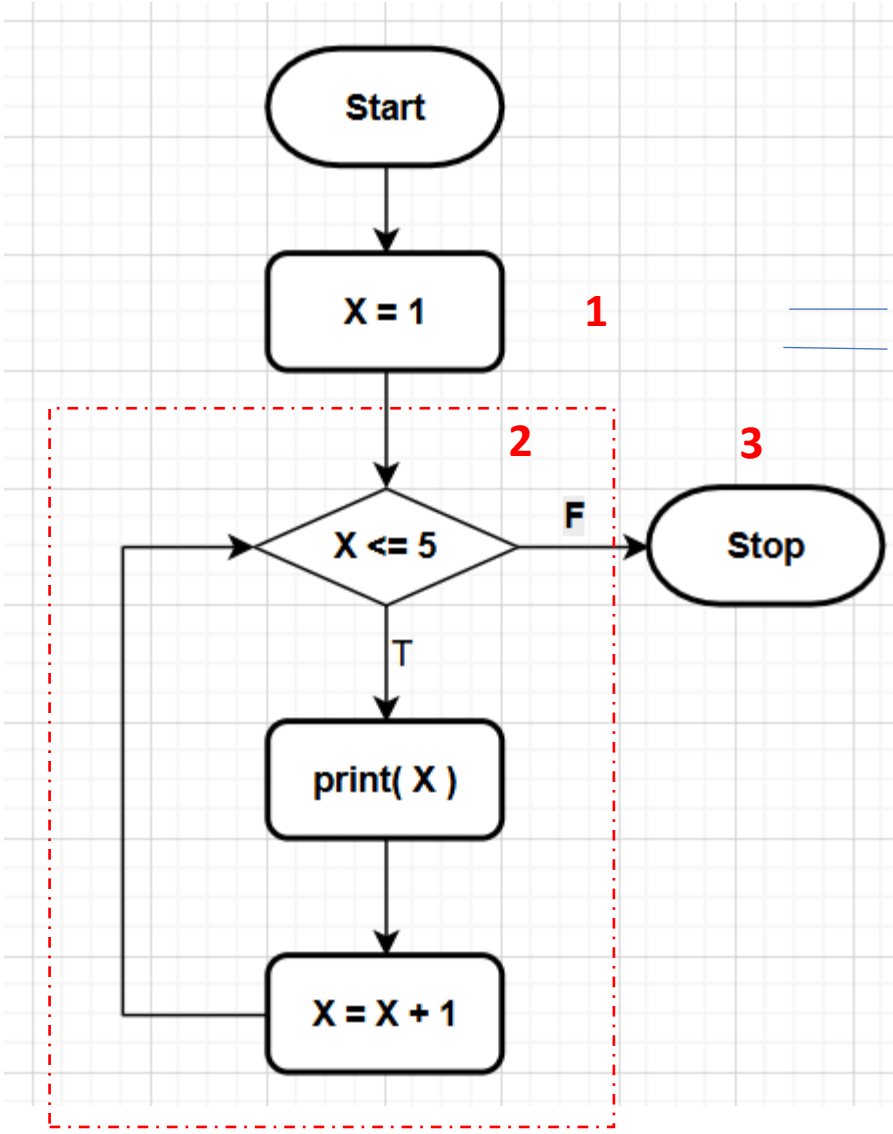
Structure programming in repetition flow

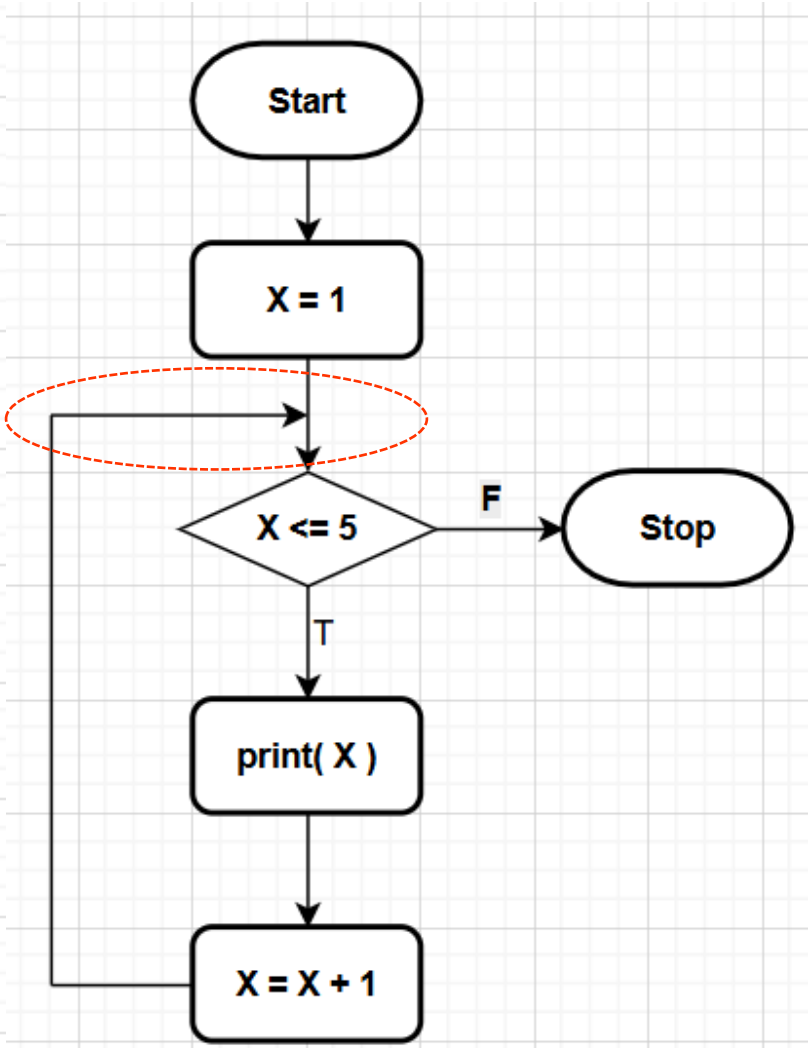
- Based on CPU execution flow ; each of logical statement has a single entry and single exist point





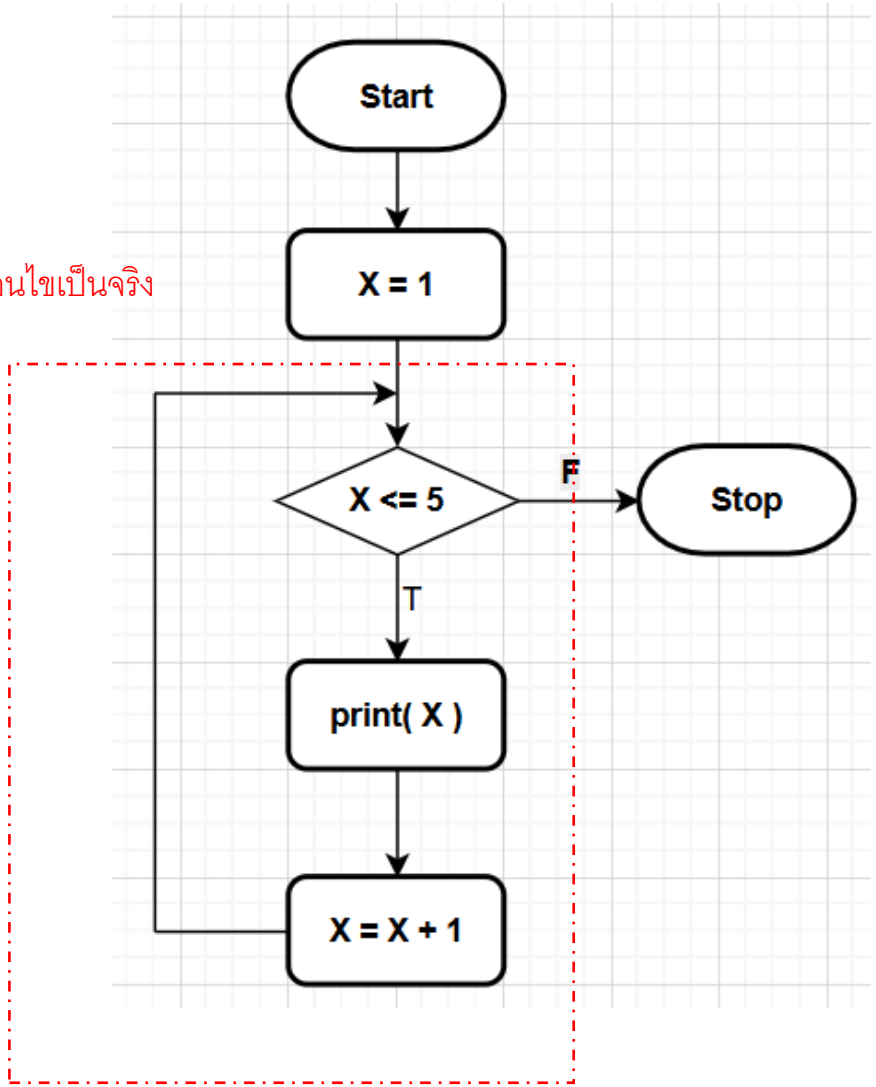

1
statement
2.2
condition
n
2.1
y
2
statement
3
statement


Start
X = 1
1
2
3
X <= 5
F
Stop
T
print(X)
X = X + 1

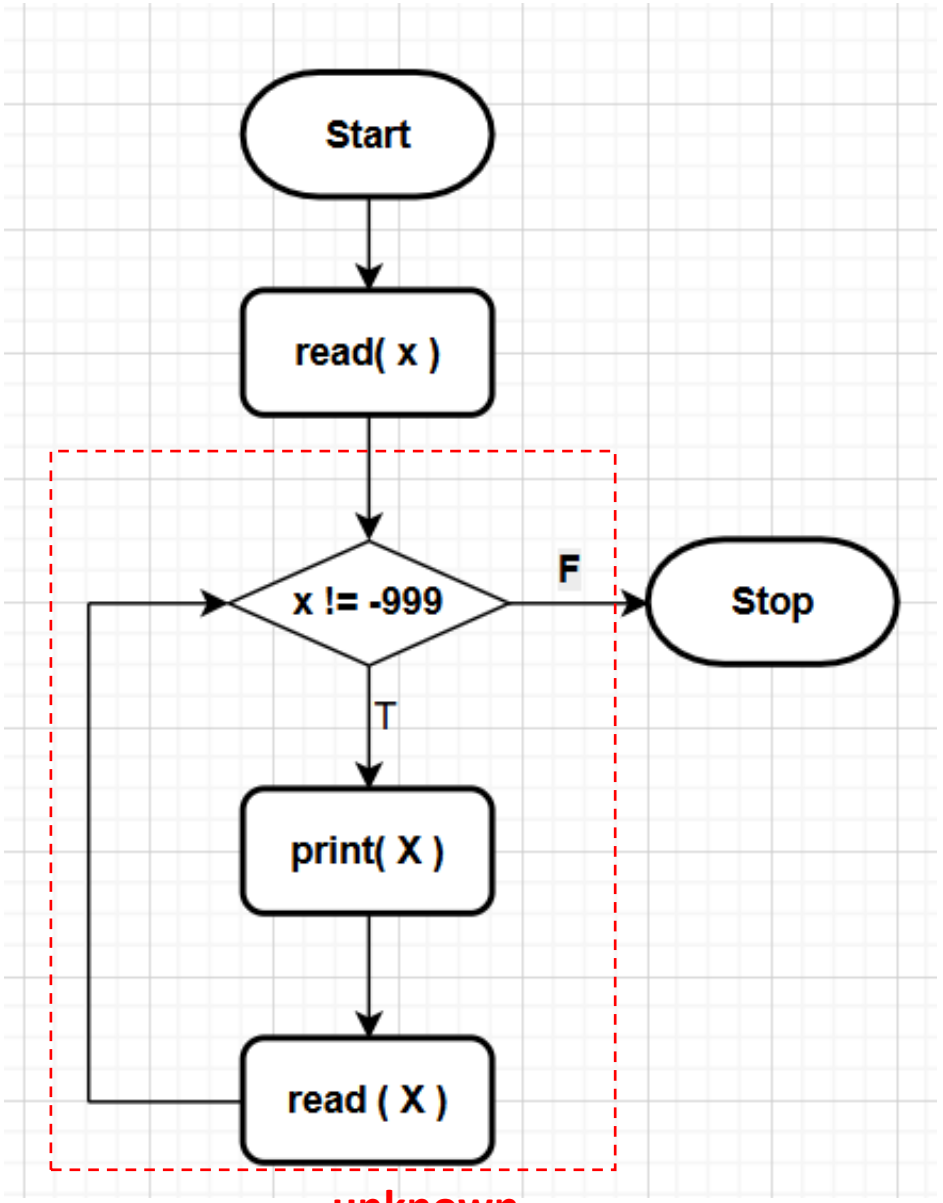

Start
X = 1
X <= 5
F
Stop
T
print(X)
X = X + 1

ลักษณะของ repetition 2 แบบ : known &unknown repetitions

จุดสังเกตทำขณะเงื่อนไขเป็นจริง

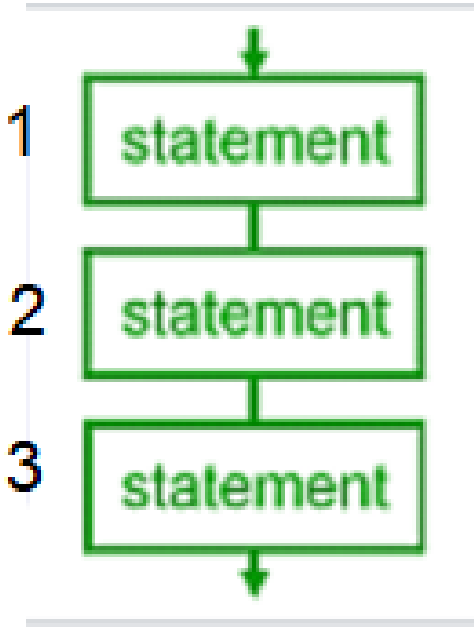


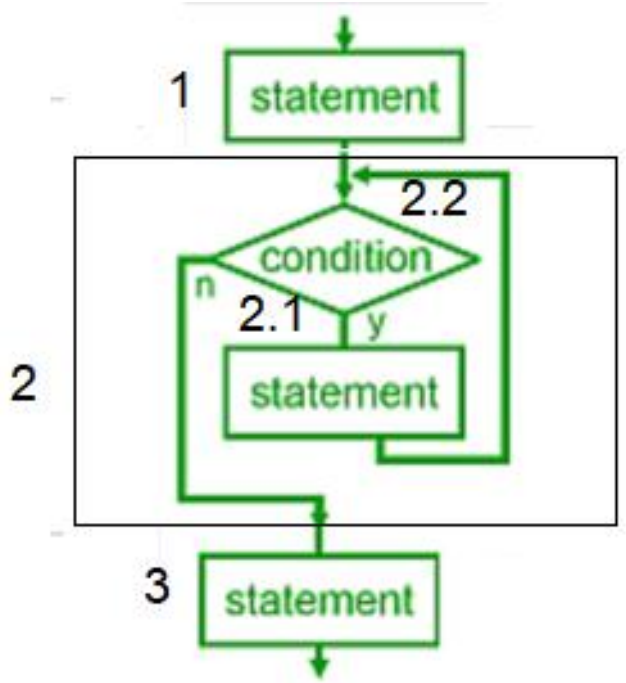
known

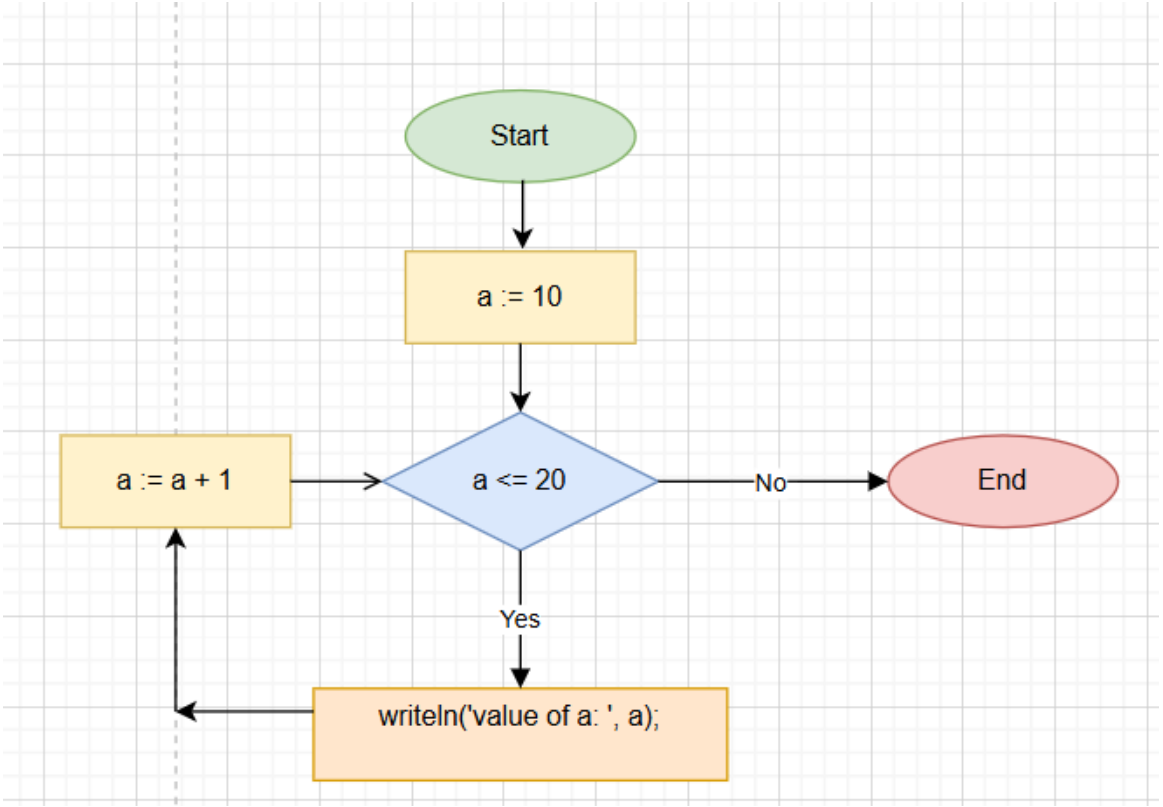


unknown

FOR – Stm : known repetitions





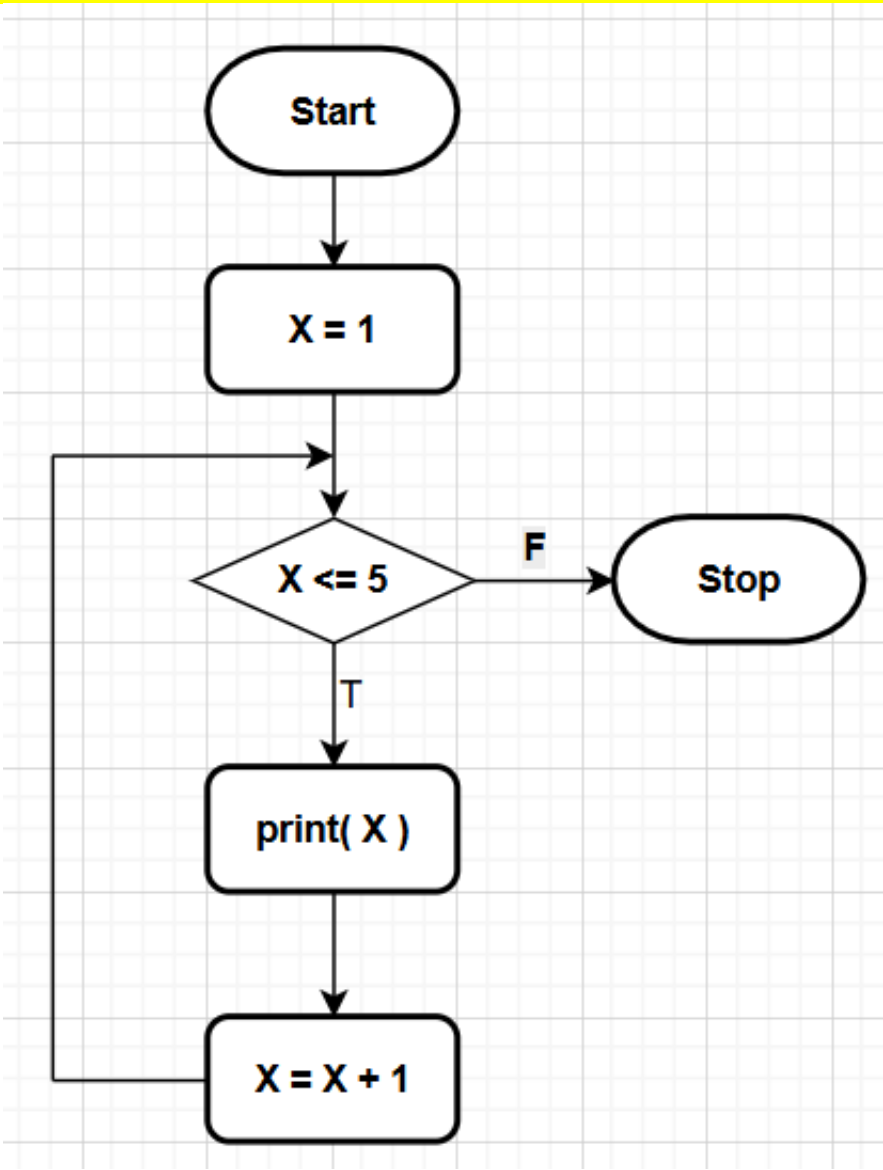


Loop / Repetition flow : Known number of repetitions

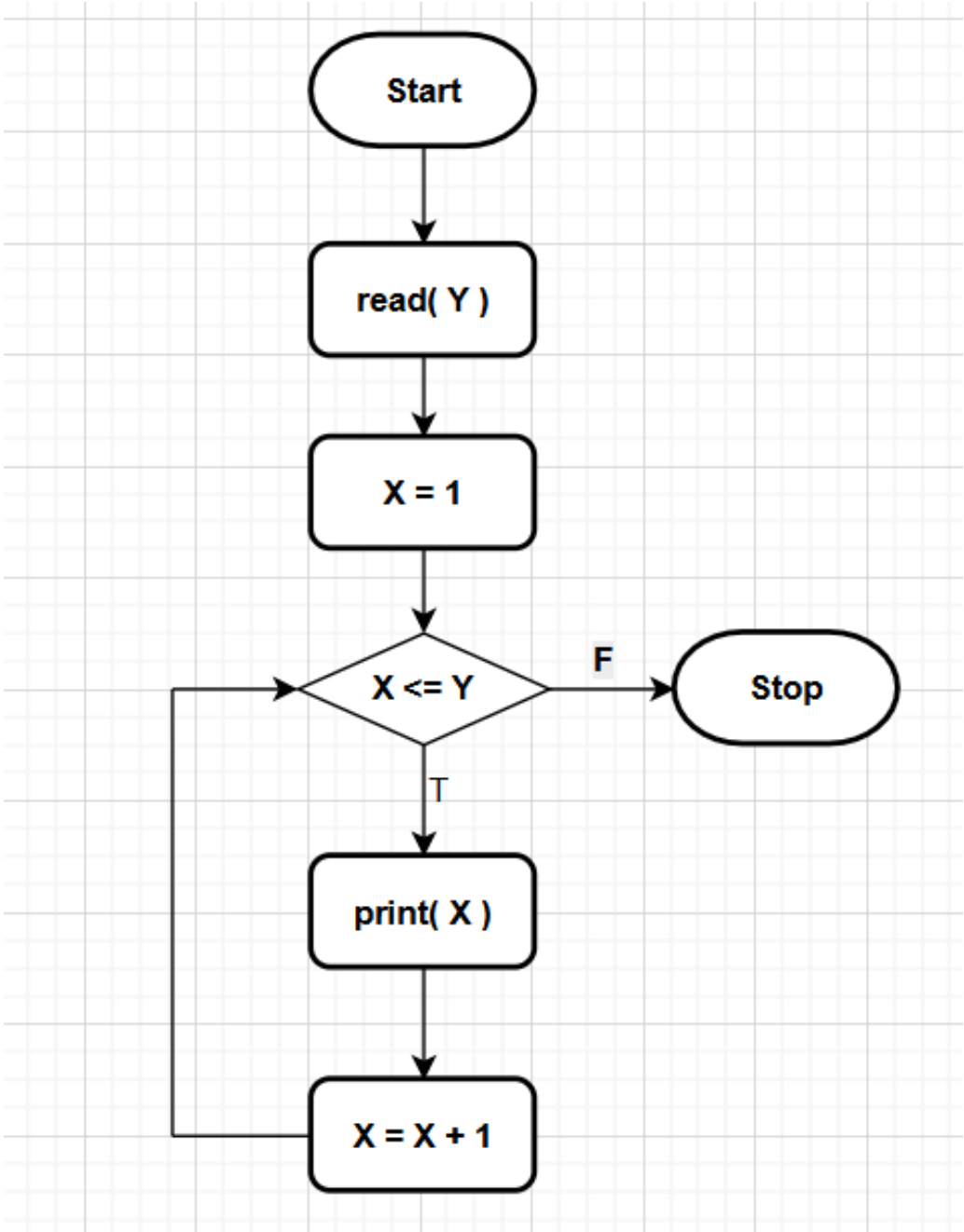
- flow นี้อ่านว่าอะไร ?

องค์ประกอบของ known number of repetition flow

1. ตัวนับ x
2. เงื่อนการนับ X<5 เพื่อตรวจสอบการทำซ้ำ
3. เพิ่มจำนวนของตัวนับ X



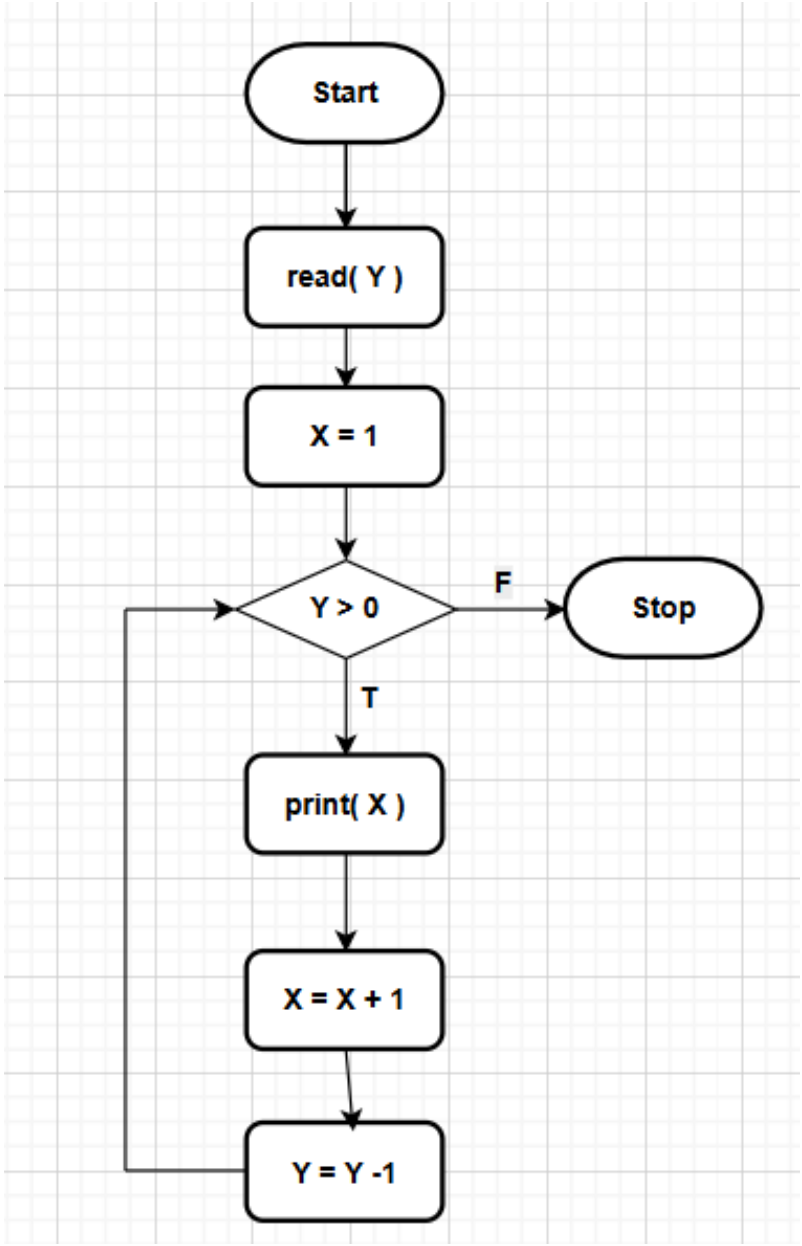
- flow นี้อ่านว่าอะไร ?

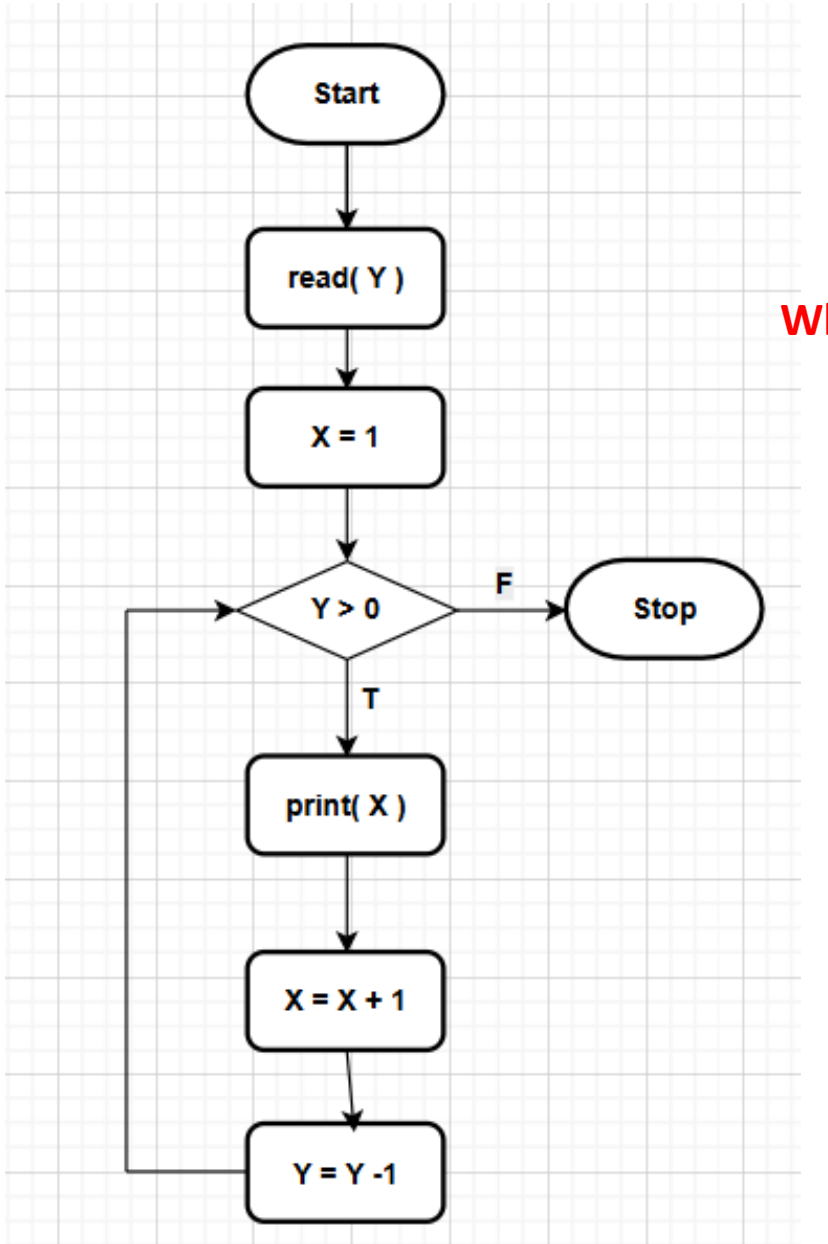


- flow นี้อ่านว่าอะไร ?

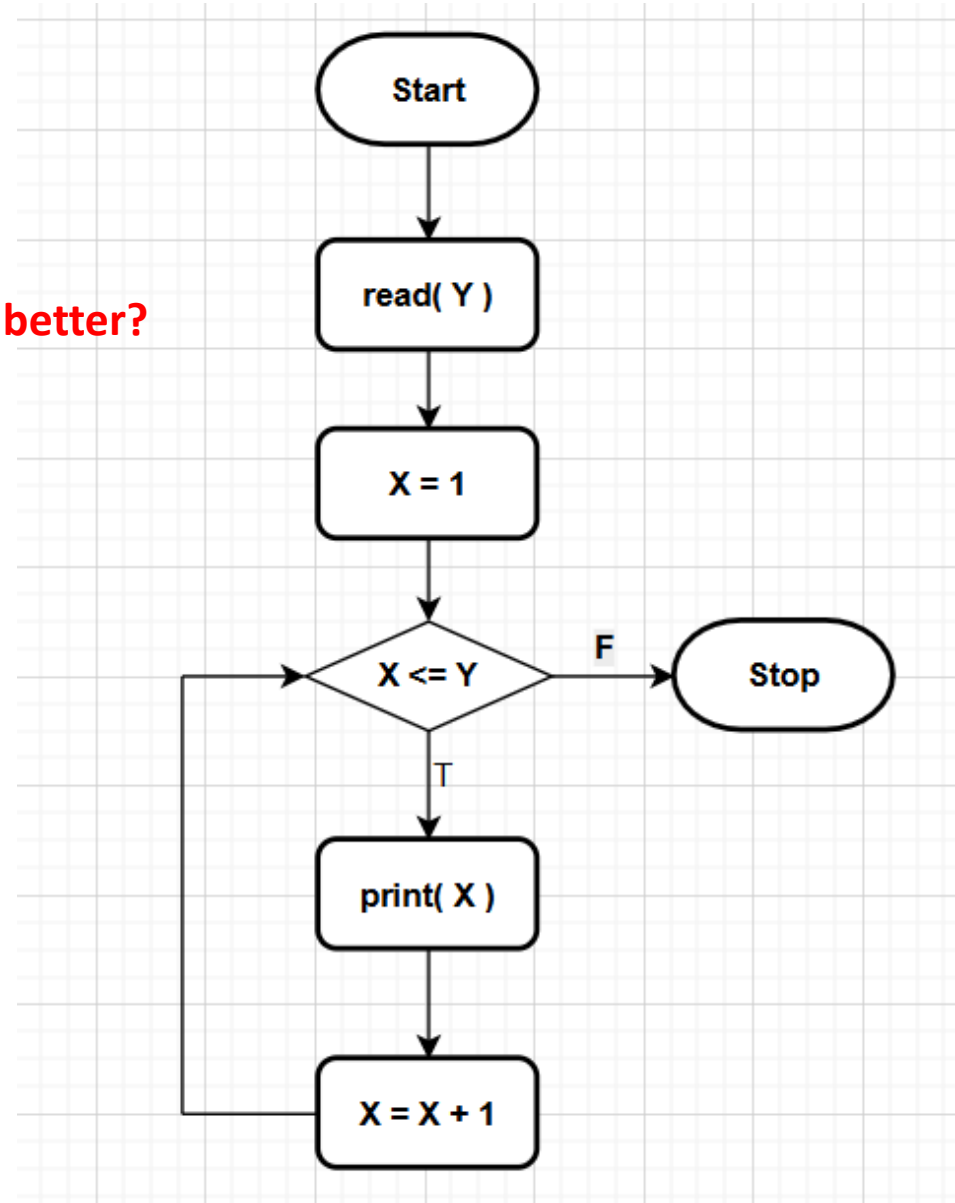
1.ตัวแปรอะไรคือ ตัวนับ

2.ลักษณะการนับเป็นแบบ forward หรือ backward

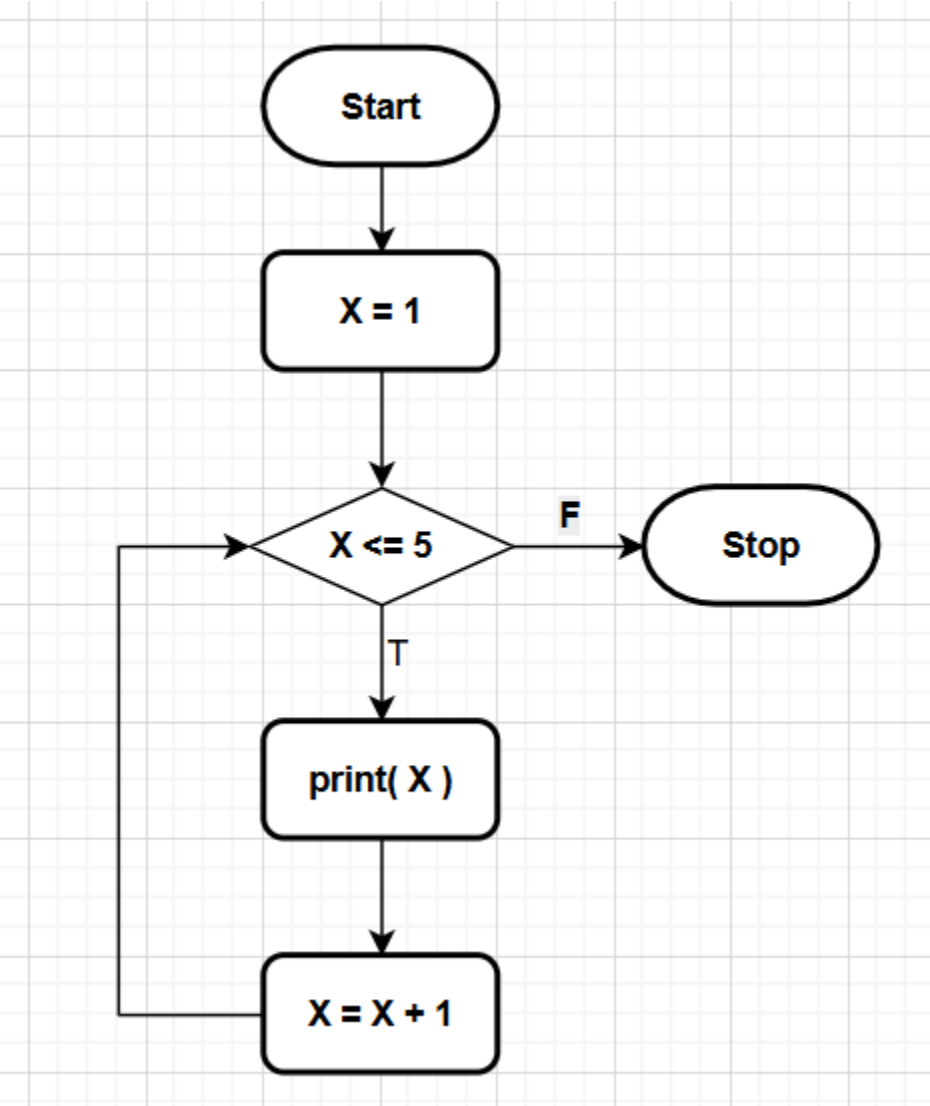




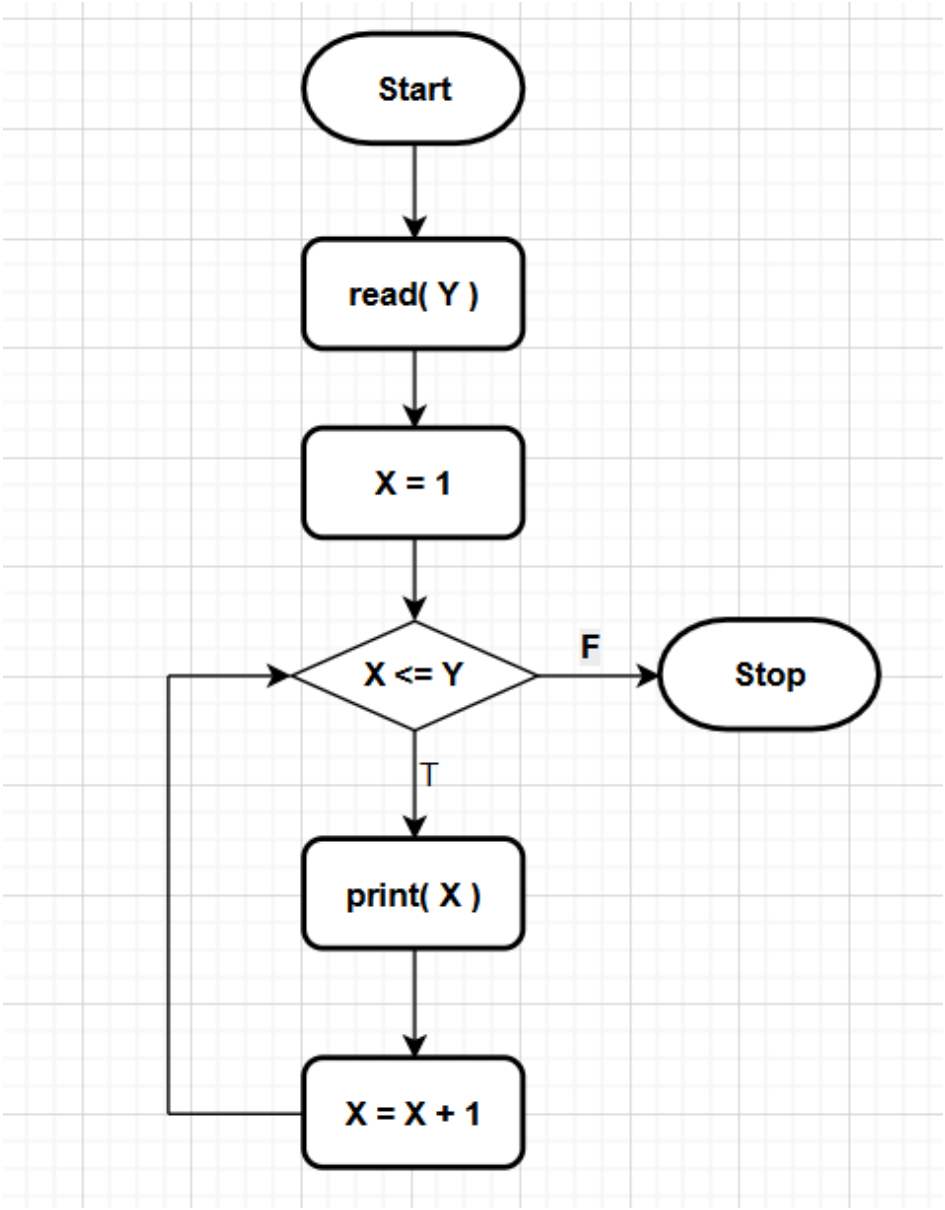
Which one is better?



เปรียบเทียบ

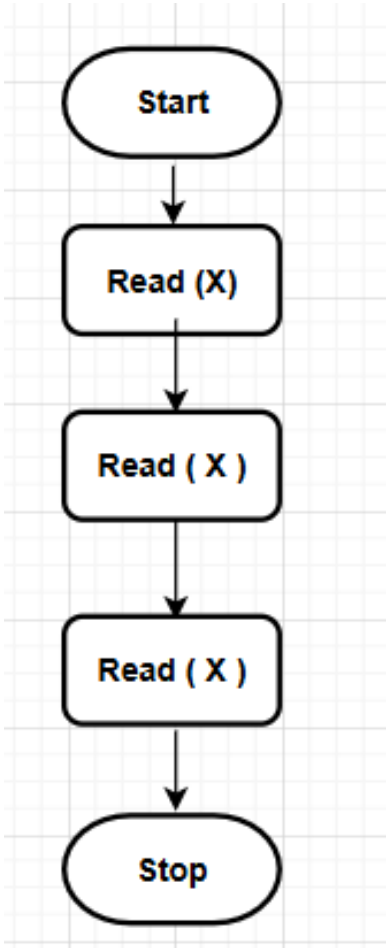


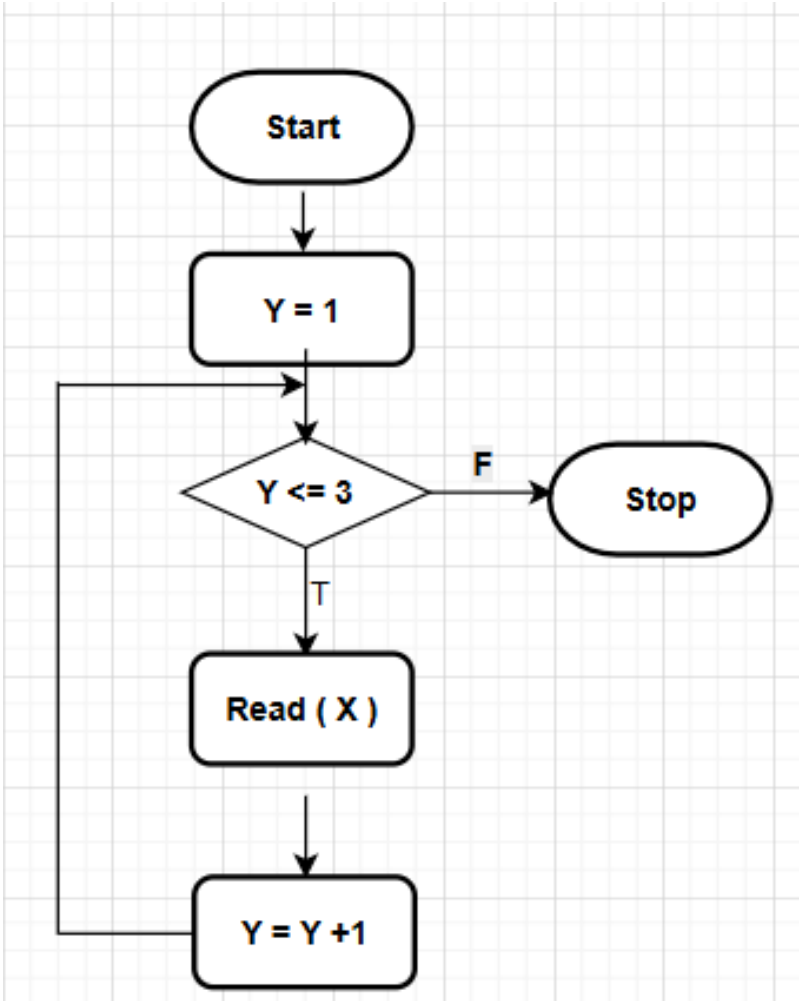
จำนวนทำซ้ำขึ้นอยู่กับ Y ที่รับมาทางแป้นพิมพ์



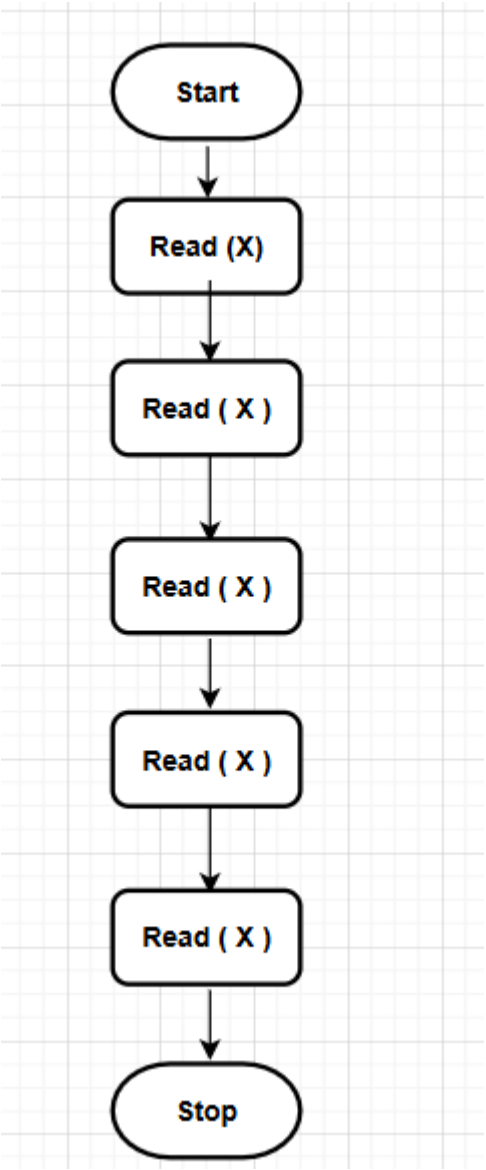
การพิจารณาว่าจะเลือก flow แบบทำซ้ำ

1.จงเขียนโปรแกรมรับข้อมูล 3 ตัว

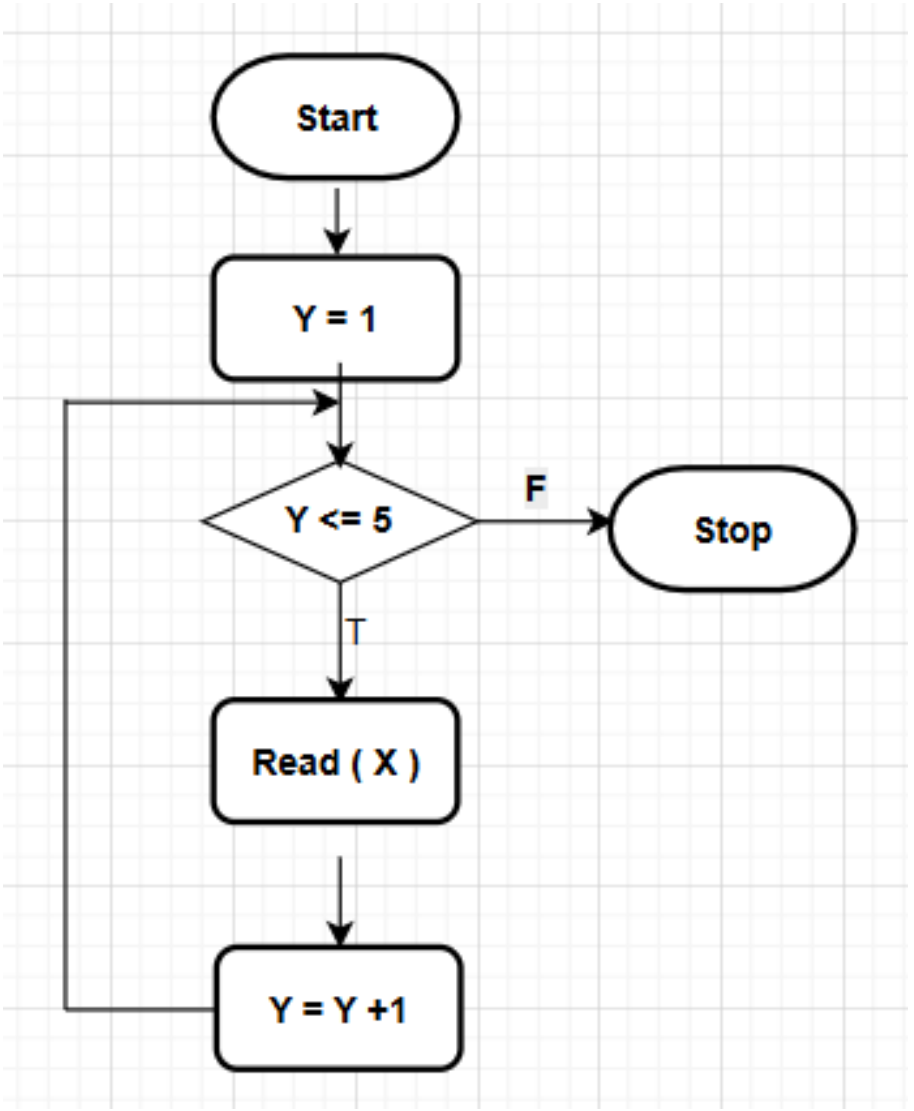




การพิจารณาว่าจะเลือก flow แบบทำซ้ำ



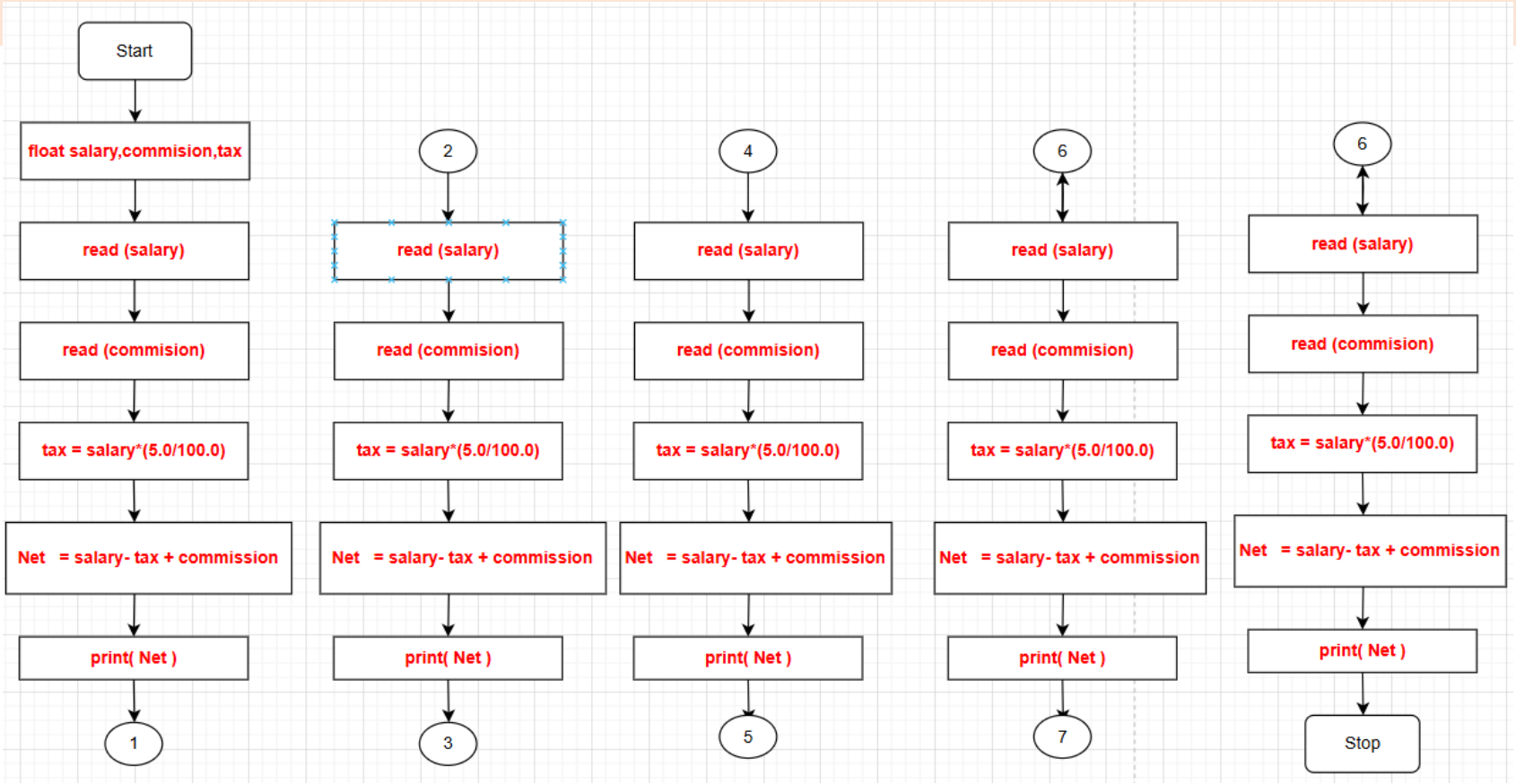
จงเขียนโปรแกรมรับข้อมูล 5 ตัว

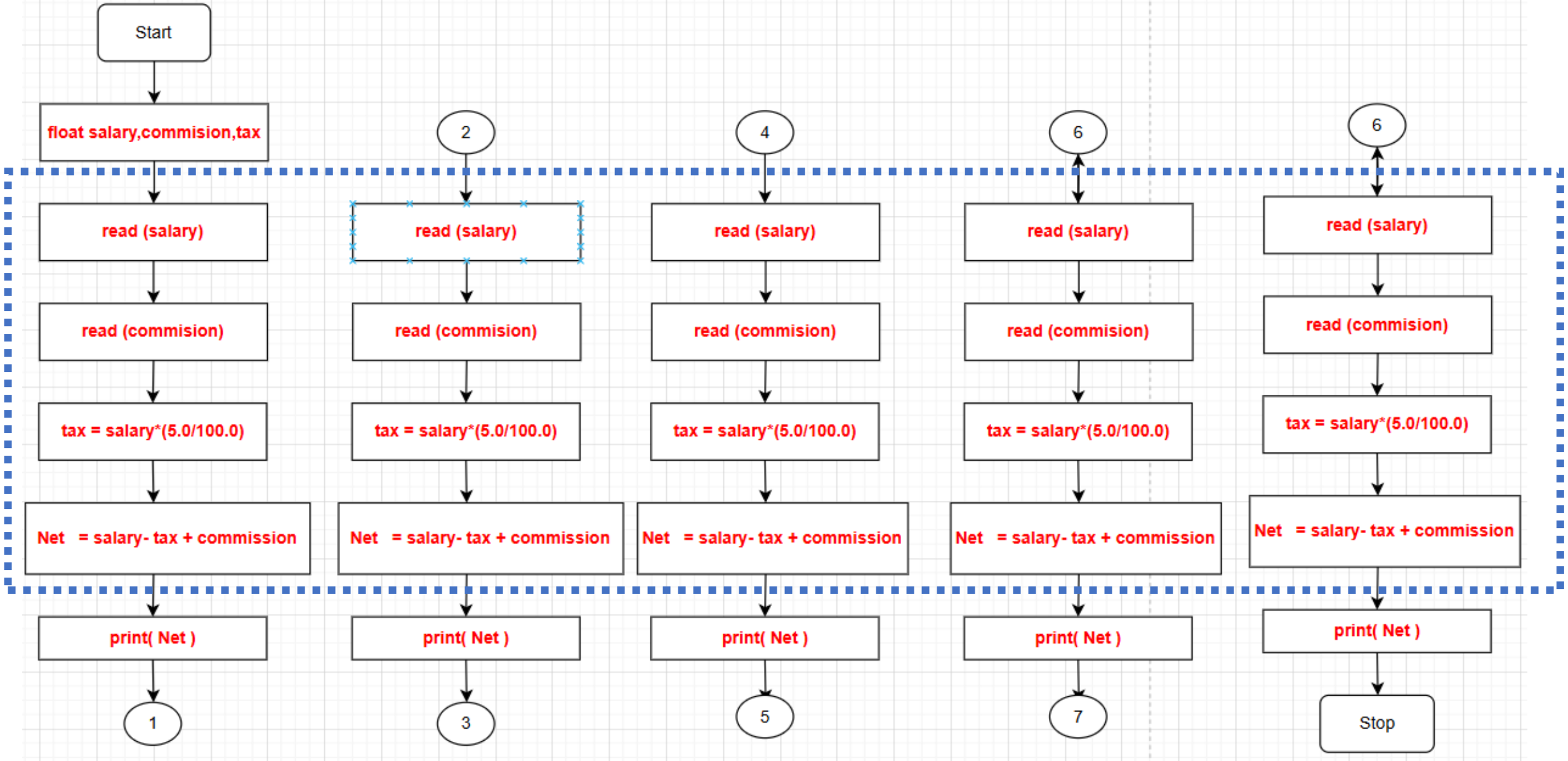


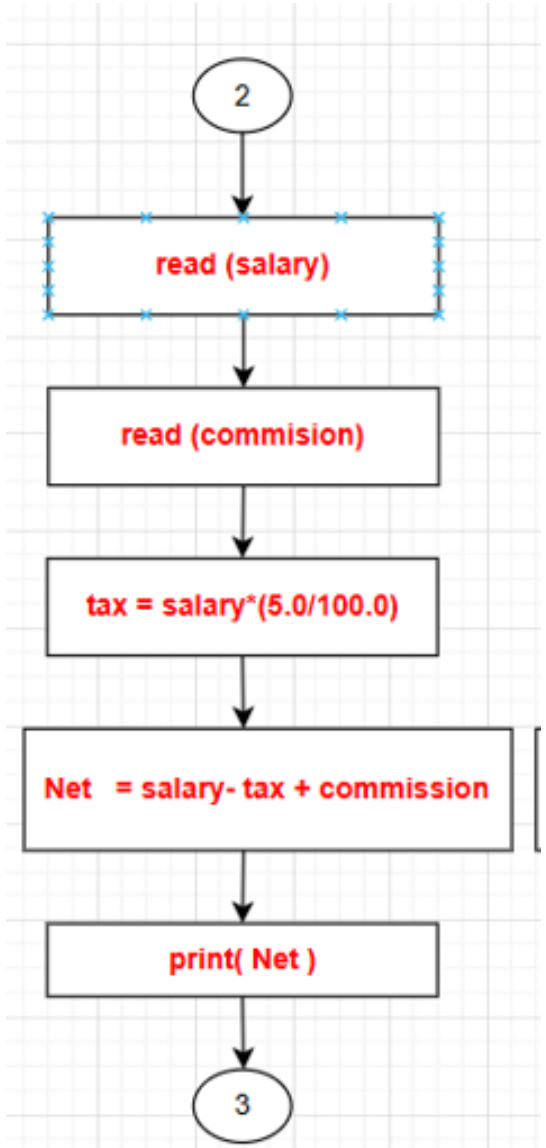
เขียนโปรแกรมรับข้อมูลของพนักงานจำนวน **5** คน เพื่อคำนวณเงินเดือนสุทธิของพนักงาน โดยมีละเอียดข้อมูลดังนี้

- รับเงินเดือนพื้นฐาน **(salary)**
- หักภาษี 5% **(salary* 5.0/100.0)**
- รับค่าคอมมิชชั่นเพิ่มเติม **(commission)**
- แสดงเงินเดือนสุทธิ = (เงินเดือน – ภาษี) + ค่าคอมมิชชั่น

Net = salary- (salary* 5.0/100.0) + commission

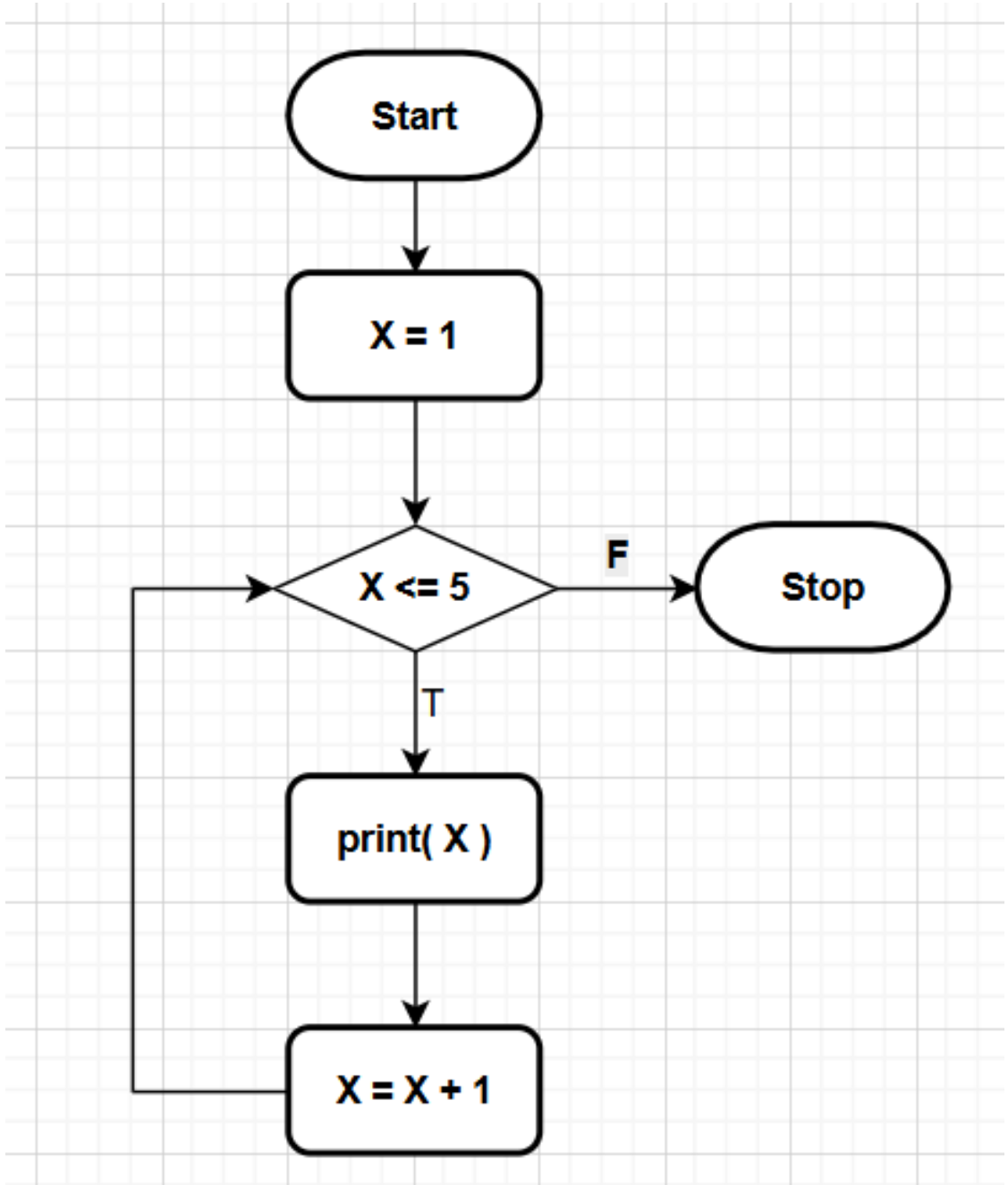


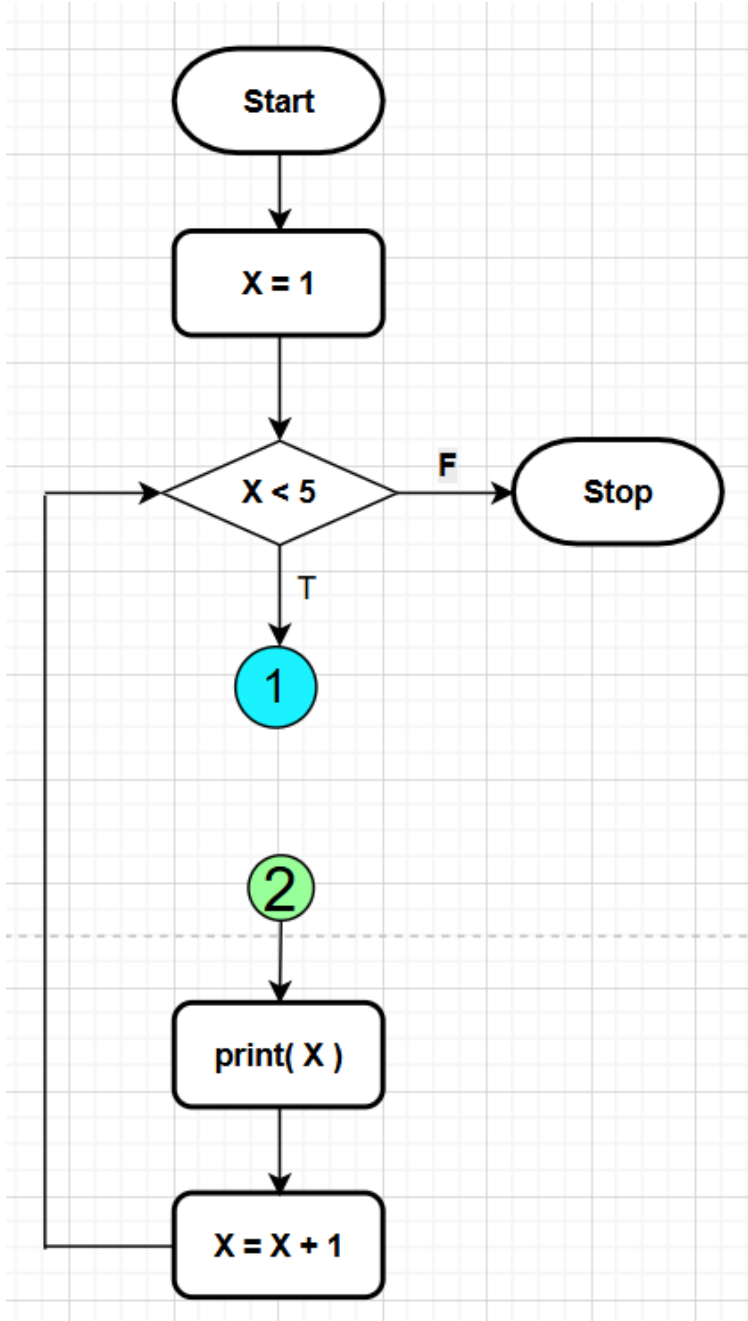

Start
float salary,commision,tax
read (salary)
read (commision)
tax = salary*(5.0/100.0)
Net = salary- tax + commission
print(Net)
1
2
read (salary)
read (commision)
tax = salary*(5.0/100.0)
Net = salary- tax + commission
print(Net)
3
4
read (salary)
read (commision)
tax = salary*(5.0/100.0)
Net = salary- tax + commission
print(Net)
5
6
read (salary)
read (commision)
tax = salary*(5.0/100.0)
Net = salary- tax + commission
print(Net)
7
6
read (salary)
read (commision)
tax = salary*(5.0/100.0)
Net = salary- tax + commission
print(Net)
Stop

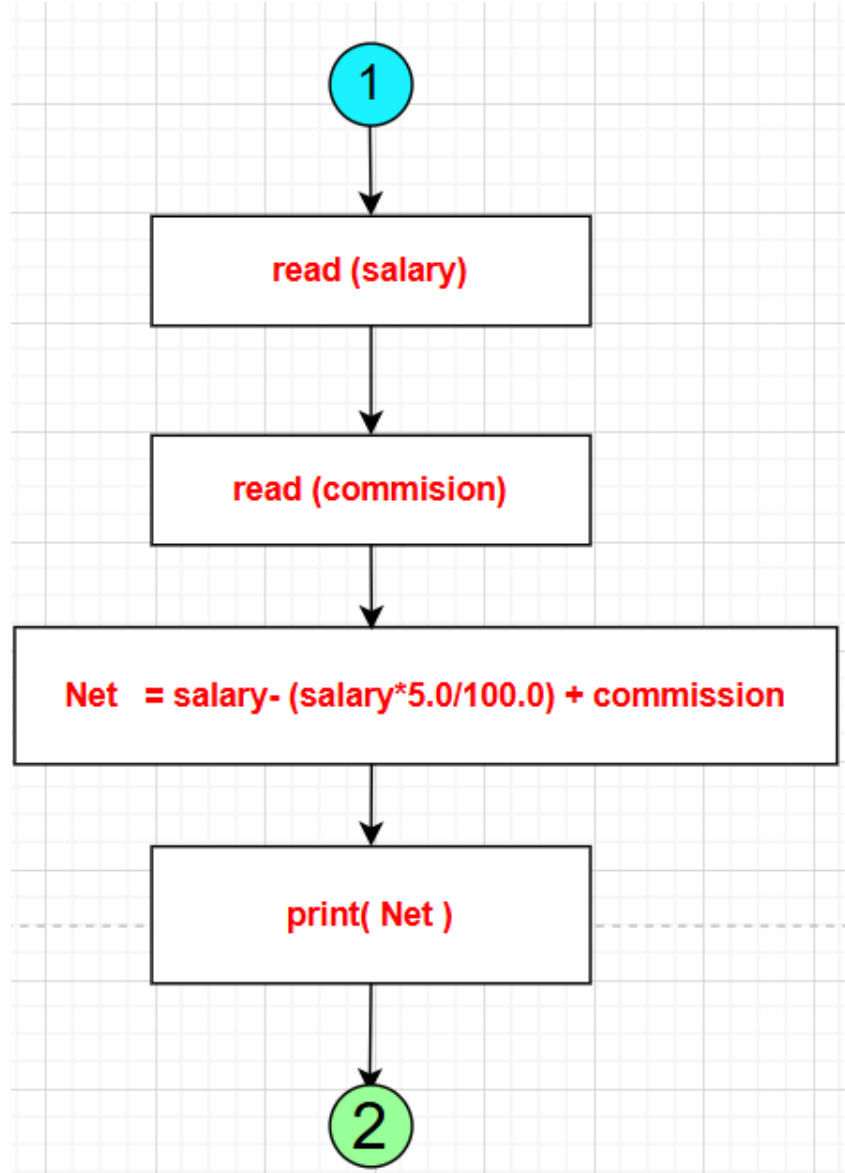

2
read (salary)
read (commision)
tax = salary*(5.0/100.0)
Net = salary- tax + commission
print(Net)
3

เพิ่ม **loop**

+


Start
X = 1
X <= 5
F
Stop
T
print(X)
X = X + 1


Start
X = 1
X < 5
F
Stop
T
1
2
print(X)
X = X + 1


1
read (salary)
read (commision)
Net = salary- (salary*5.0/100.0) + commission
print(Net)
2

เขียนโปรแกรมคำนวณเงินเดือนสุทธิของพนักงาน **โดยรับจำนวนพนักงานจากแป้นพิมพ์**

- รับเงินเดือนพื้นฐาน **(salary)**
- หักภาษี 5% **(salary* 5.0/100.0)**
- รับค่าคอมมิชชั่นเพิ่มเติม **(commission)**
- แสดงเงินเดือนสุทธิ = (เงินเดือน – ภาษี) + ค่าคอมมิชชั่น

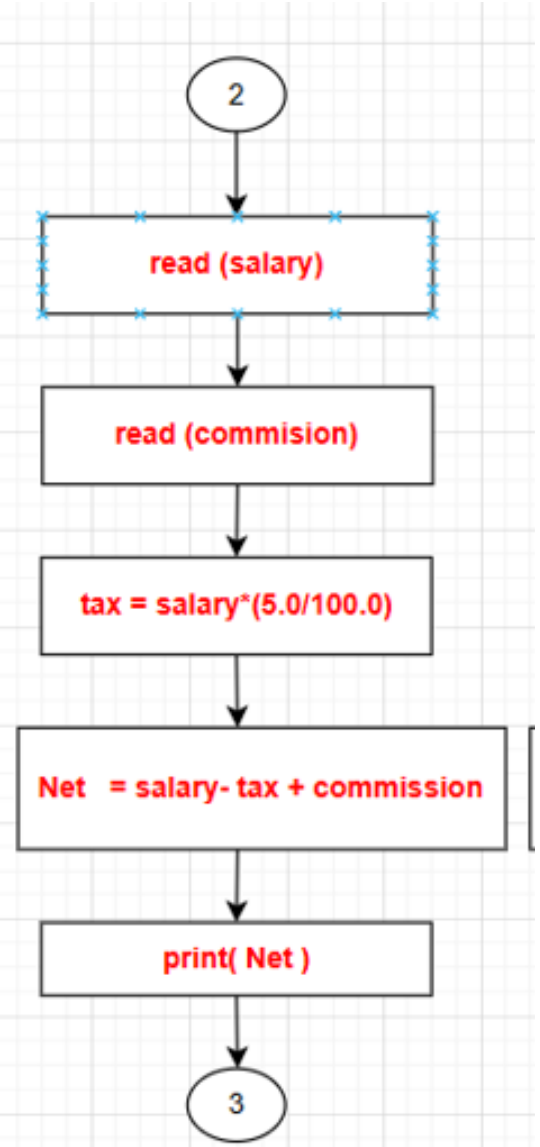
Net = salary- (salary* 5.0/100.0) + commission

เขียนโปรแกรมรับข้อมูลของพนักงาน**จำนวน 5 คน เพื่อ**คำนวณเงินเดือนสุทธิของพนักงาน โดยมีละเอียดข้อมูลดังนี้

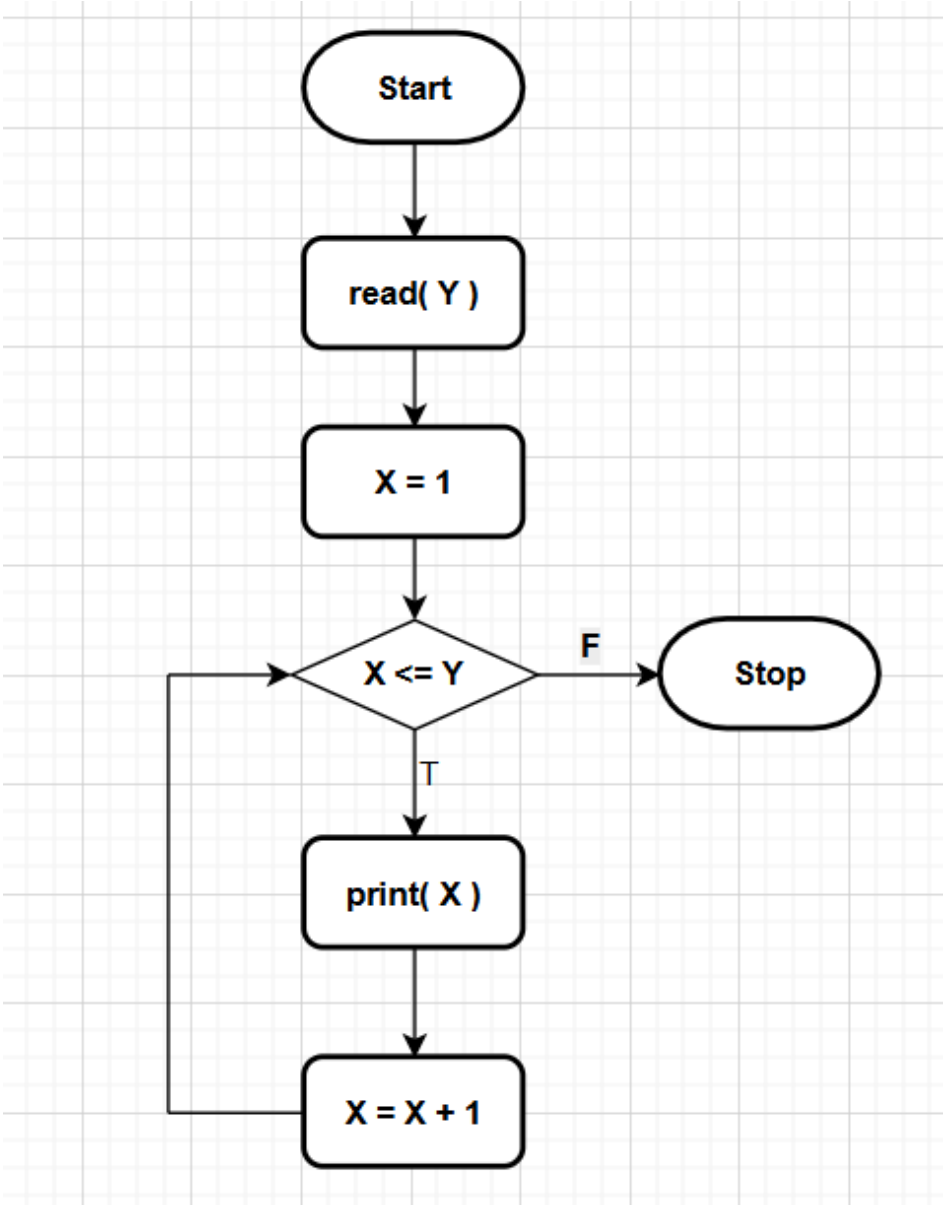
- รับเงินเดือนพื้นฐาน **(salary)**
- หักภาษี 5% **(salary* 5.0/100.0)**
- รับค่าคอมมิชชั่นเพิ่มเติม **(commission)**
- แสดงเงินเดือนสุทธิ = (เงินเดือน – ภาษี) + ค่าคอมมิชชั่น

Net = salary- (salary* 5.0/100.0) + commission

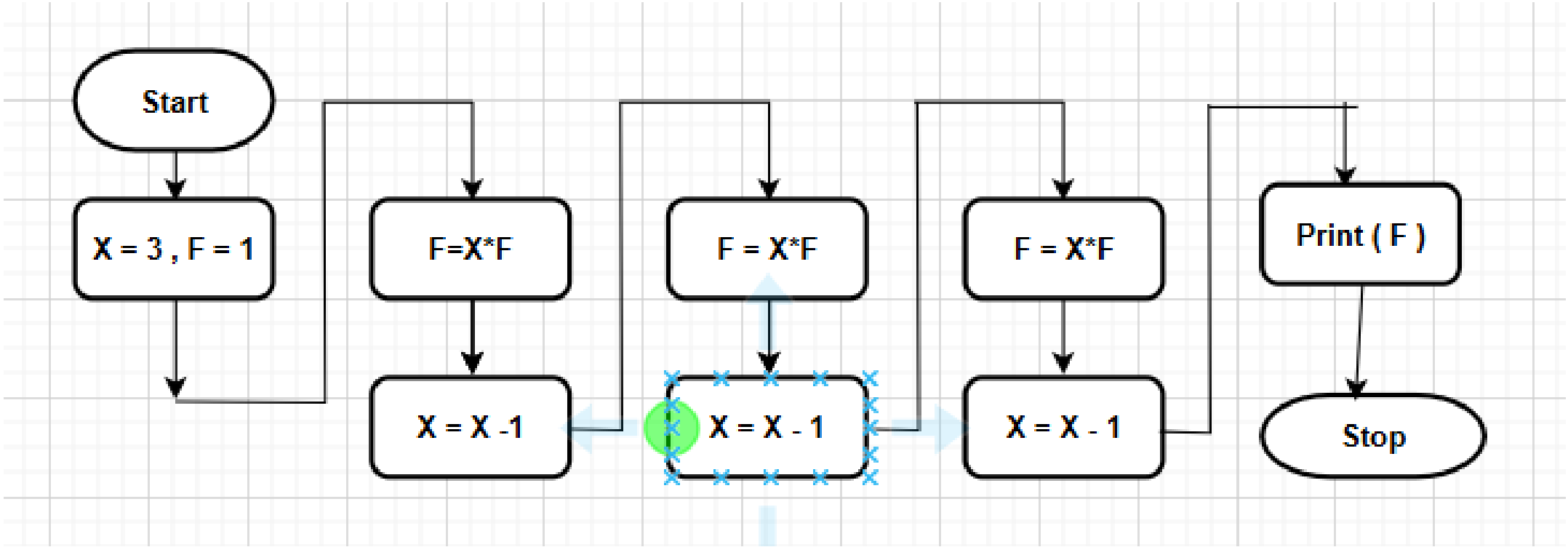
ฝึกเขียน

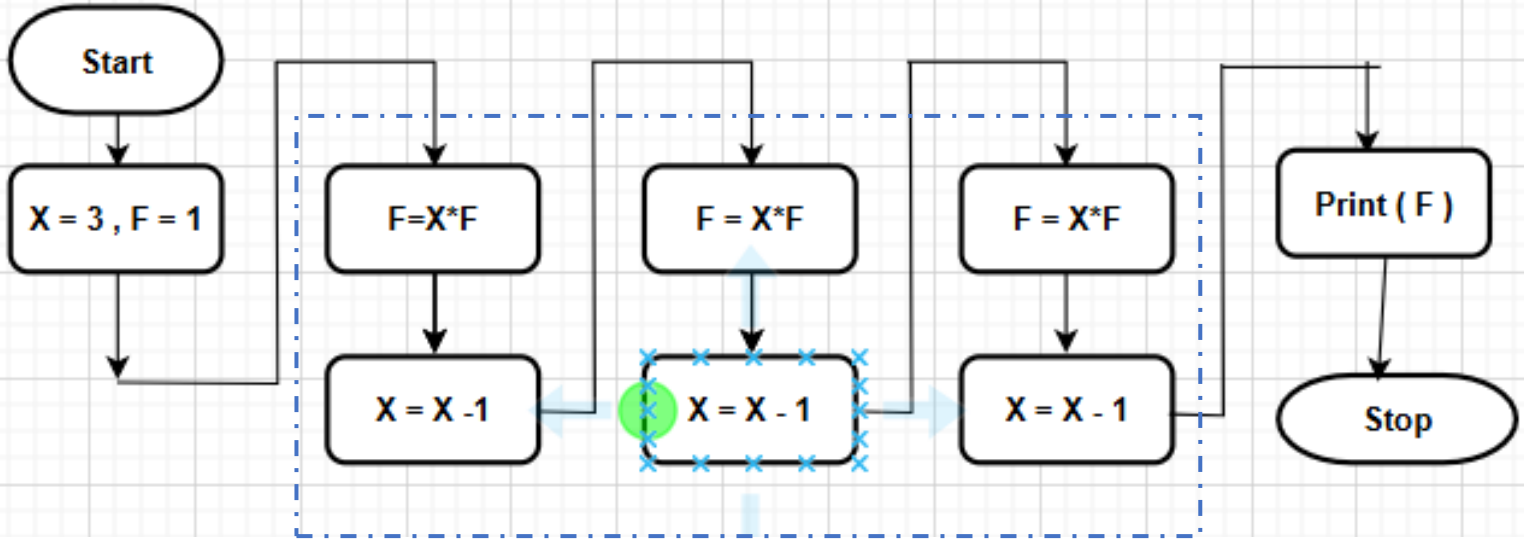

2
read (salary)
read (commision)
tax = salary*(5.0/100.0)
Net = salary- tax + commission
print(Net)
3

+

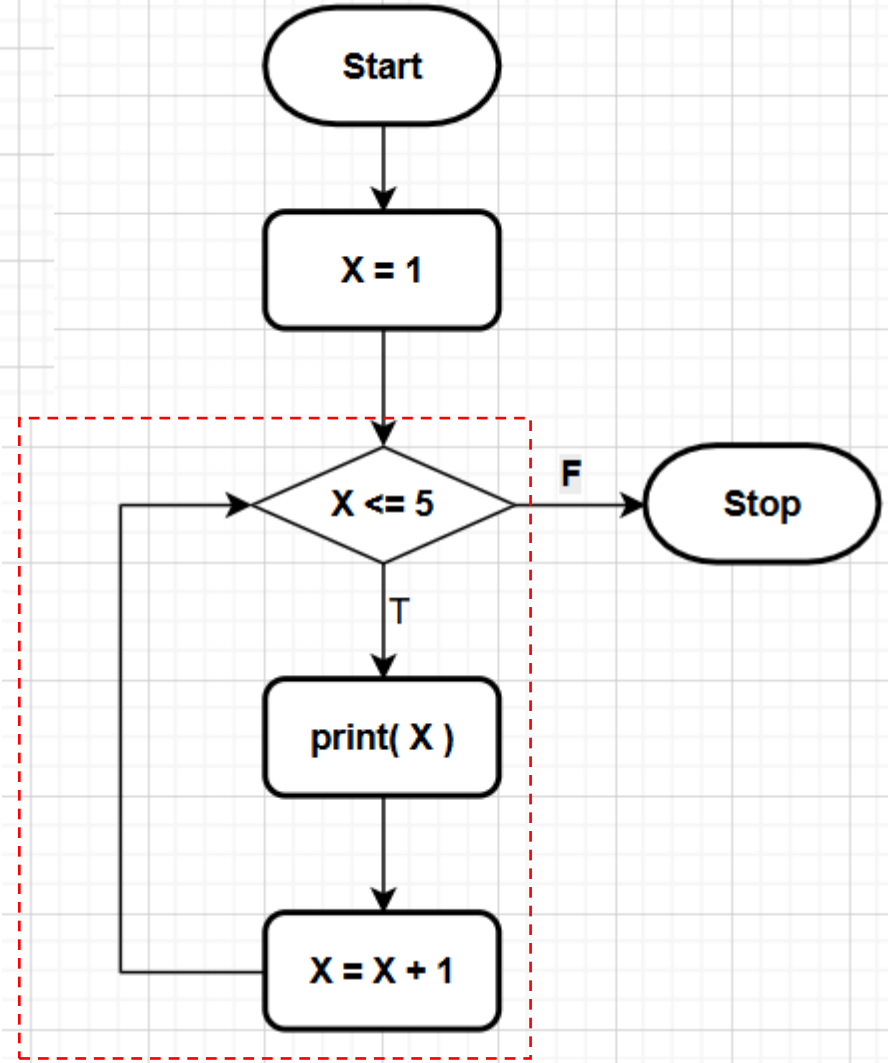

Start
read(Y)
X = 1
X <= Y
F
Stop
T
print(X)
X = X + 1

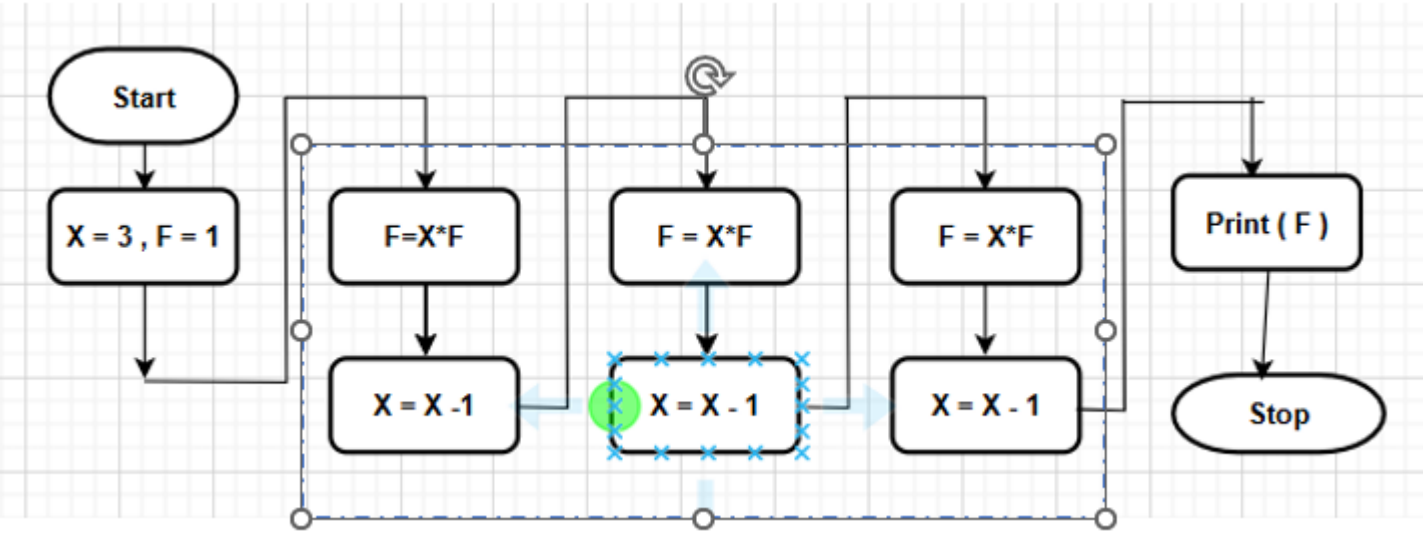
ฝึกอ่าน Sequential Flow => convert to be a repetition flow

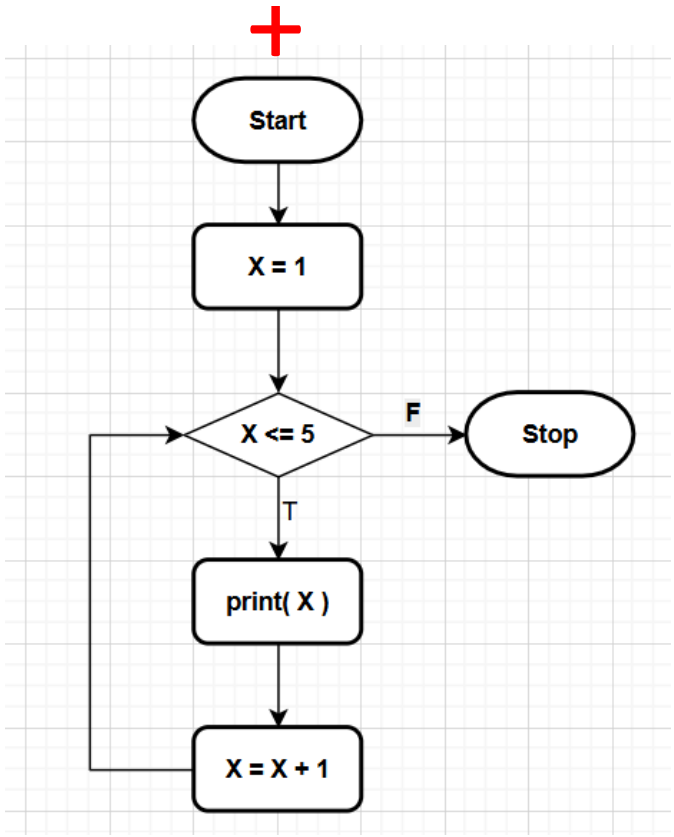


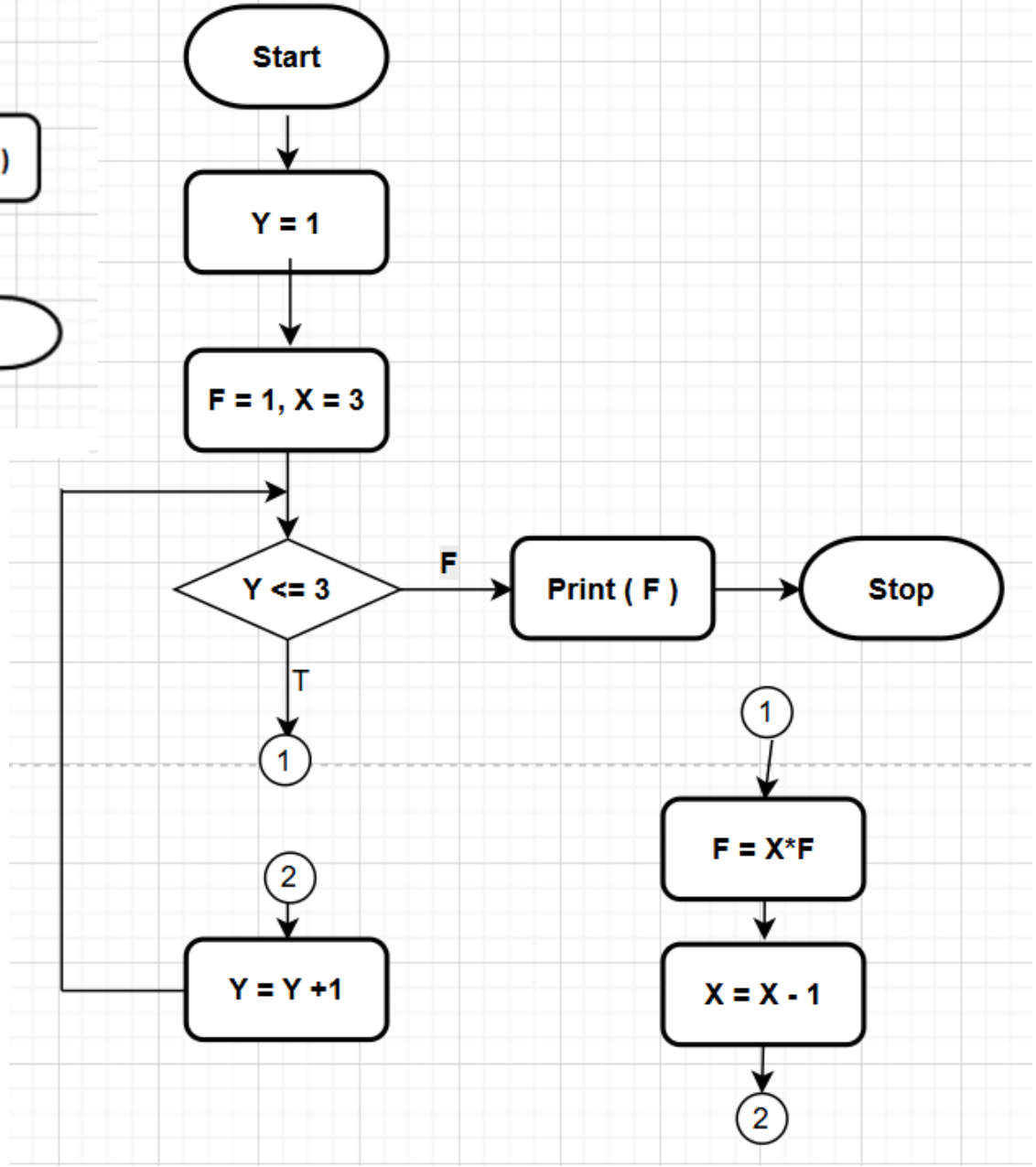

Start
X = 3 , F = 1
F=X*F
X = X -1
F = X*F
X = X - 1
F = X*F
X = X - 1
Print (F)
Stop

+


Start
X = 1
X <= 5
F
Stop
T
print(X)
X = X + 1


Start
X = 3 , F = 1
F=X*F
F = X*F
F = X*F
Print (F)
X = X -1
X = X - 1
X = X - 1
Stop

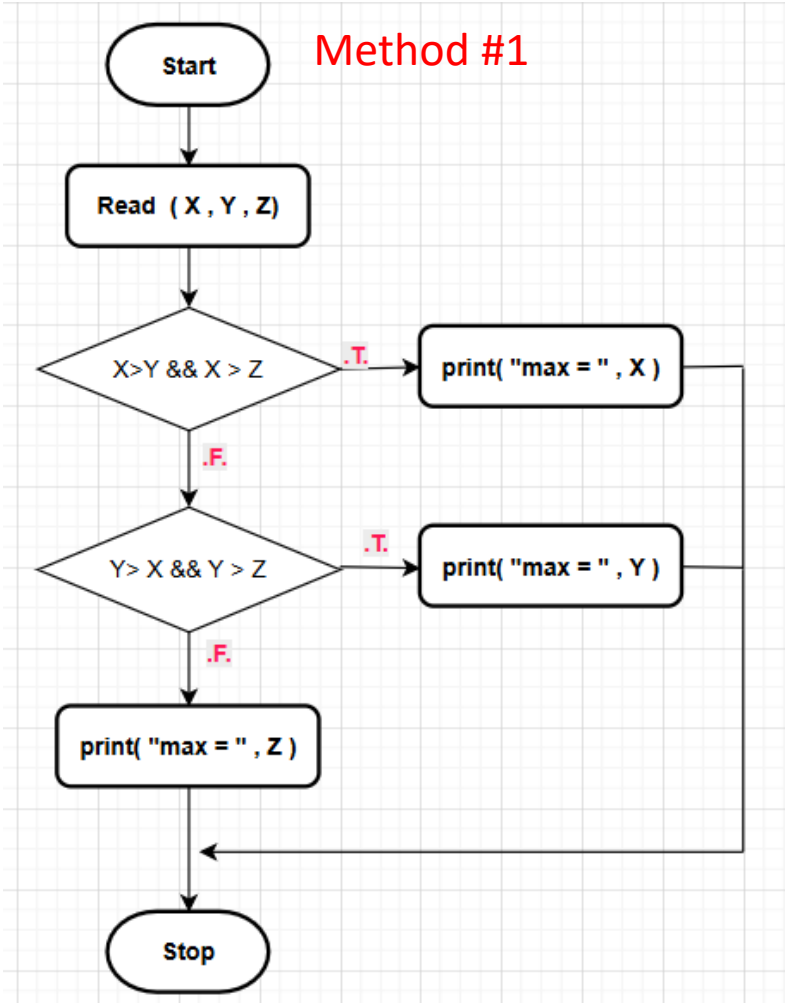

Start
X = 1
X <= 5
F
Stop
T
print(X)
X = X + 1


Start
Y = 1
F = 1, X = 3
Y <= 3
F
Print (F)
Stop
T
1
2
Y = Y +1
1
F = X*F
X = X - 1
2

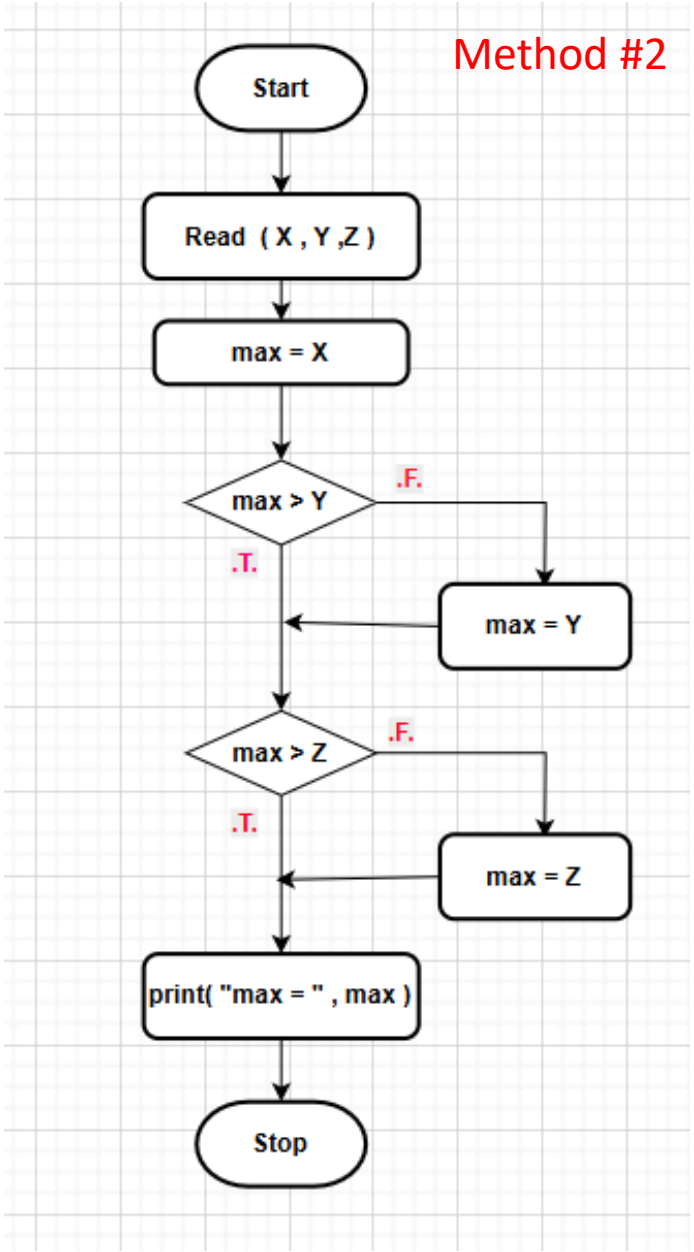
Which one could be converted to repetition flow? :

รับข้อมูล 3 ค่า เพื่อหาค่ามากสุด

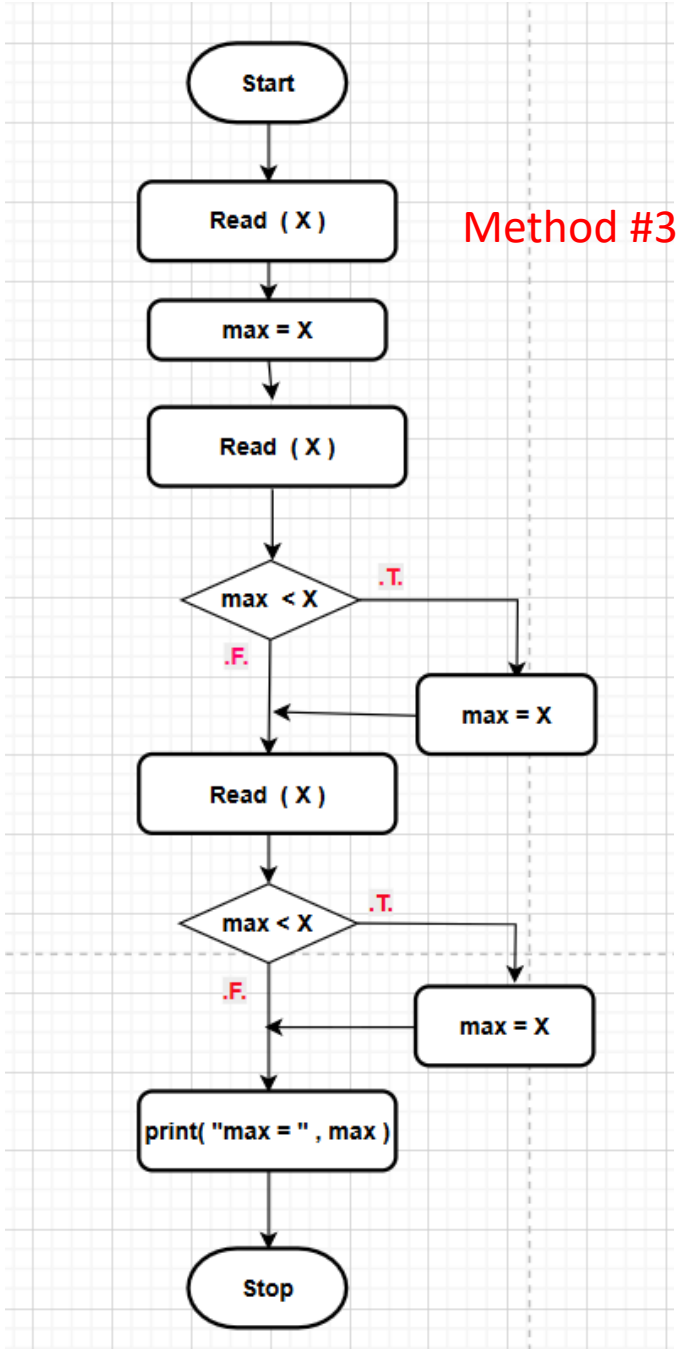
Method #1



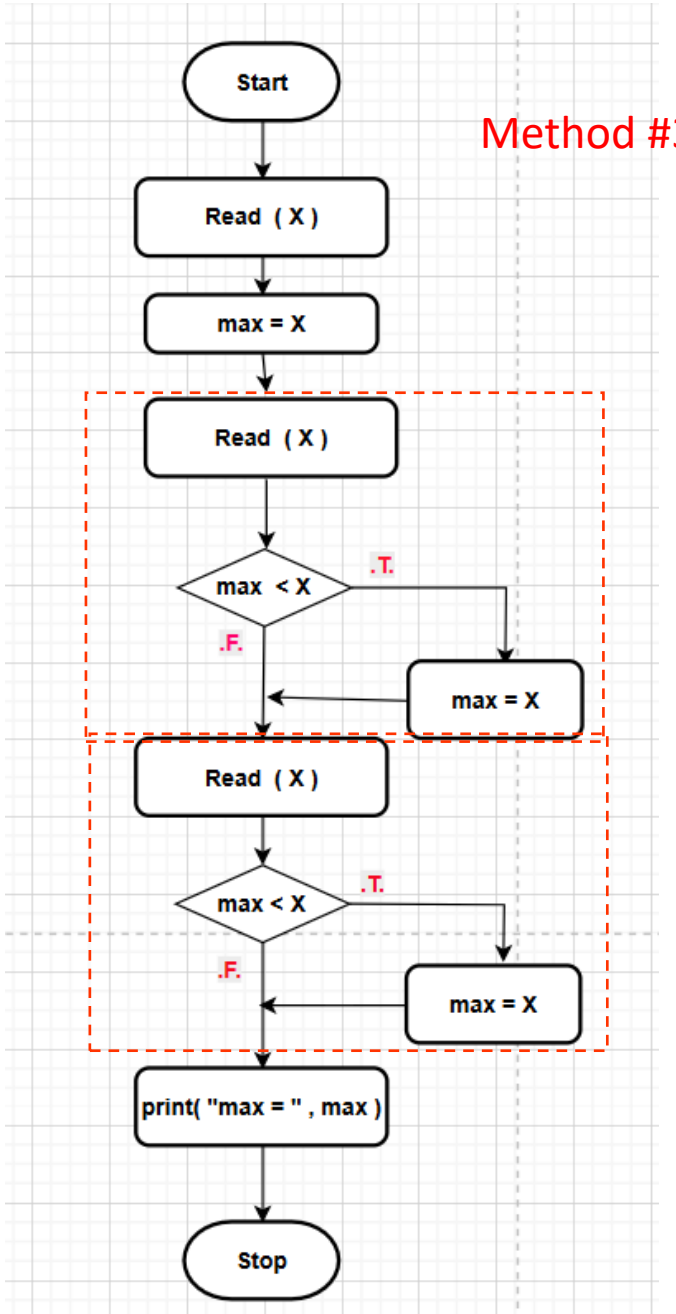
Method #2



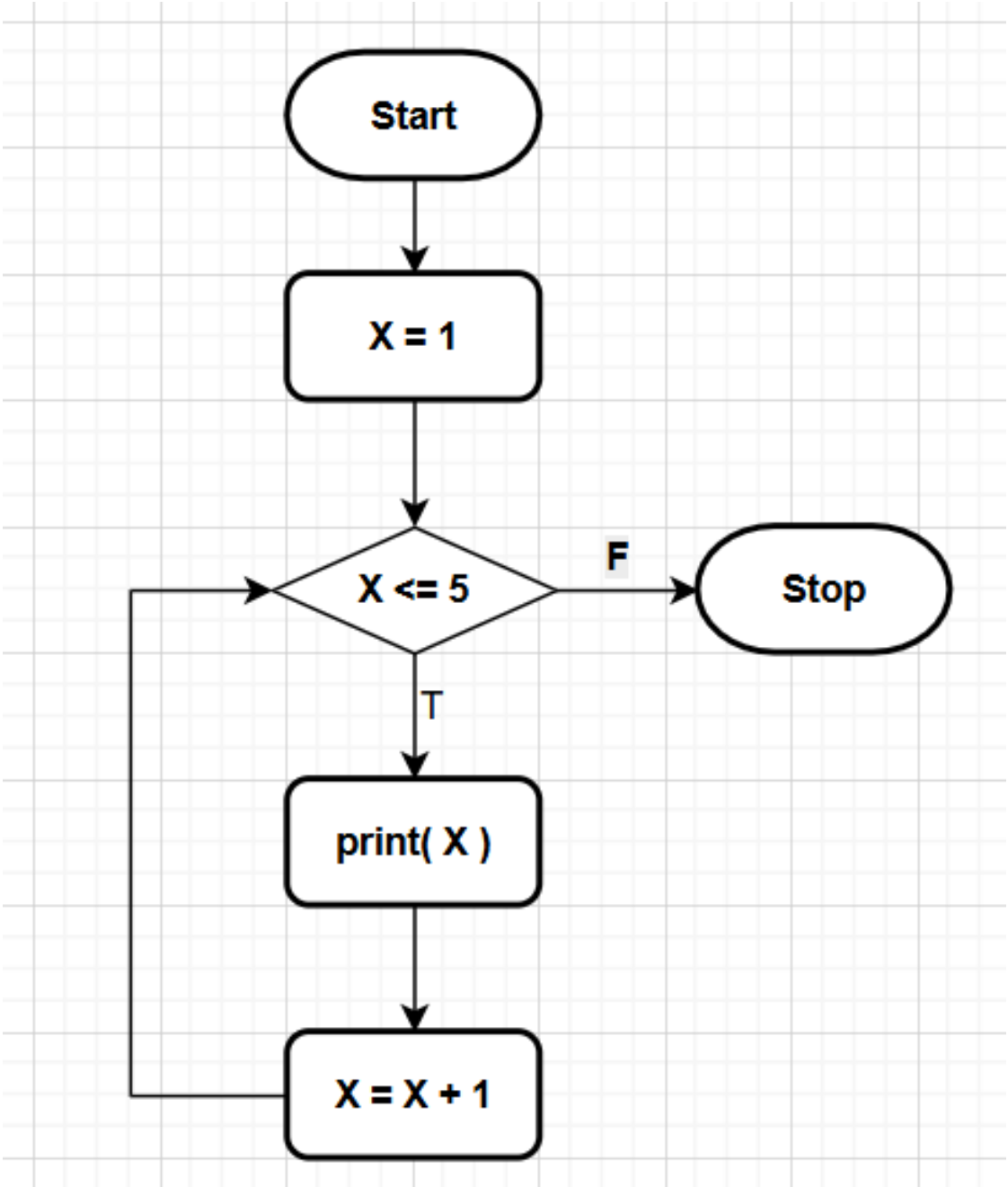
Method #3



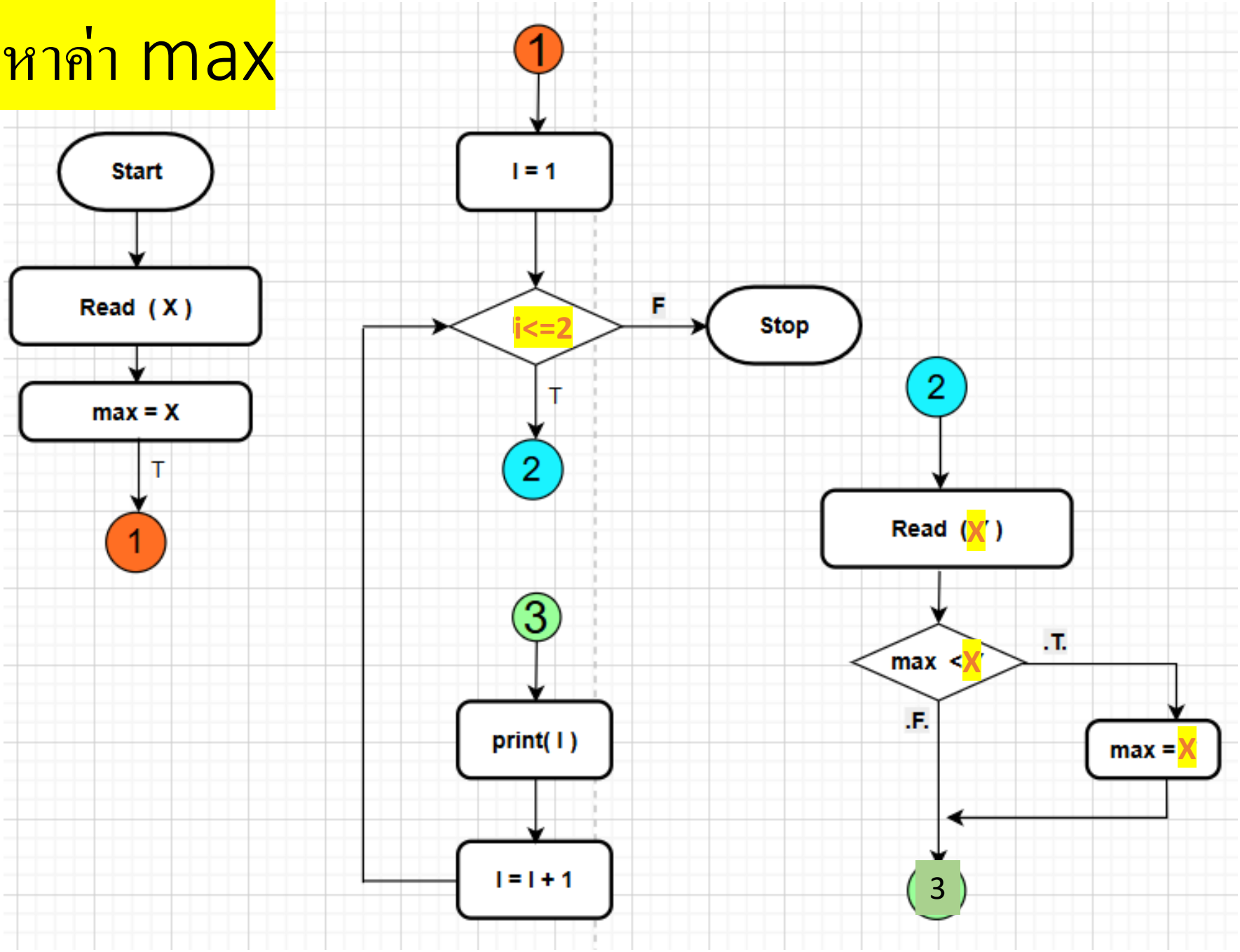
Method #3


Start
Read (X)
max = X
Read (X)
max < X
.T.
.F.
max = X
Read (X)
max < X
.T.
.F.
max = X
print("max = " , max)
Stop

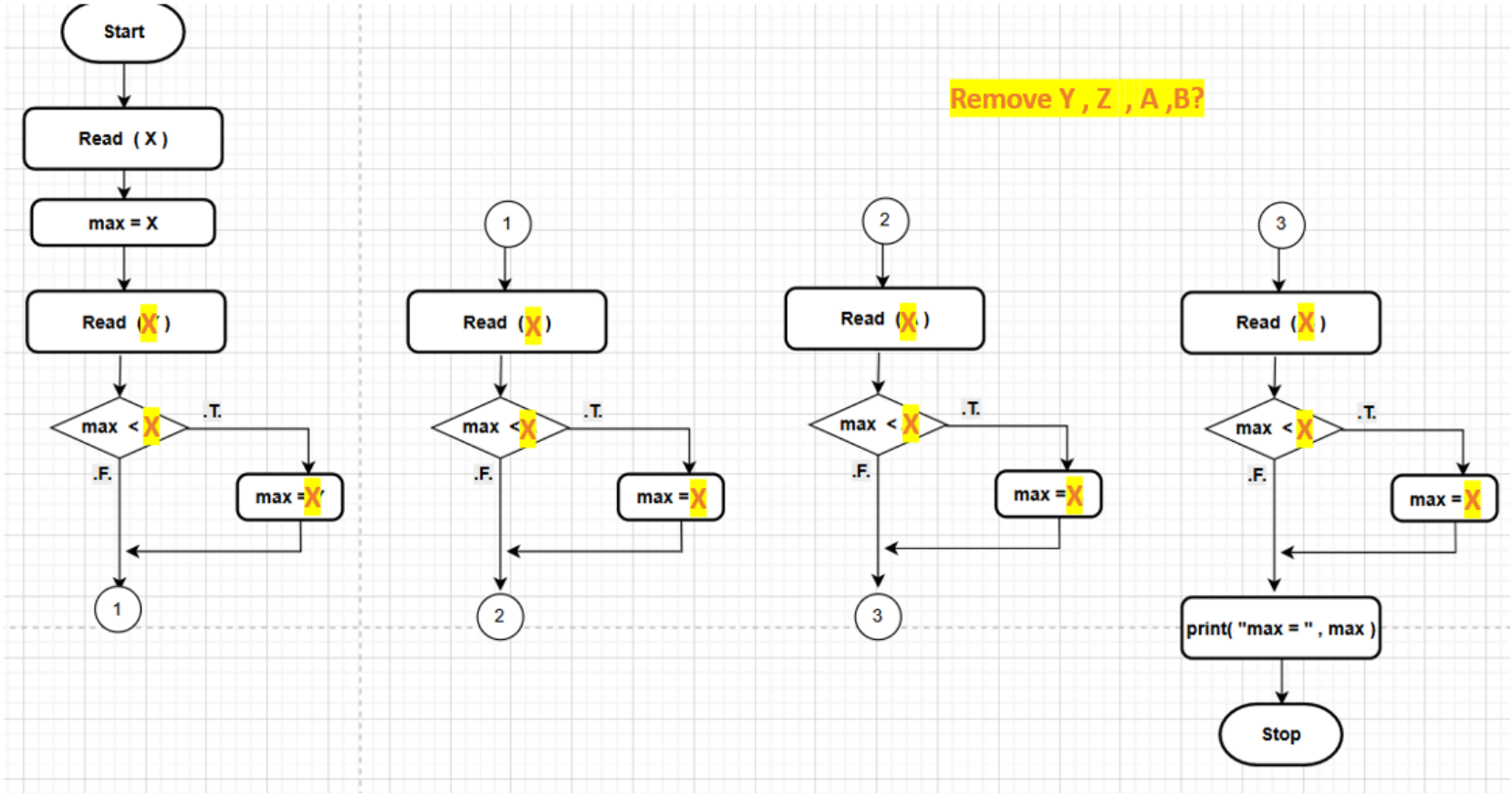
+


Start
X = 1
X <= 5
F
Stop
T
print(X)
X = X + 1

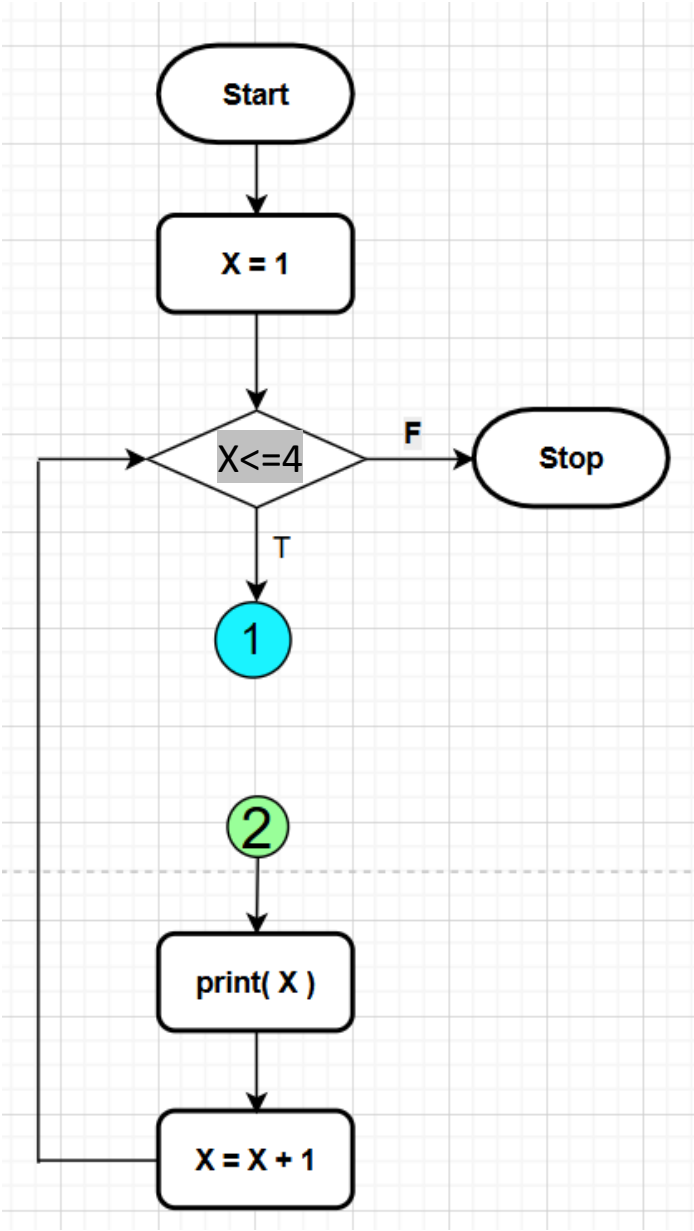
ถ้ารับข้อมูล 2 ตัวและหาค่า max



ถ้ารับข้อมูล 5 ตัวและหาค่า max



+

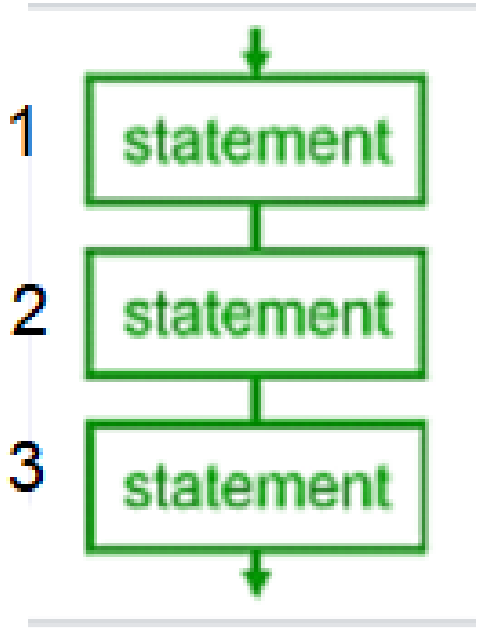


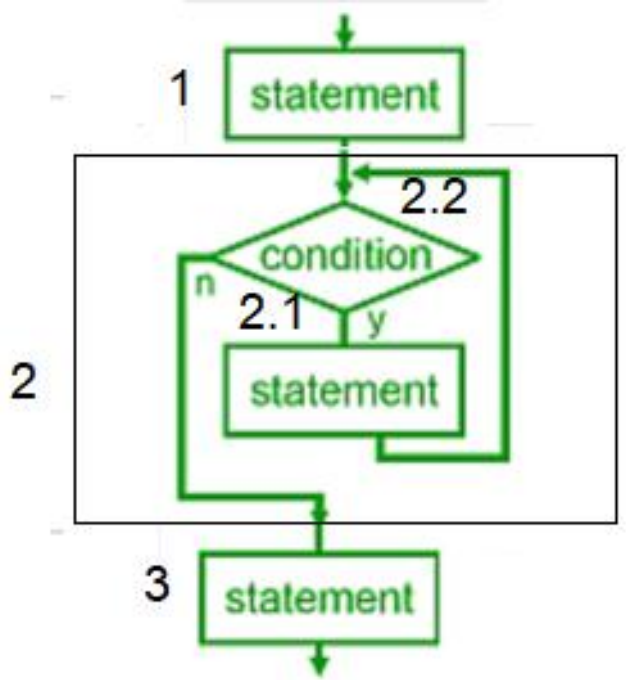
for-Loops

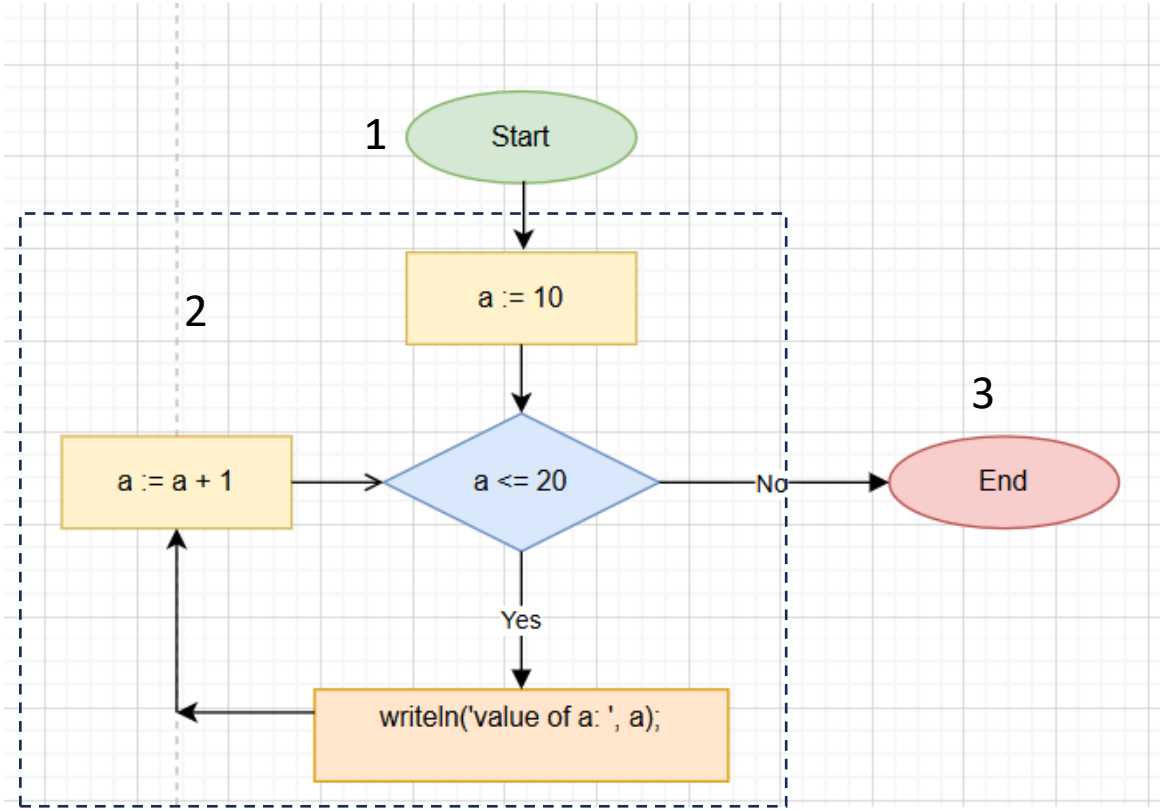
- *The for-loop repeats one or more statements a specified number of time.*

- *if the solution has be known exactly number of repeats time , the for-loop is suitable for using.*

for-Loops

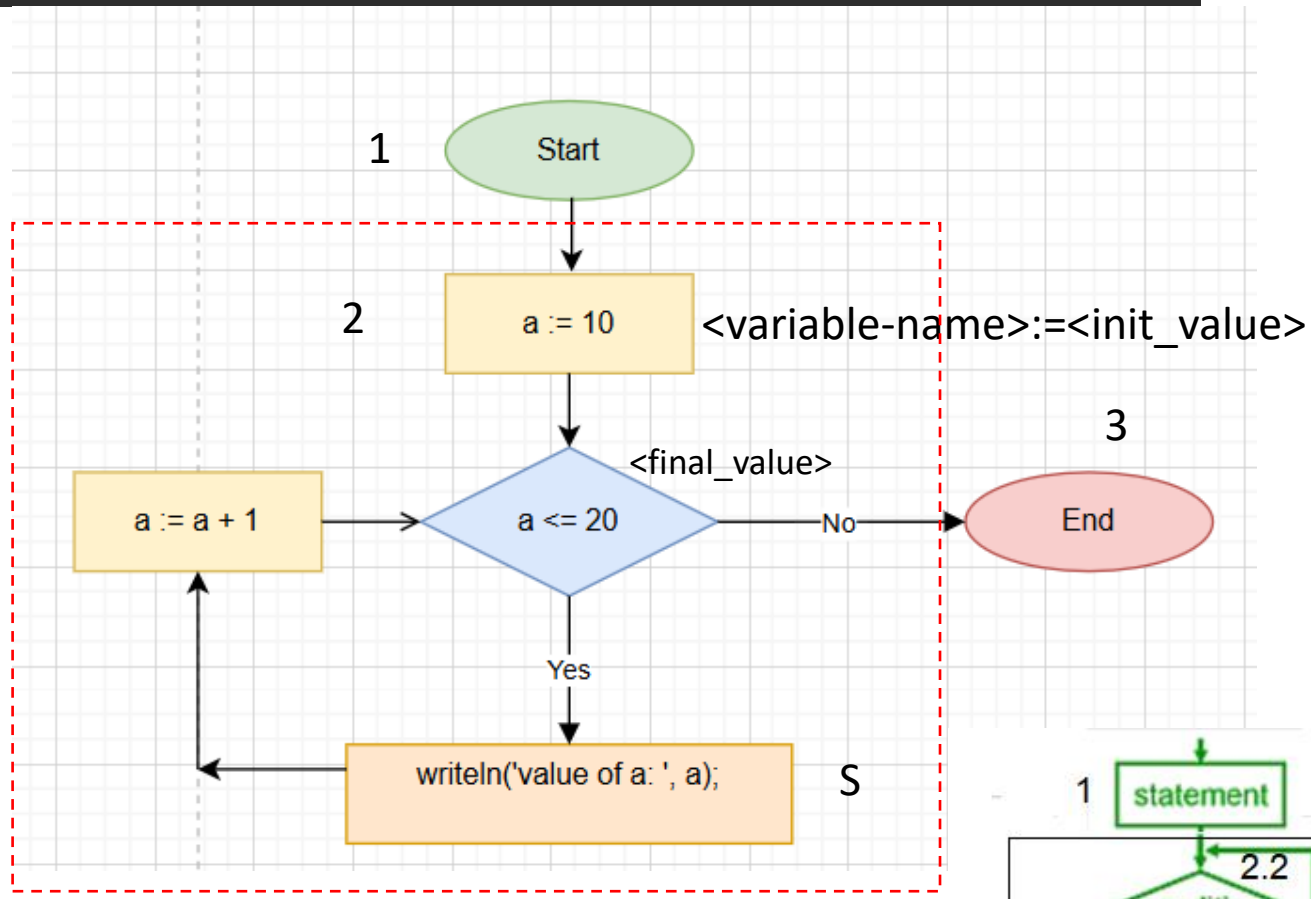

1
statement
2
statement
3
statement

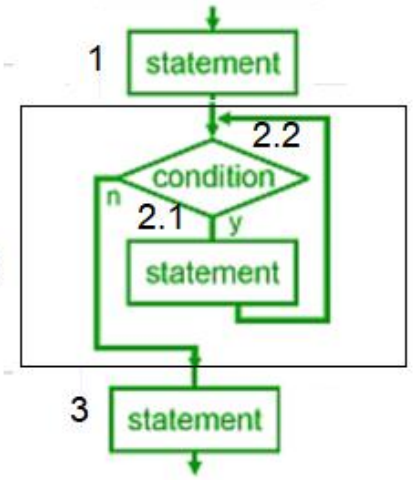

1
statement
2.2
condition
n
2.1
y
statement
2
3
statement


1
Start
2
a := 10
3
a := a + 1
a <= 20
No
End
Yes
writeln('value of a: ', a);

```
for < variable-name > := < initial_value > to [down to] < final_value > do
   S;
```

```
program forLoop;
var
   a: integer;

begin
   for a := 10  to 20 do

   begin
      writeln('value of a: ', a);
   end;
end.
```



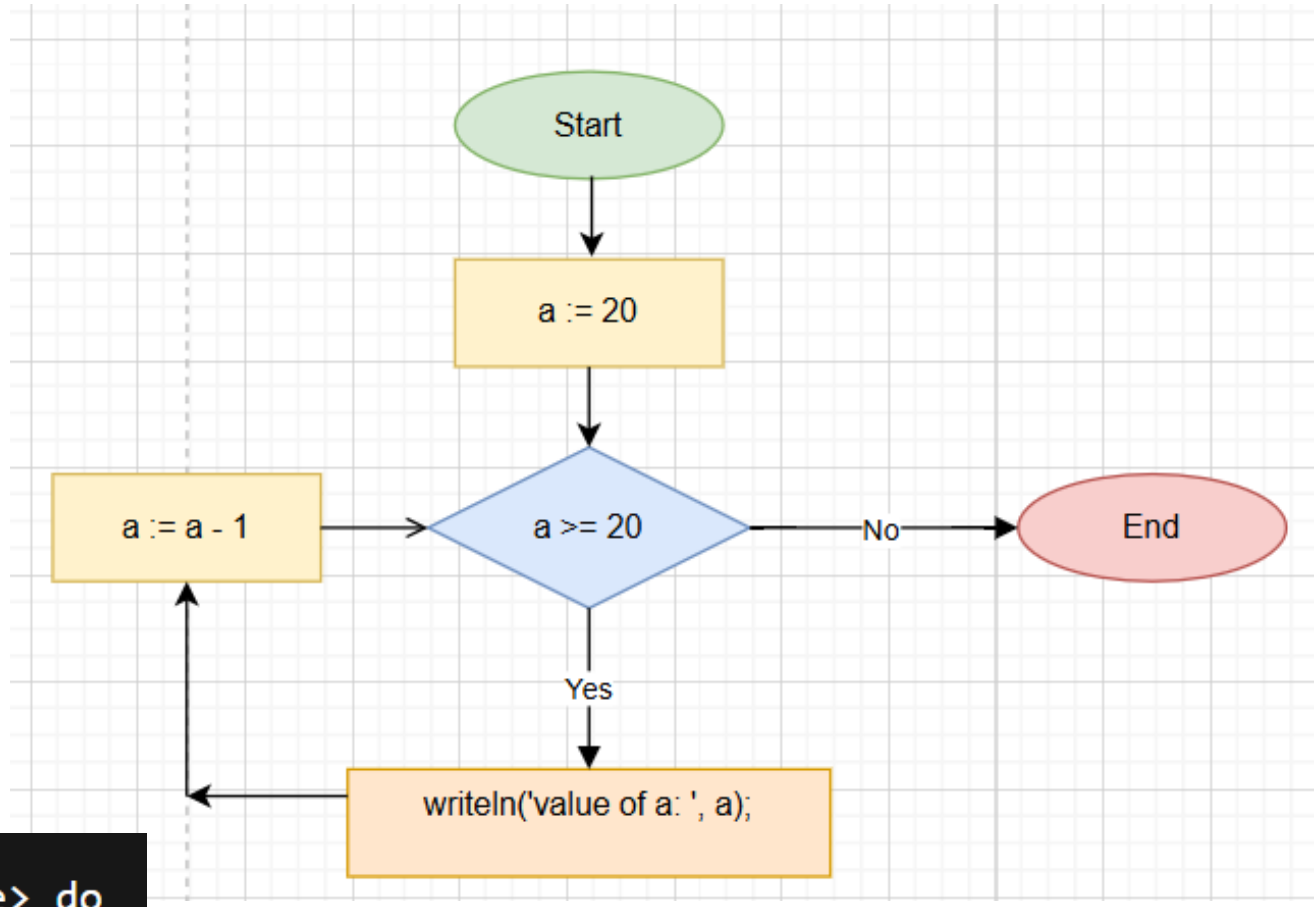


หน้าที่ของ **<variable-name>** คือ นับจำนวนการทำซ้ำ

```
for < variable-name > := < initial_value > to [down to] < final_value > do
    S;
```

forloop.pas

```pascal
program forLoop;
var
   a: integer;

begin
   for a := 20  downto 10 do

   begin
      writeln('value of a: ', a);
   end;
end.
```




```
for <variable> := <start_value> downto <end_value> do
    <statement>;
```

```pascal
program TestFor;
var
  x: integer;
begin
  for x := 1 to 4 do
    WriteLn('inside: x = ', x);
  WriteLn('after: x = ', x);  { ดูค่าว่าคอมไพเลอร์ของคุณแสดงอะไร }
end.
```

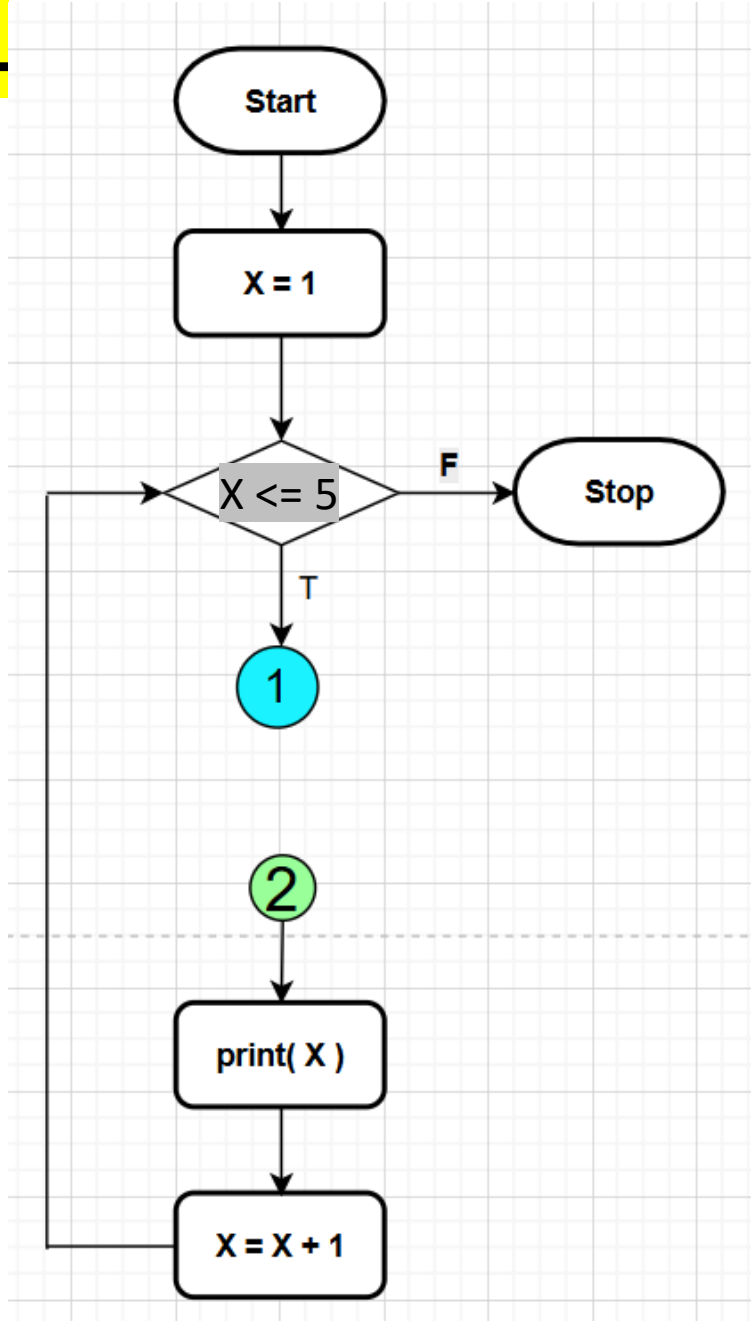
TestFor.pas

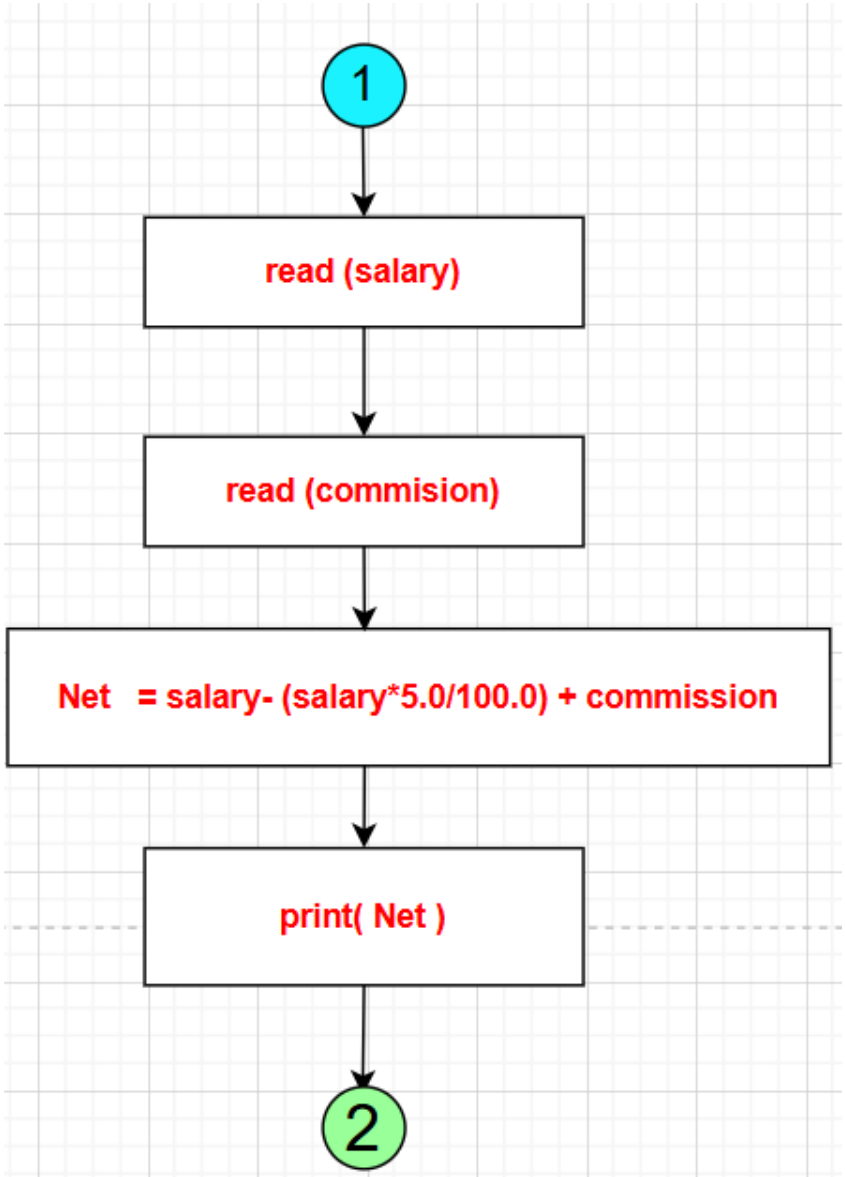
จุดที่ต้องระวัง

Output:

```
inside: x = 1
inside: x = 2
inside: x = 3
inside: x = 4
after: x = 4
```

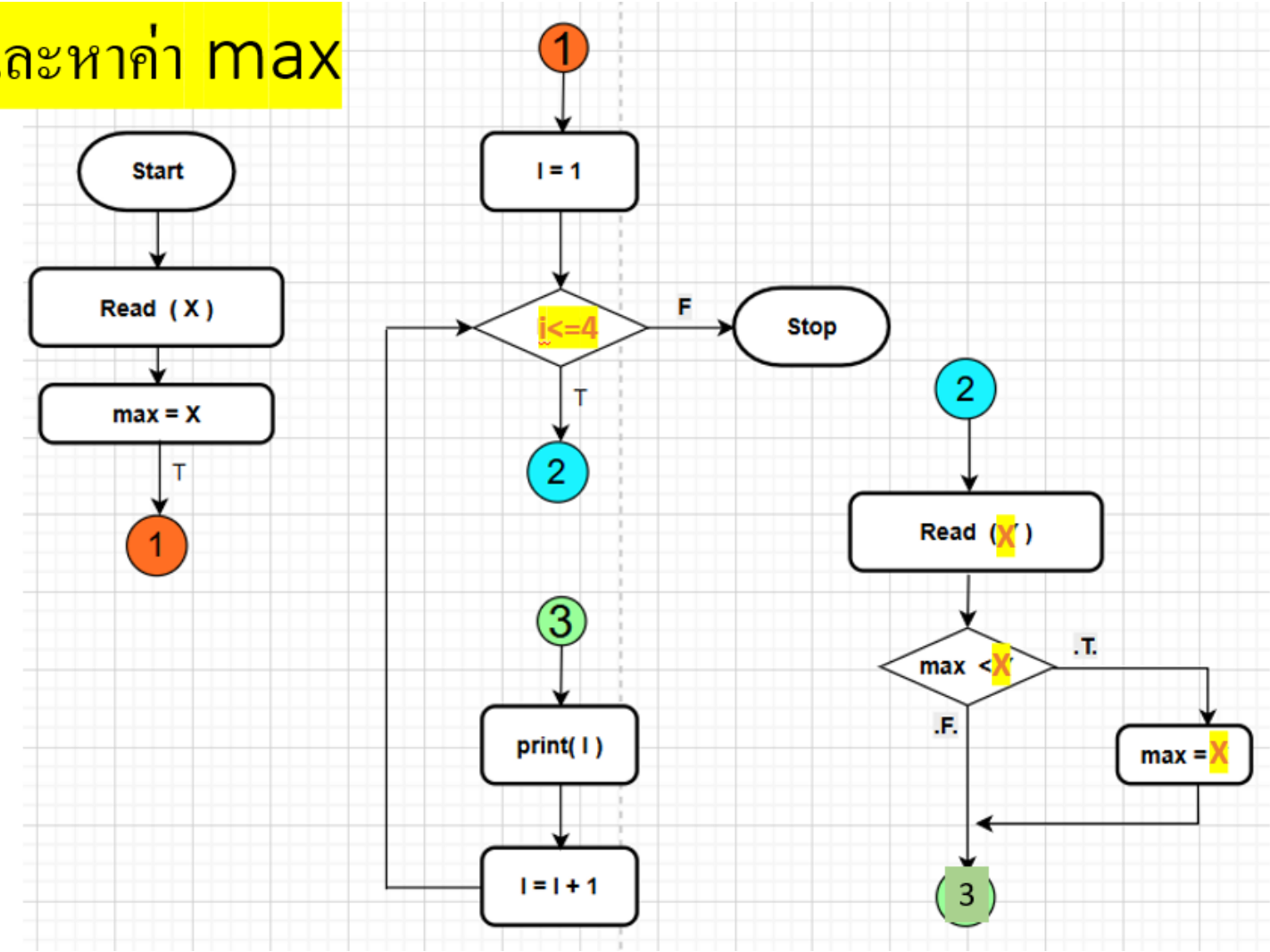
Convert to PASCAL



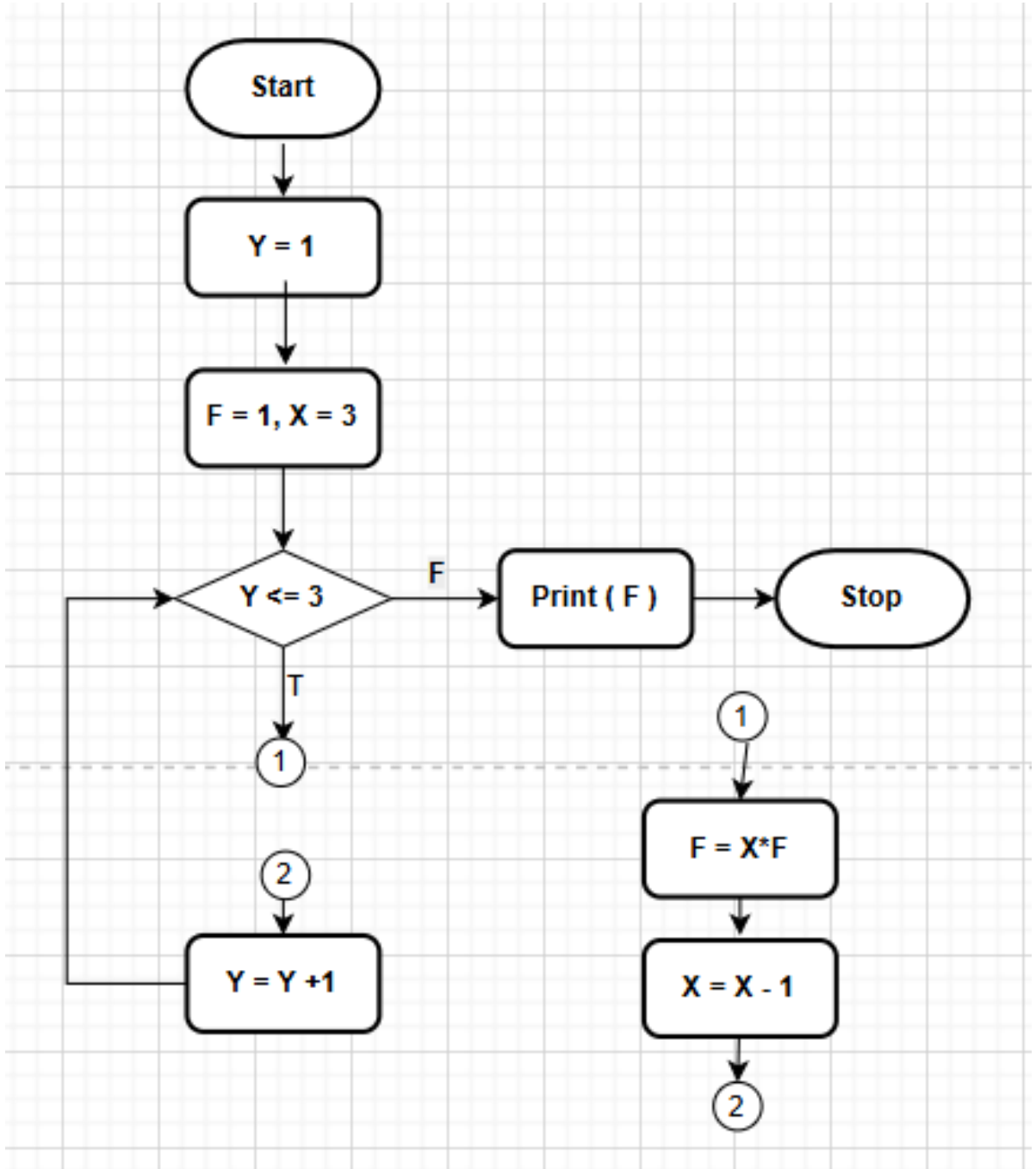


Convert to PASCAL

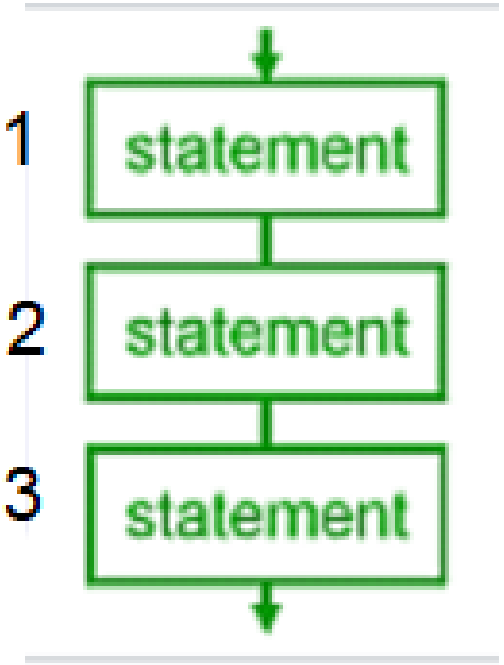
ถ้ารับข้อมูล 5 ตัวและหาค่า max

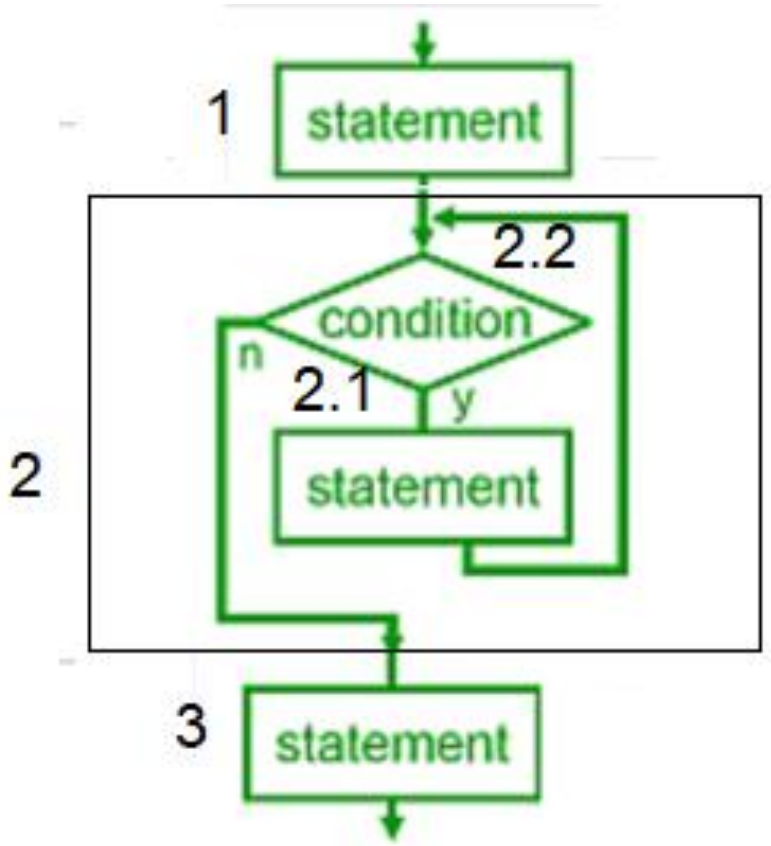


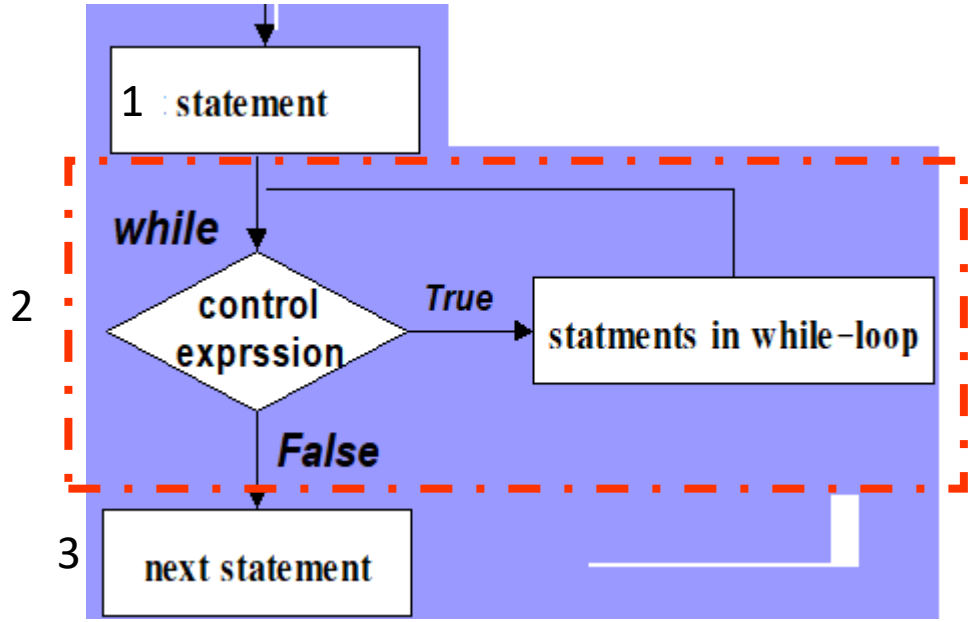
Convert to PASCAL



While – Stm : unknown repetitions

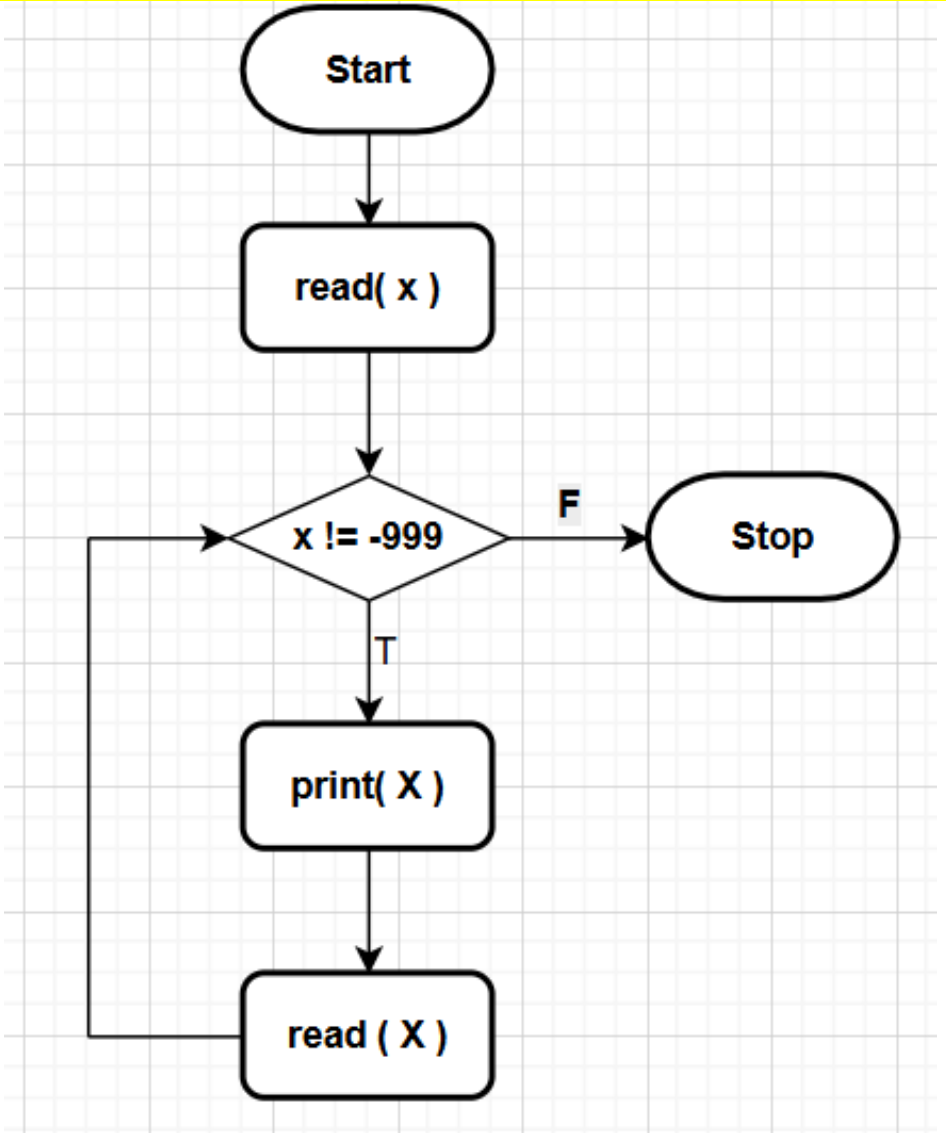






Loop / Repetition flow : Unknown number of repetition

- flow นี้อ่านว่าอะไร ?



เขียนโปรแกรมรับข้อมูลของพนักงานเพื่อคำนวณเงินเดือนสุทธิของพนักงาน โดยมีละเอียดข้อมูลดังนี้

- รับเงินเดือนพื้นฐาน **(salary)**
- หักภาษี 5% **(salary* 5.0/100.0)**
- รับค่าคอมมิชชั่นเพิ่มเติม **(commission)**
- แสดงเงินเดือนสุทธิ = (เงินเดือน – ภาษี) + ค่าคอมมิชชั่น

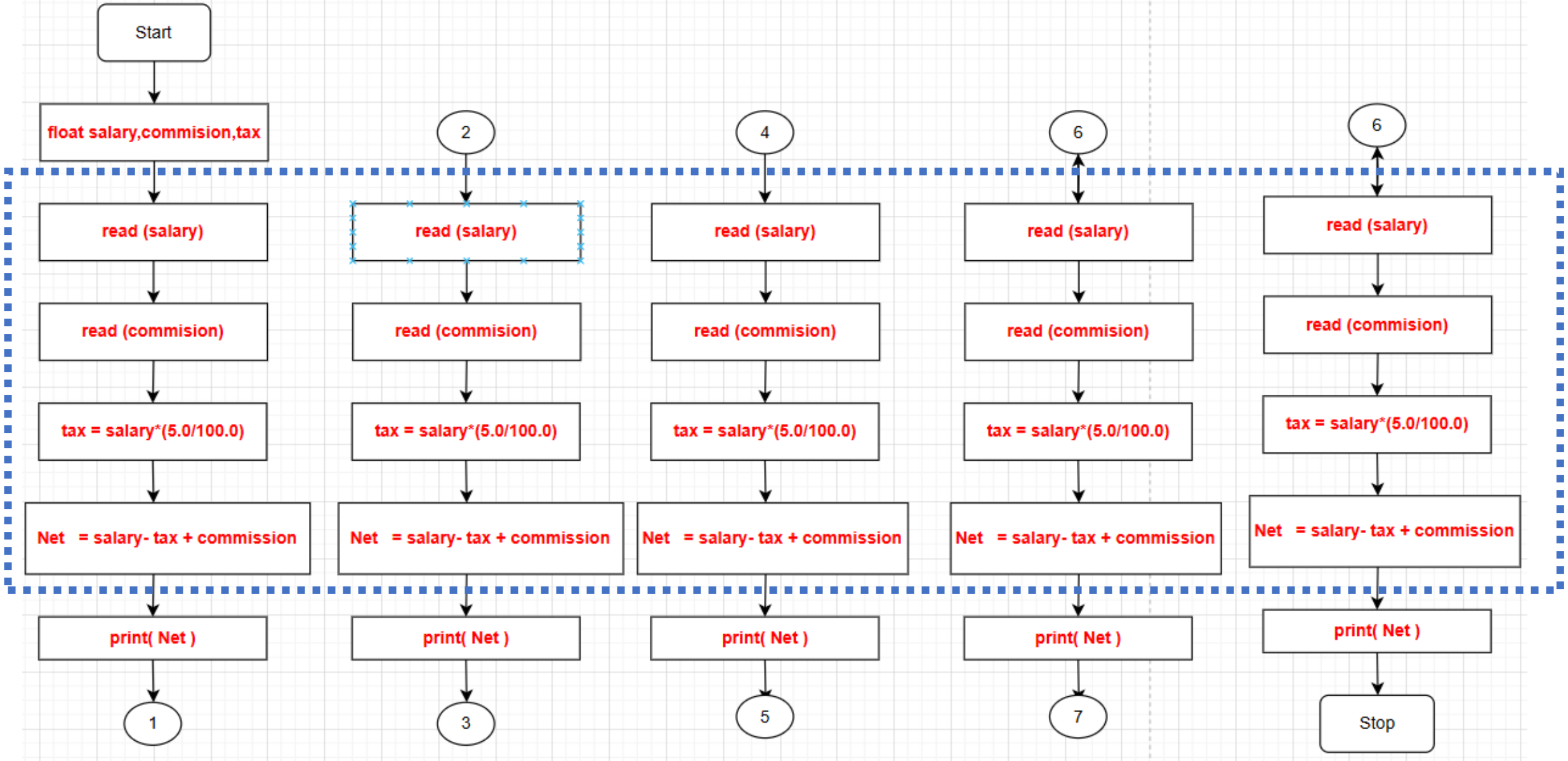
Net = salary- (salary* 5.0/100.0) + commission

โดยรับข้อมูลจนกระทั้งค่าเงินเดือนพื้นฐาน **(salary)** มีค่าน้อยกว่า **0**

เขียนโปรแกรมรับข้อมูลของพนักงานจำนวน **5** คน เพื่อคำนวณเงินเดือนสุทธิของพนักงาน โดยมีละเอียดข้อมูลดังนี้

- รับเงินเดือนพื้นฐาน **(salary)**
- หักภาษี 5% **(salary* 5.0/100.0)**
- รับค่าคอมมิชชั่นเพิ่มเติม **(commission)**
- แสดงเงินเดือนสุทธิ = (เงินเดือน – ภาษี) + ค่าคอมมิชชั่น

Net = salary- (salary* 5.0/100.0) + commission

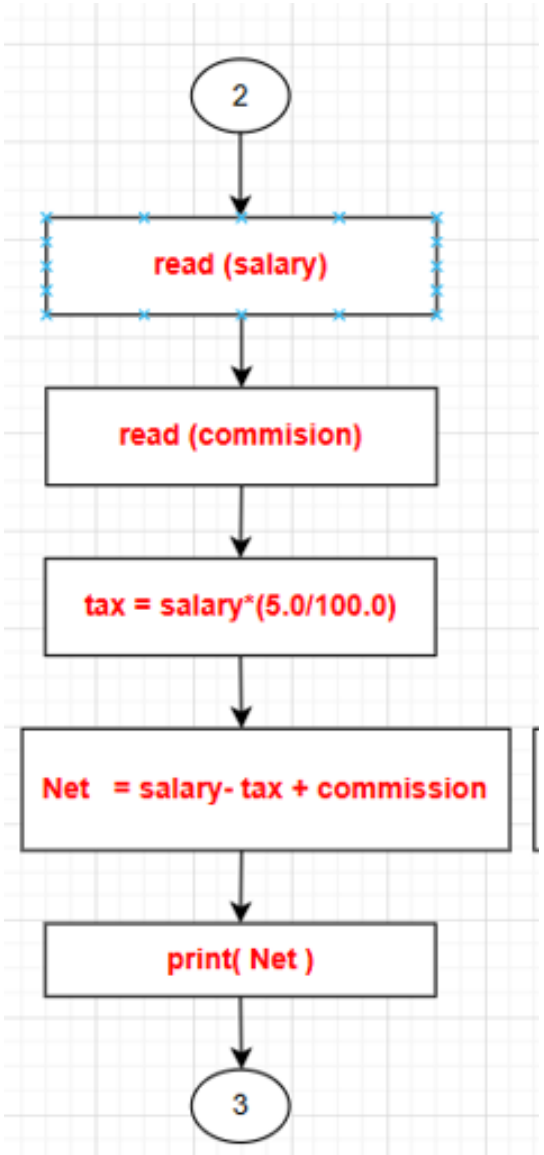

Start
float salary,commision,tax
read (salary)
read (commision)
tax = salary*(5.0/100.0)
Net = salary- tax + commission
print(Net)
1
2
read (salary)
read (commision)
tax = salary*(5.0/100.0)
Net = salary- tax + commission
print(Net)
3
4
read (salary)
read (commision)
tax = salary*(5.0/100.0)
Net = salary- tax + commission
print(Net)
5
6
read (salary)
read (commision)
tax = salary*(5.0/100.0)
Net = salary- tax + commission
print(Net)
7
6
read (salary)
read (commision)
tax = salary*(5.0/100.0)
Net = salary- tax + commission
print(Net)
Stop

เขียนโปรแกรมรับข้อมูลของพนักงานเพื่อคำนวณเงินเดือนสุทธิของพนักงาน โดยมีละเอียดข้อมูลดังนี้

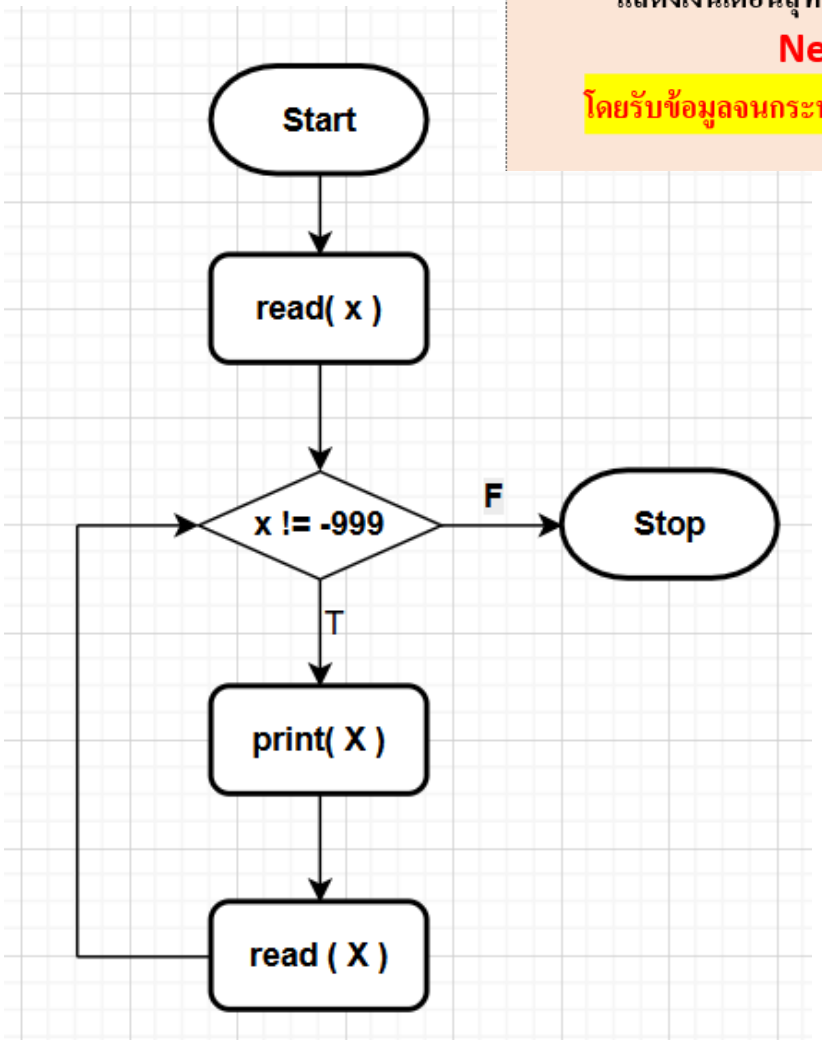
- รับเงินเดือนพื้นฐาน (salary)
- หักภาษี 5% (salary* 5.0/100.0)
- รับค่าคอมมิชชั่นเพิ่มเติม (commission)
- แสดงเงินเดือนสุทธิ = (เงินเดือน – ภาษี) + ค่าคอมมิชชั่น

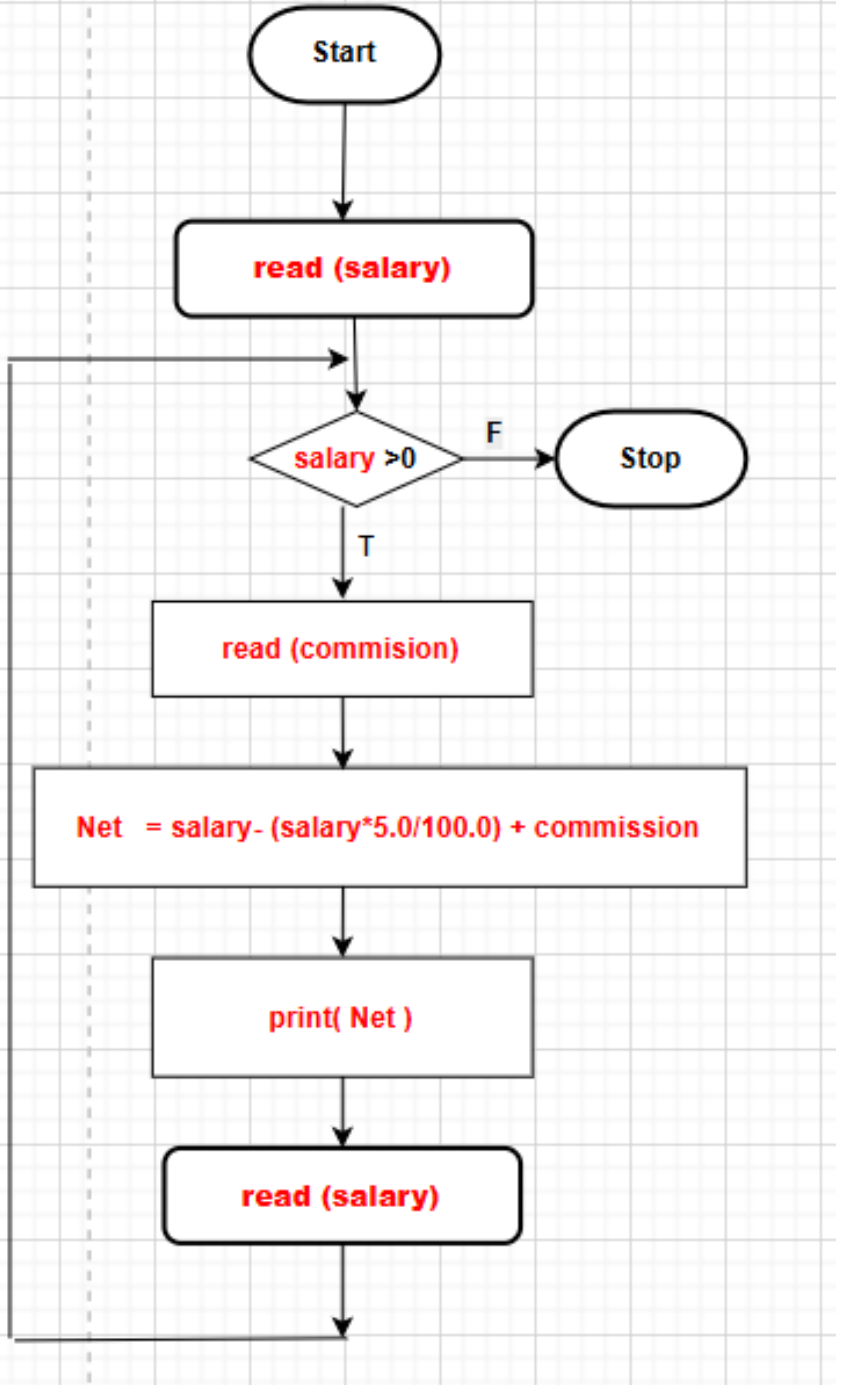
Net = salary- (salary* 5.0/100.0) + commission

โดยรับข้อมูลจนกระทั่งค่าเงินเดือนพื้นฐาน (salary) มีค่าน้อยกว่า 0

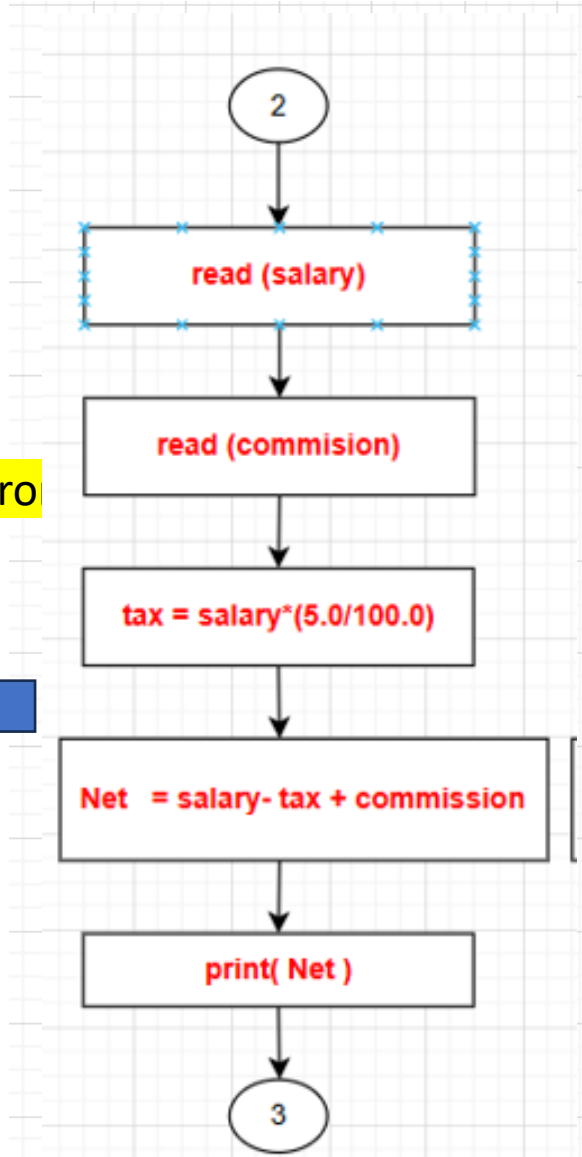


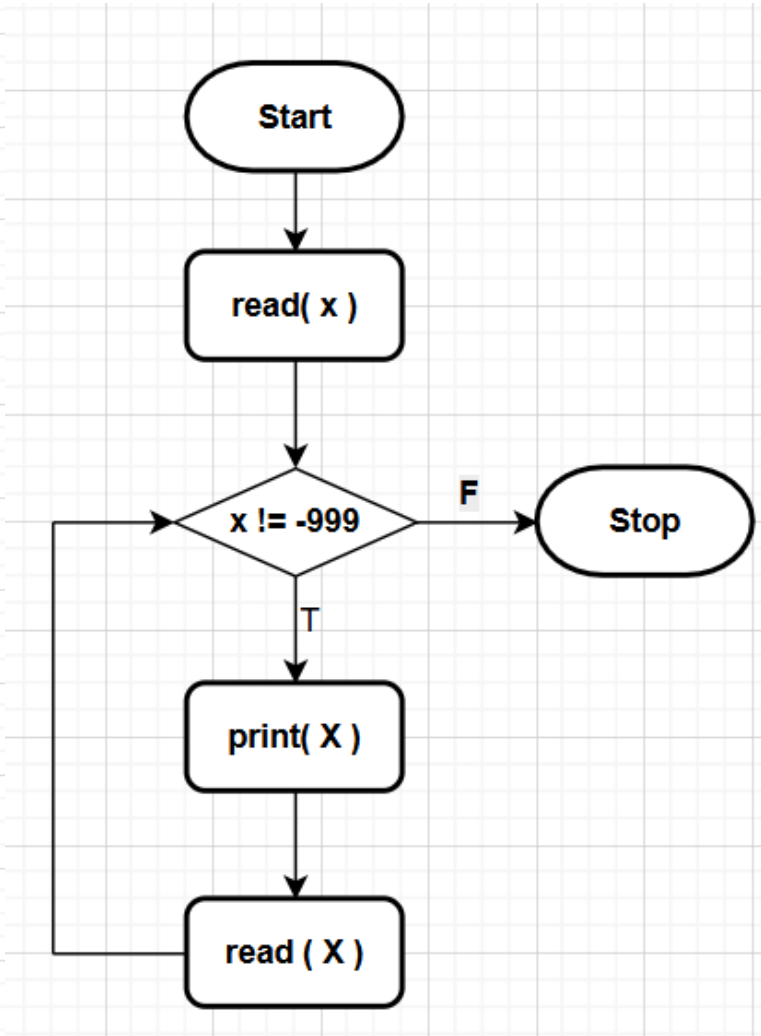
+




Start
read (salary)
salary >0
F
Stop
T
read (commision)
Net = salary- (salary*5.0/100.0) + commission
print(Net)
read (salary)

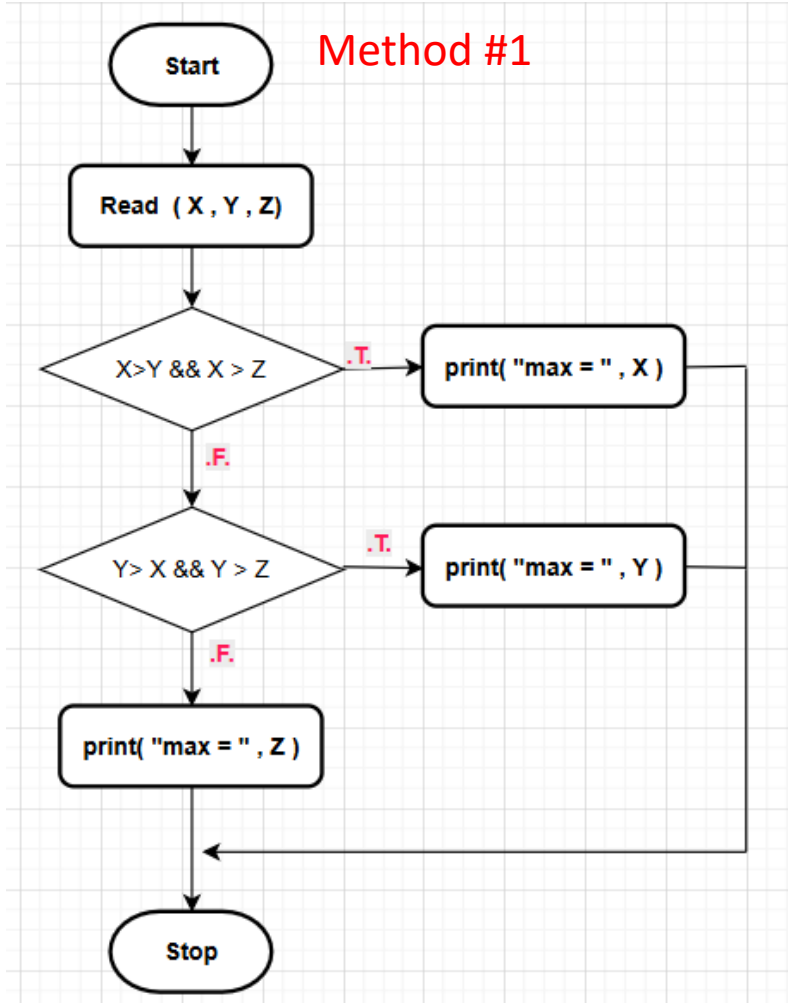
Convert fro

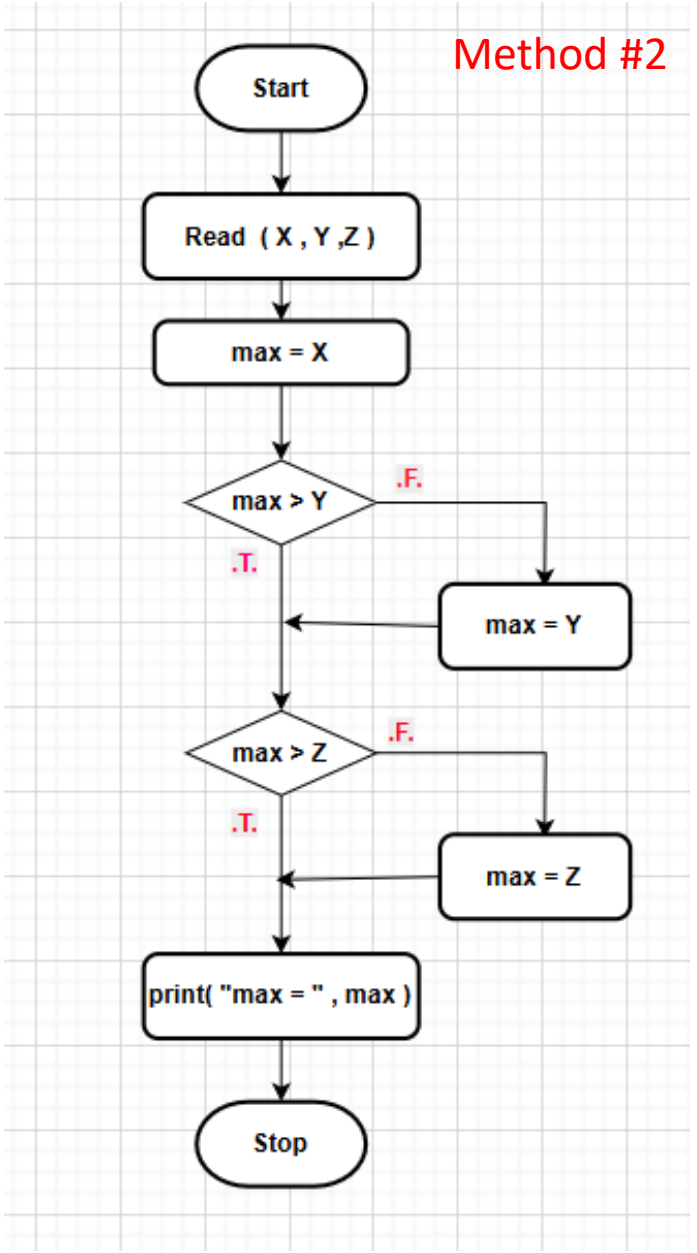

2
read (salary)
read (commision)
tax = salary*(5.0/100.0)
Net = salary- tax + commission
print(Net)
3

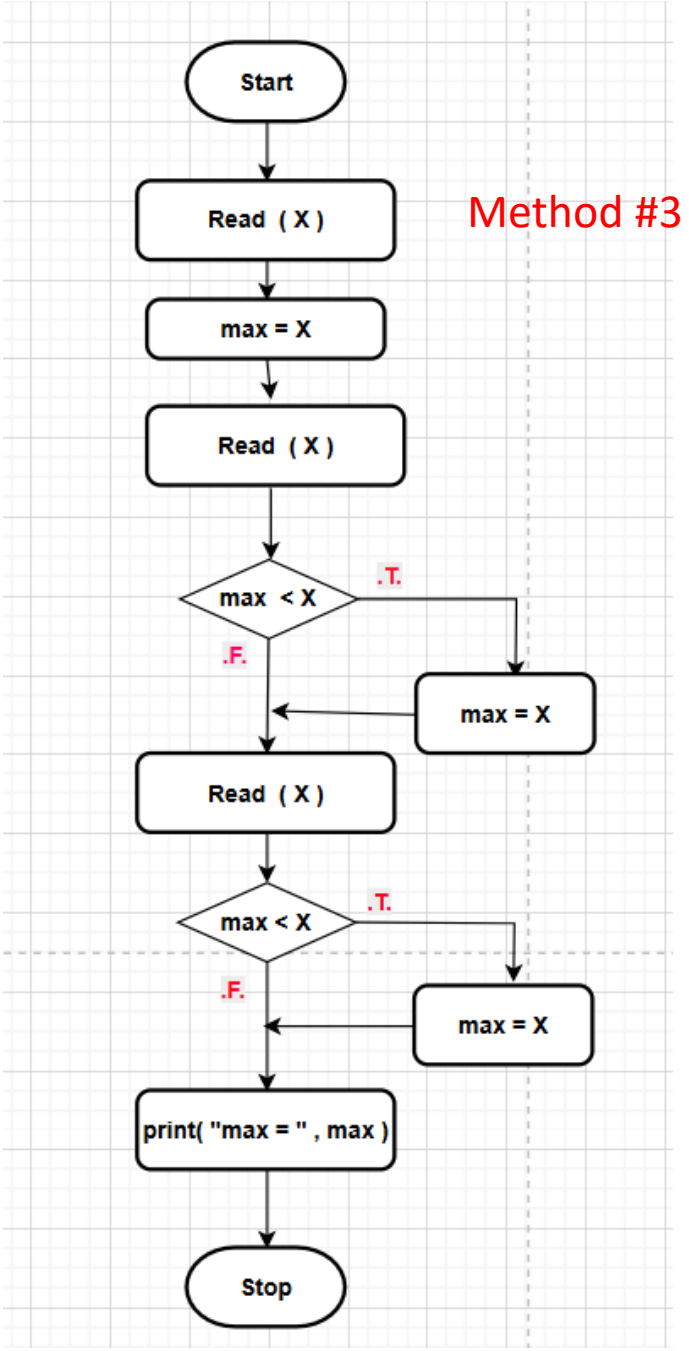

Start
read(x)
x != -999
F
Stop
T
print(X)
read (X)

จงแปลงให้เป็นรับข้อมูลจน X=-999 (หาค่า max)

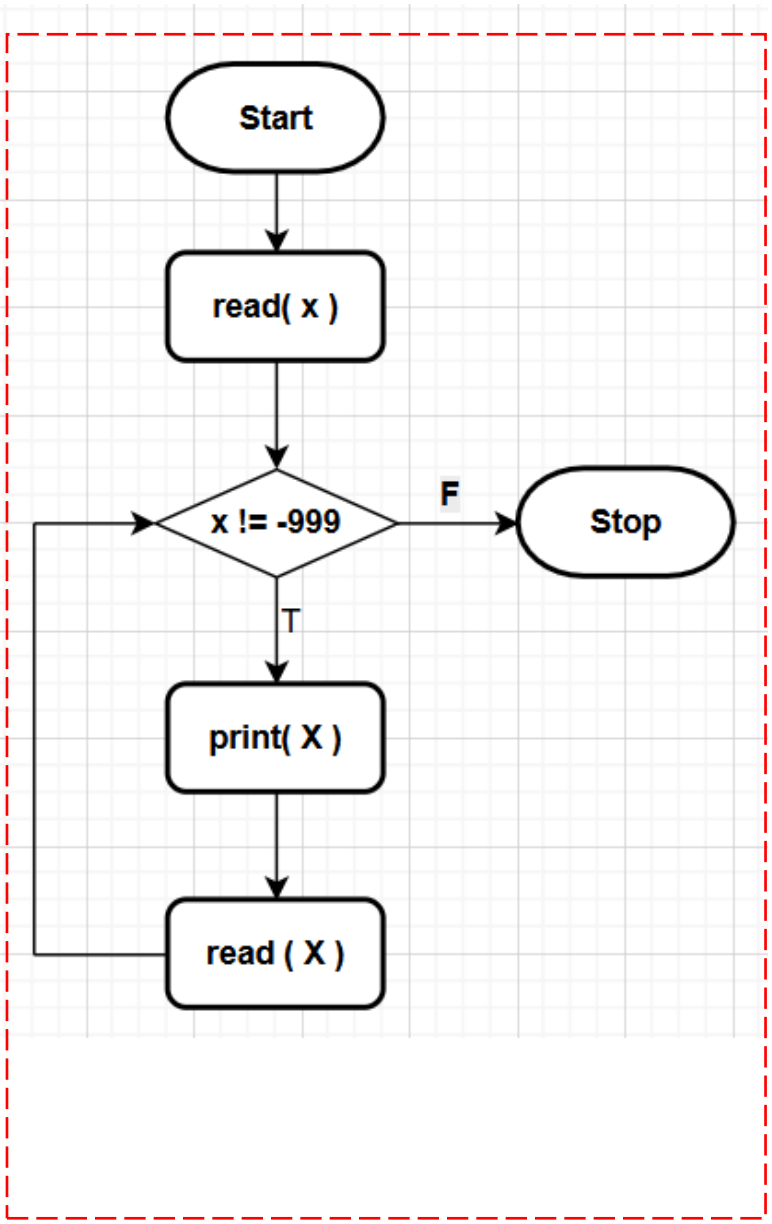
รับข้อมูล 3 ค่า เพื่อหาค่ามากสุด



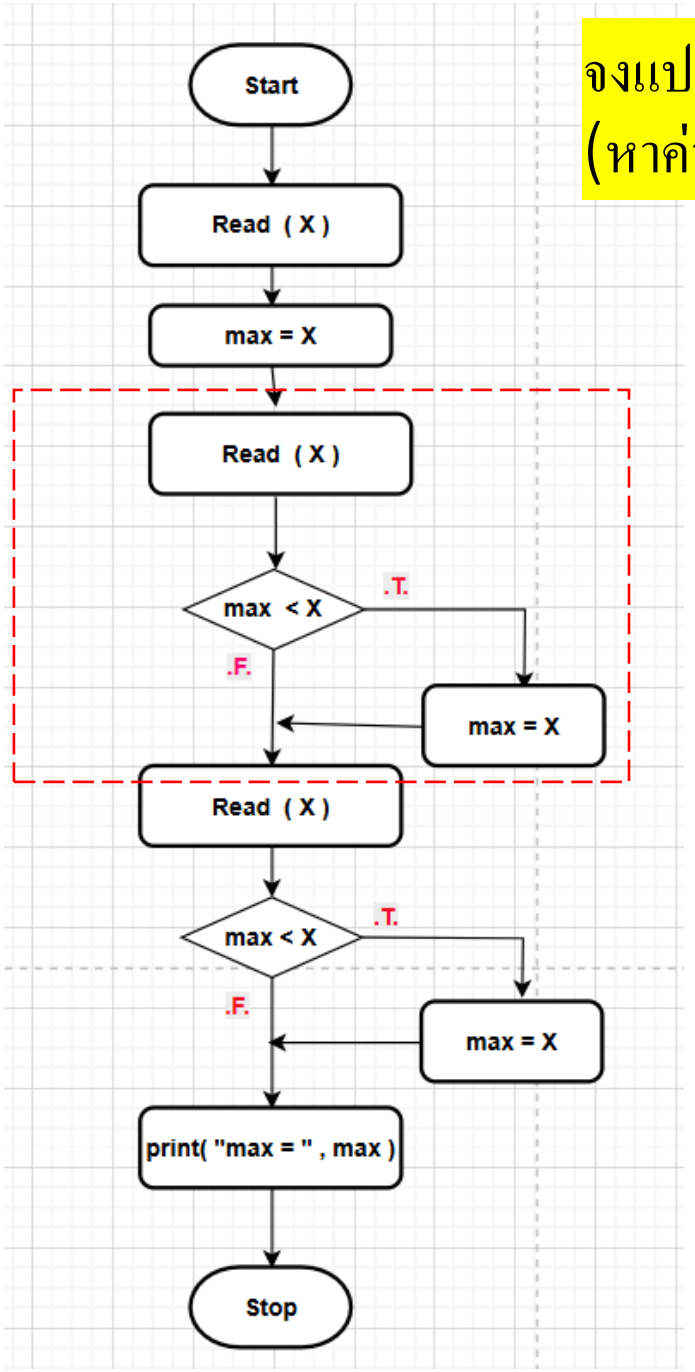




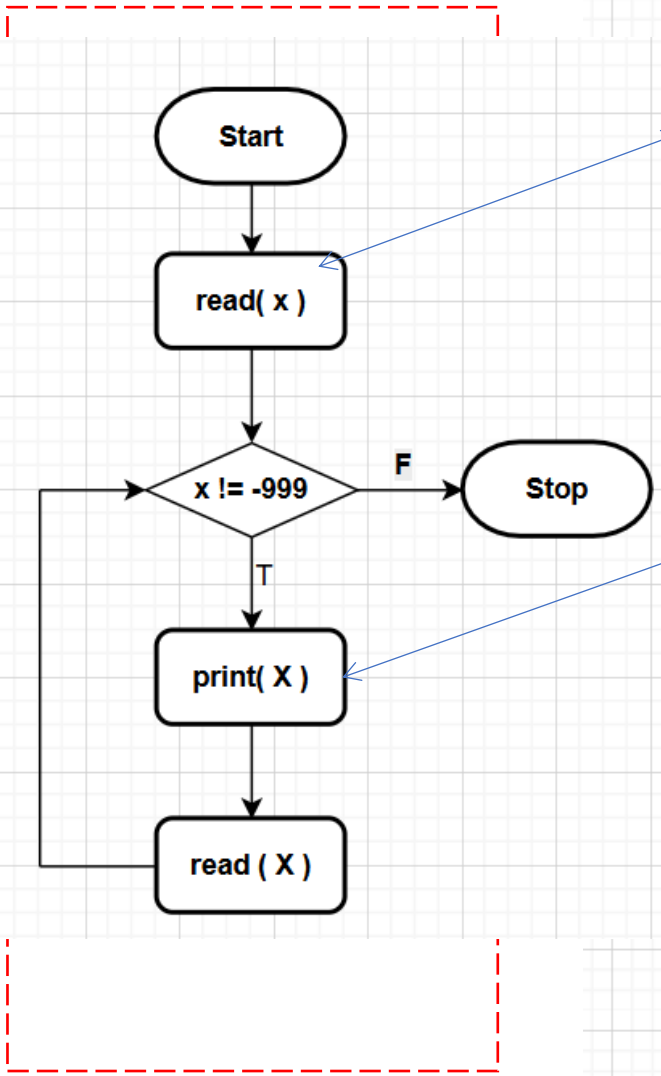
Flow หาค่า max จากข้อมูลเลข 3 ตัว

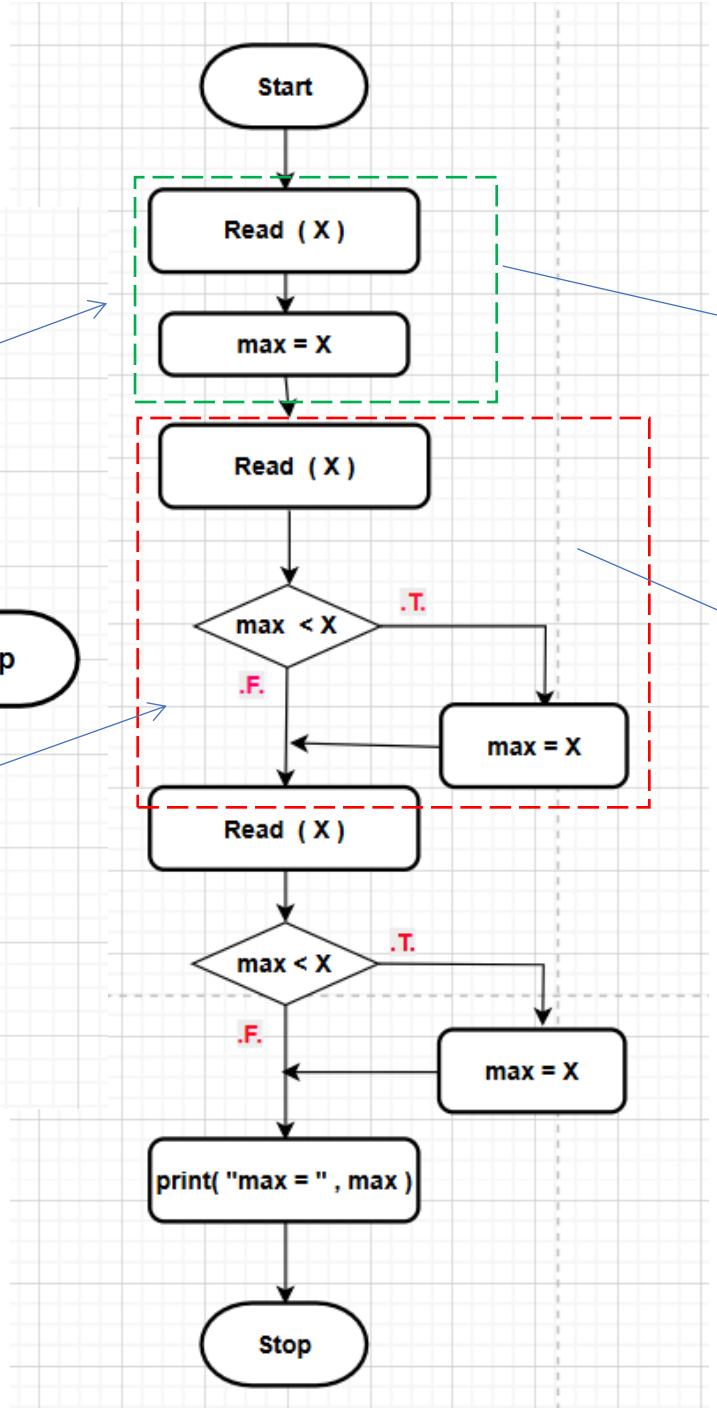


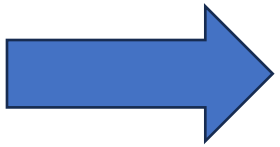
+

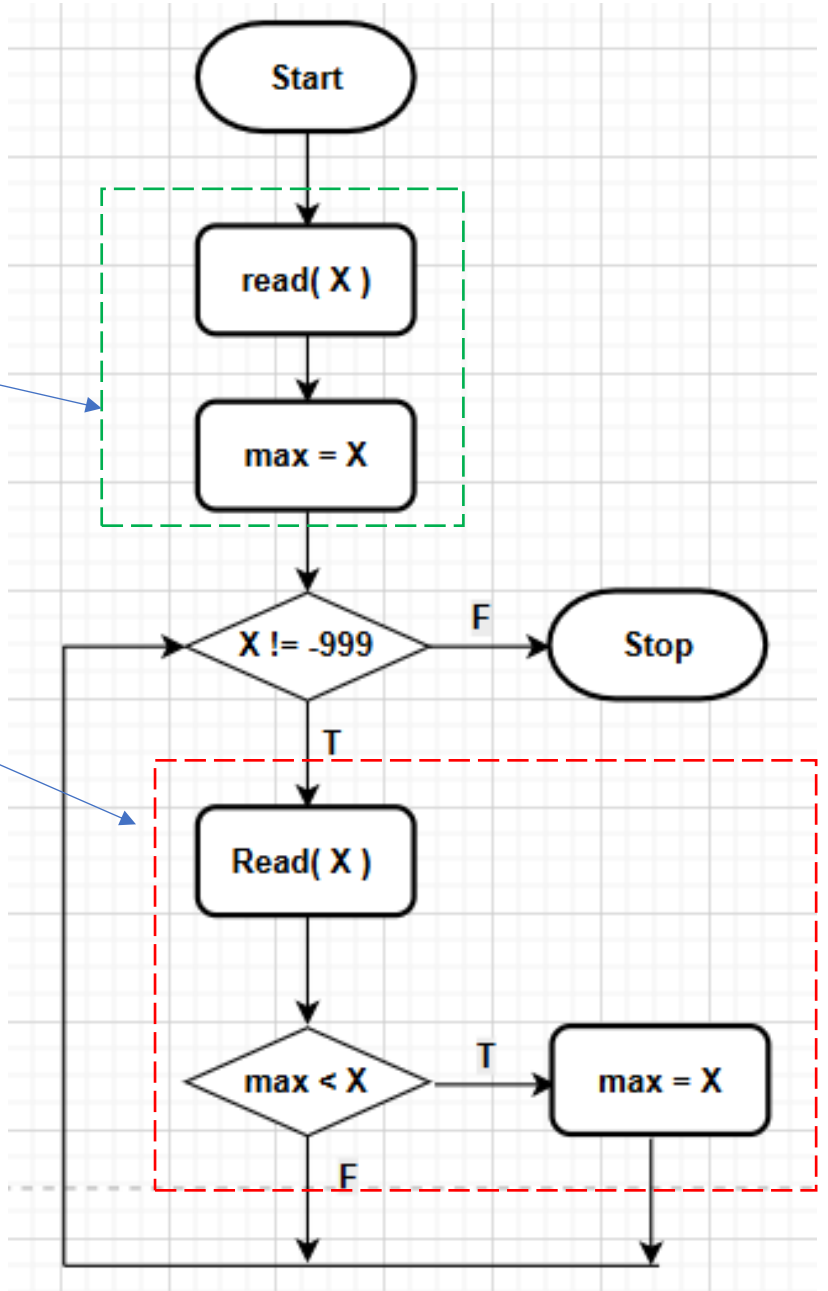


จงแปลงให้เป็นรับข้อมูลจน X=-999 (หาค่า max)

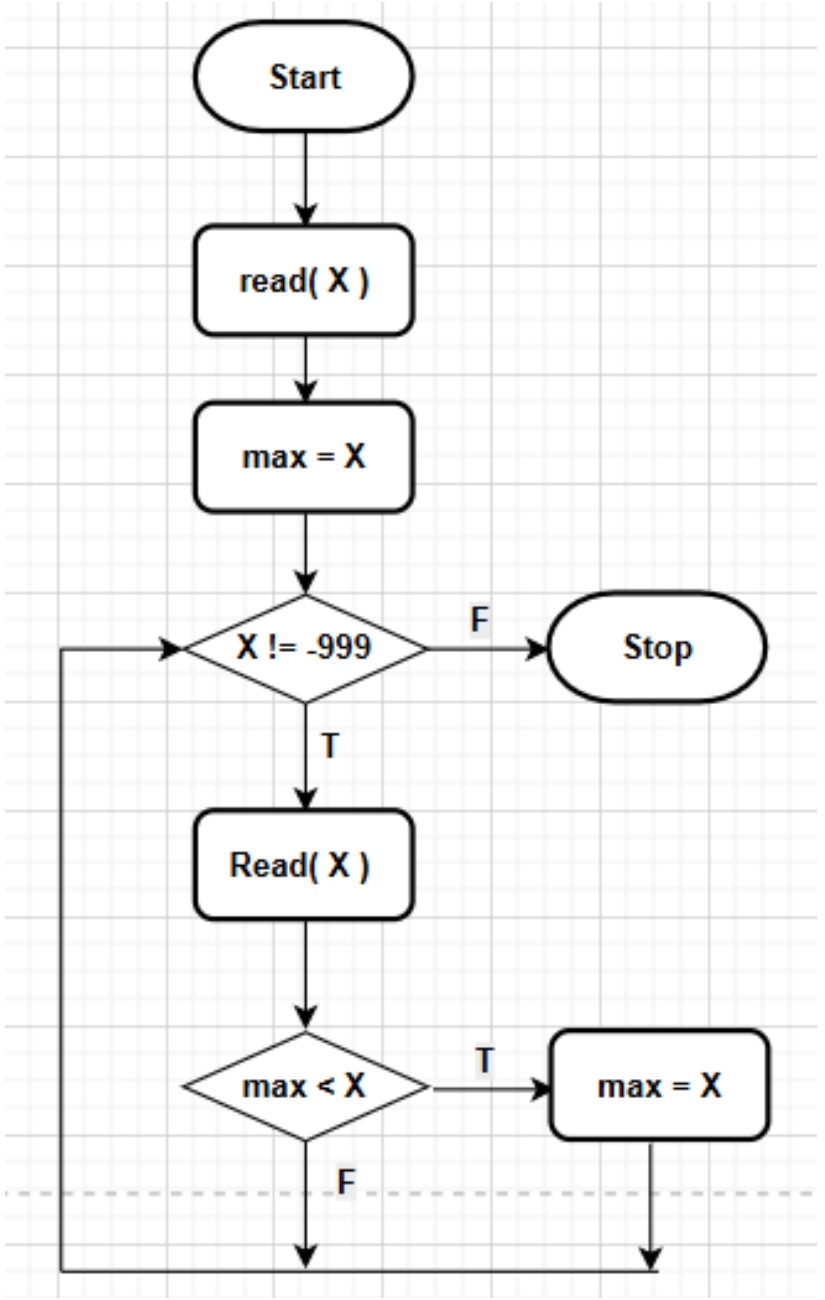

Start
read(x)
x != -999
F
Stop
T
print(X)
read (X)


Start
Read (X)
max = X
Read (X)
max < X
.T.
.F.
max = X
Read (X)
max < X
.T.
.F.
max = X
print("max = " , max)
Stop

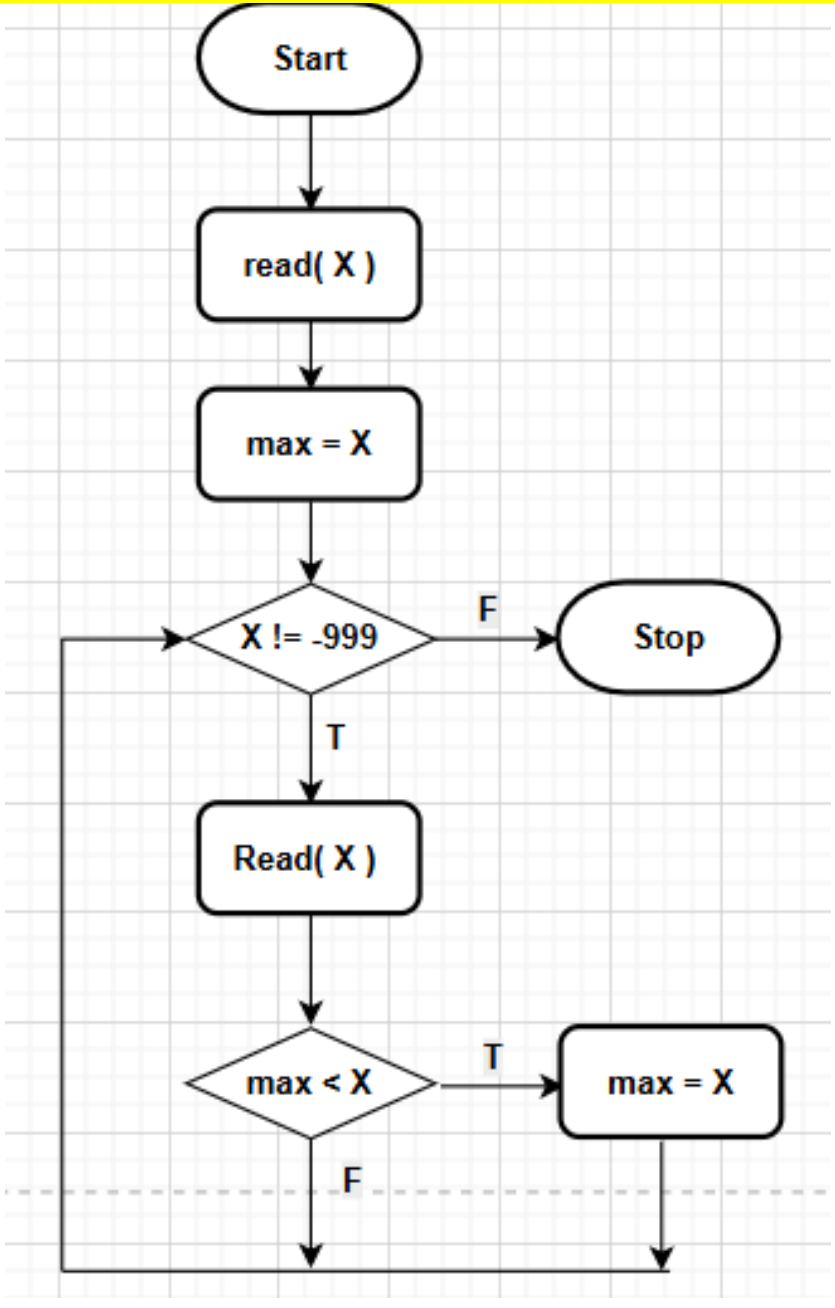


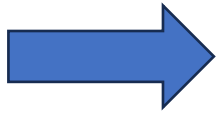

Start
read(X)
max = X
X != -999
F
Stop
T
Read(X)
max < X
T
max = X
F

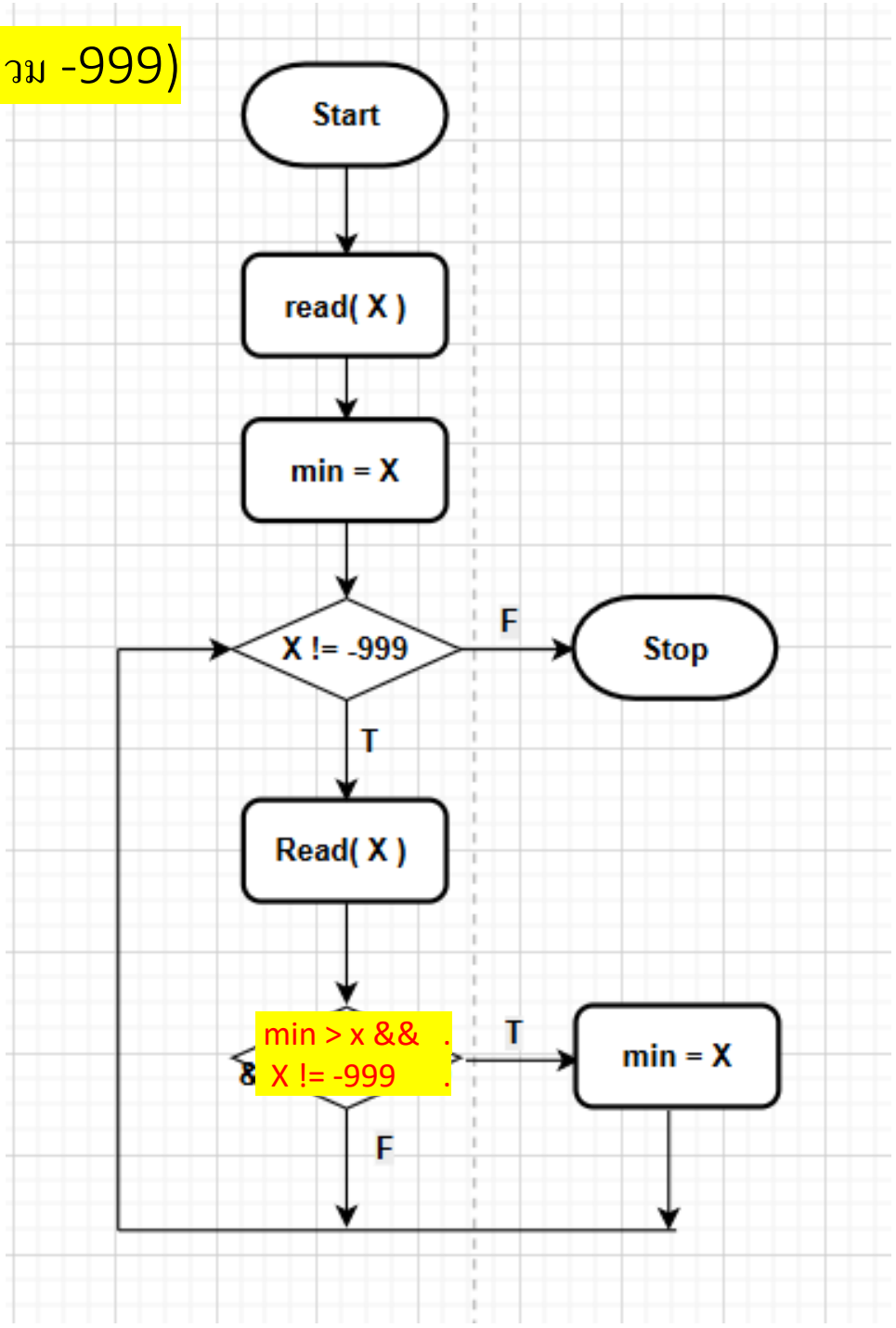
จงเขียน flow เพื่อหาค่าน้อยสุด
โดยรับข้อมูลจนกระทั้ง X = -999
(ค่าน้องสุดต้องไม่รวม -999)



จงเขียน flow เพื่อหาค่าน้อยสุด โดยรับข้อมูลจนกระทั้ง X = -999 (ค่าน้องสุดต้องไม่รวม -999)

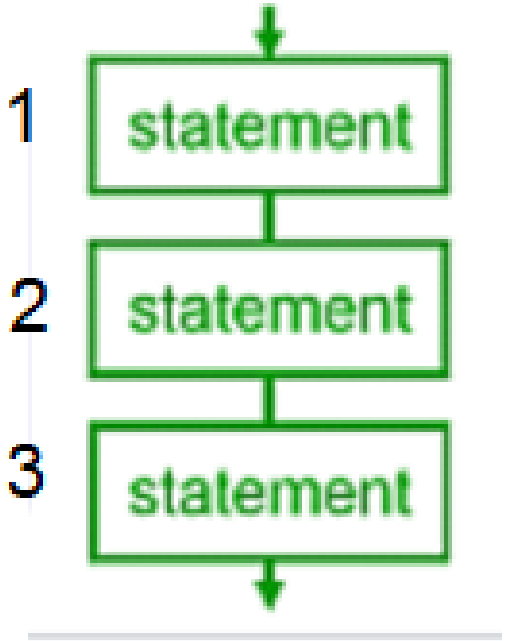


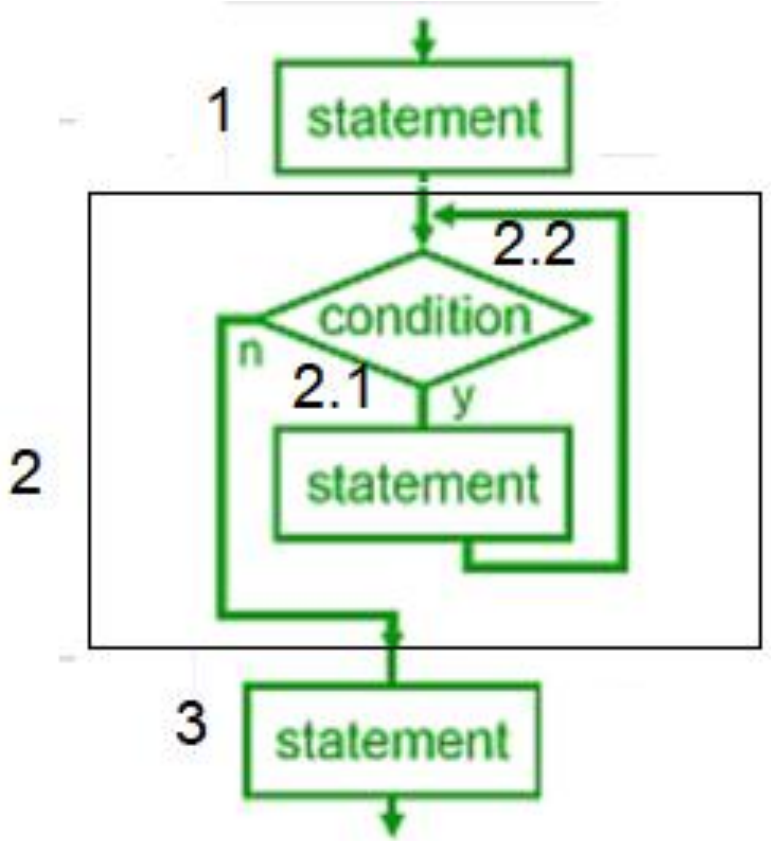


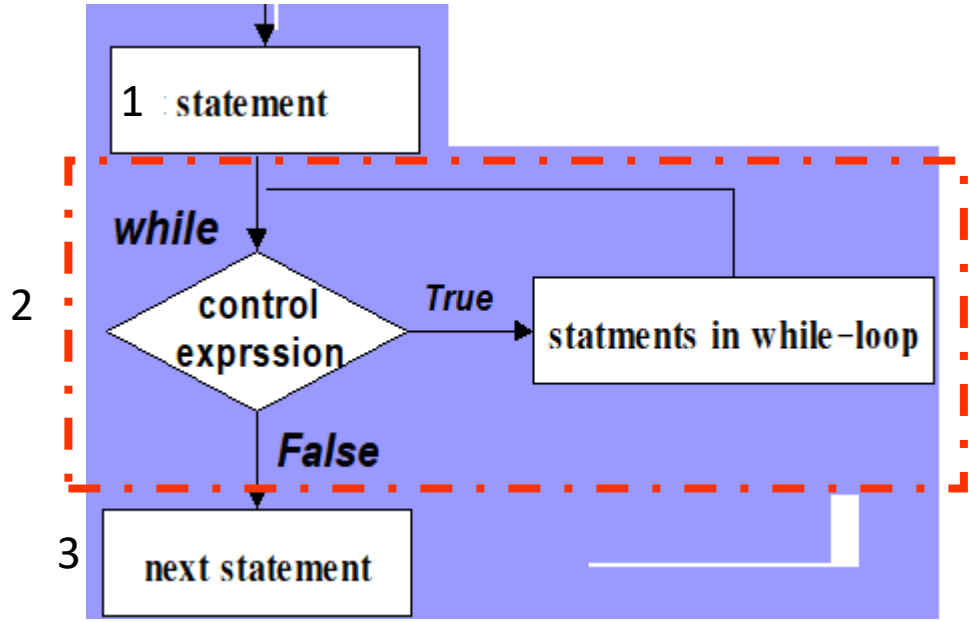


6.2. while Loops

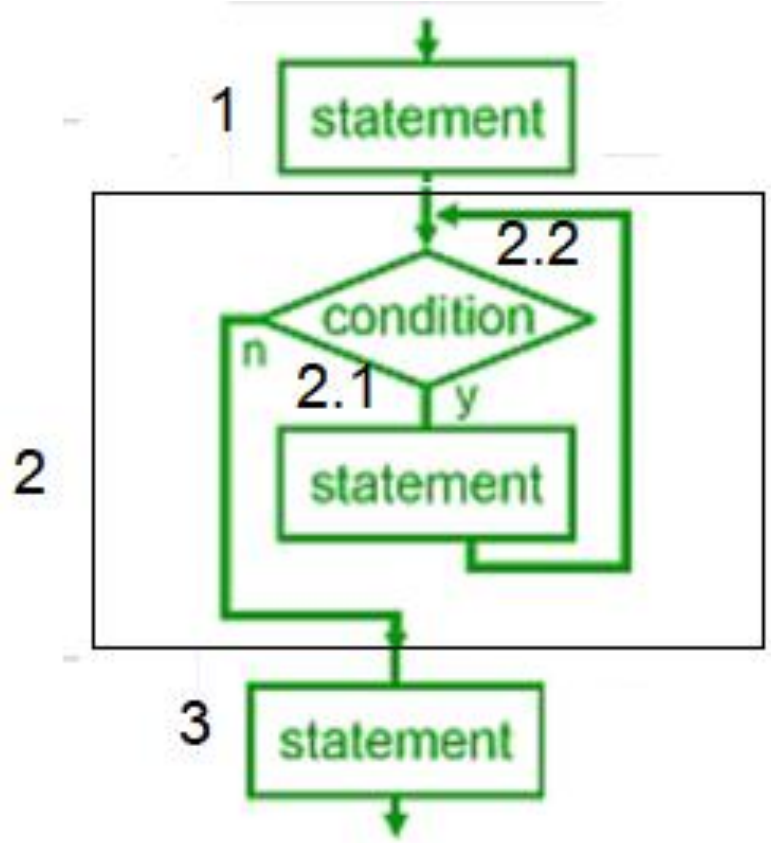
- *A while loop is similar to a for loops.*
- *A while loop is suitable when a loop-number is unknown exactly.*
- *A while loop repeats a statement within block as long as a control expression is true at the beginning of the loop.*

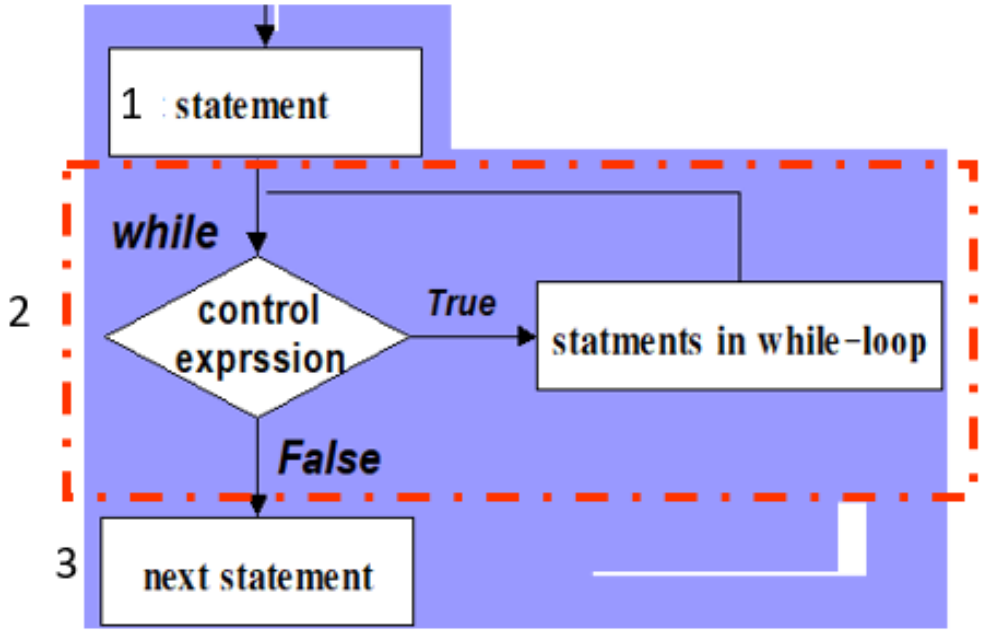

1
statement
2
statement
3
statement


1
statement
2.2
condition
n
2.1
y
statement
2
3
statement


1 statement
while
2
control
exprssion
True
statments in while-loop
False
3
next statement

```
while  control-expression do
 begin
       statements in while-loop
 end;
next statement
```





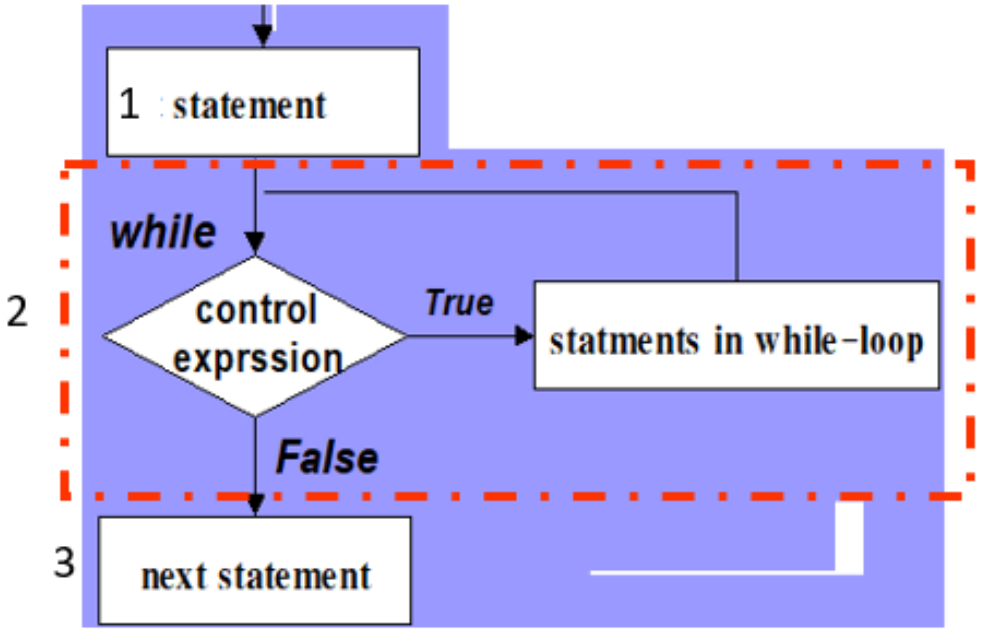
main.pas

```
1  program DivideNumber;
2  var
3    num: Real;
4  begin
5    Write('Please enter the number to divide: ');
6    ReadLn(num);
7
8    while num > 1.0 do
9    begin
10     WriteLn(num:0:2);
11     num := num / 2;
12   end;
13 end.
14
```

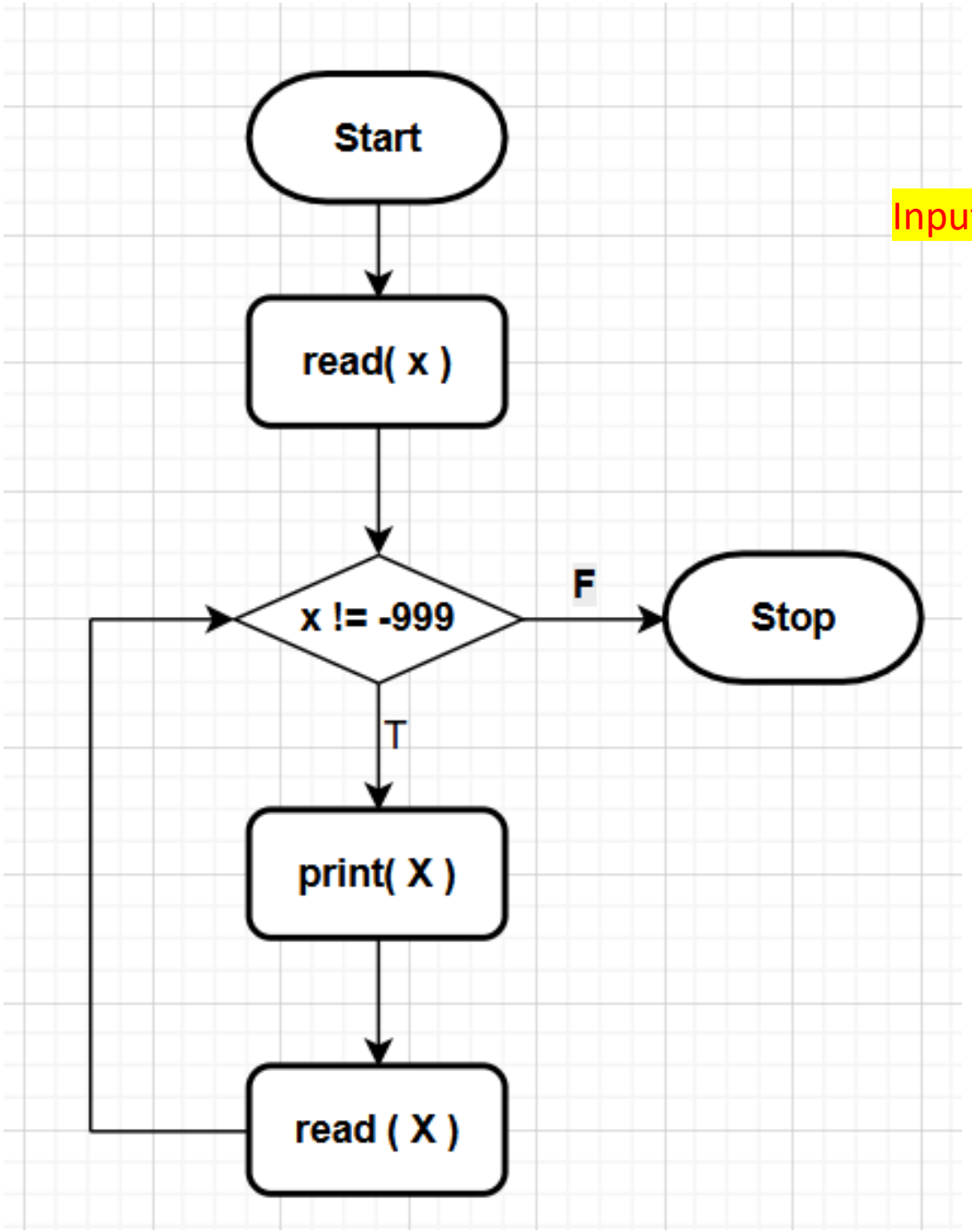
DivideNm.pas

```
Compiling main.pas
Linking a.out
13 lines compiled, 0.0 sec
Please enter the number to divide: 5
5.00
2.50
1.25
```

```
while control-expression do
  begin
    statements in while-loop
  end;
next statement
```



Convert to Pascal

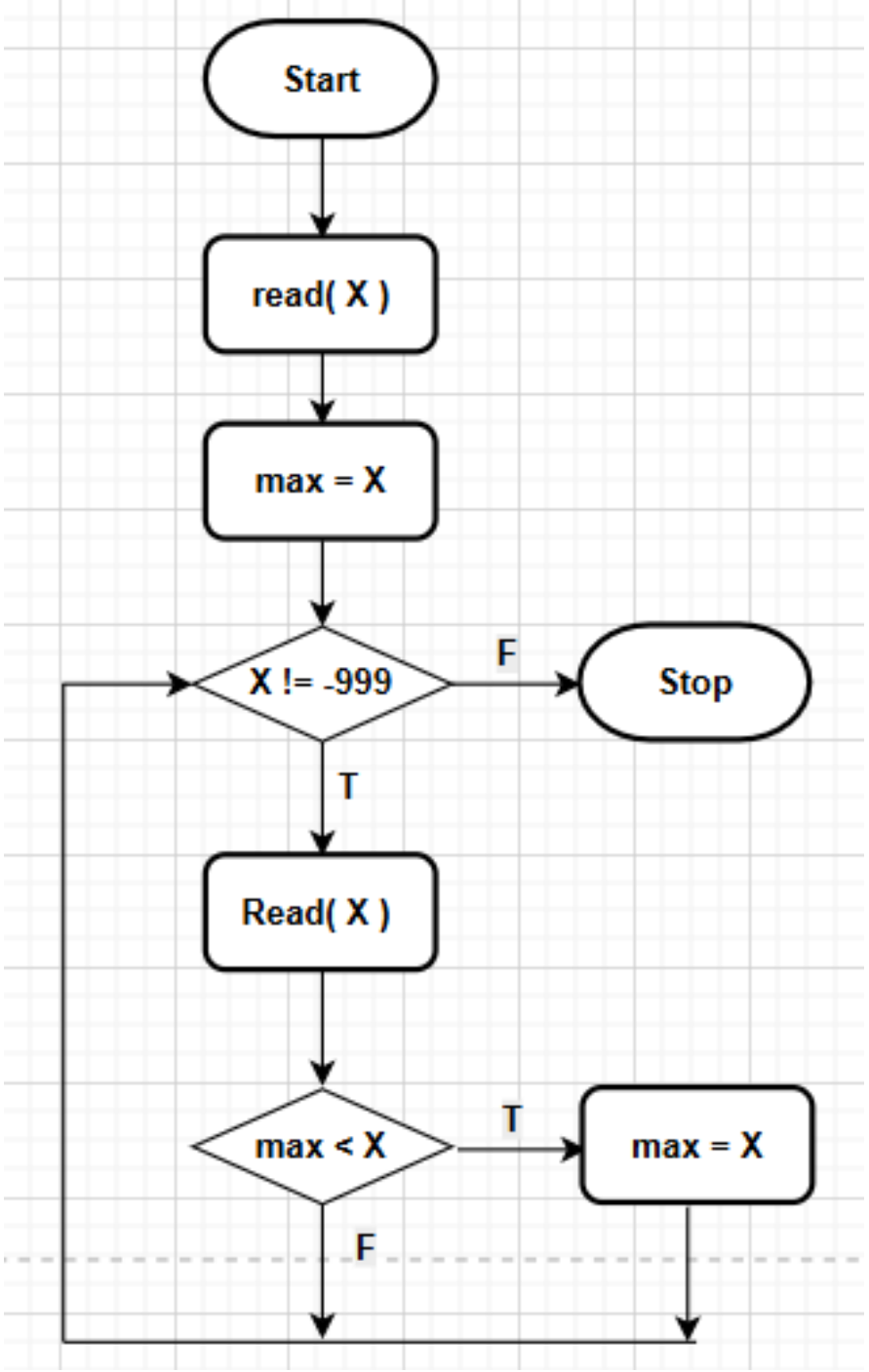


InputMin.pas

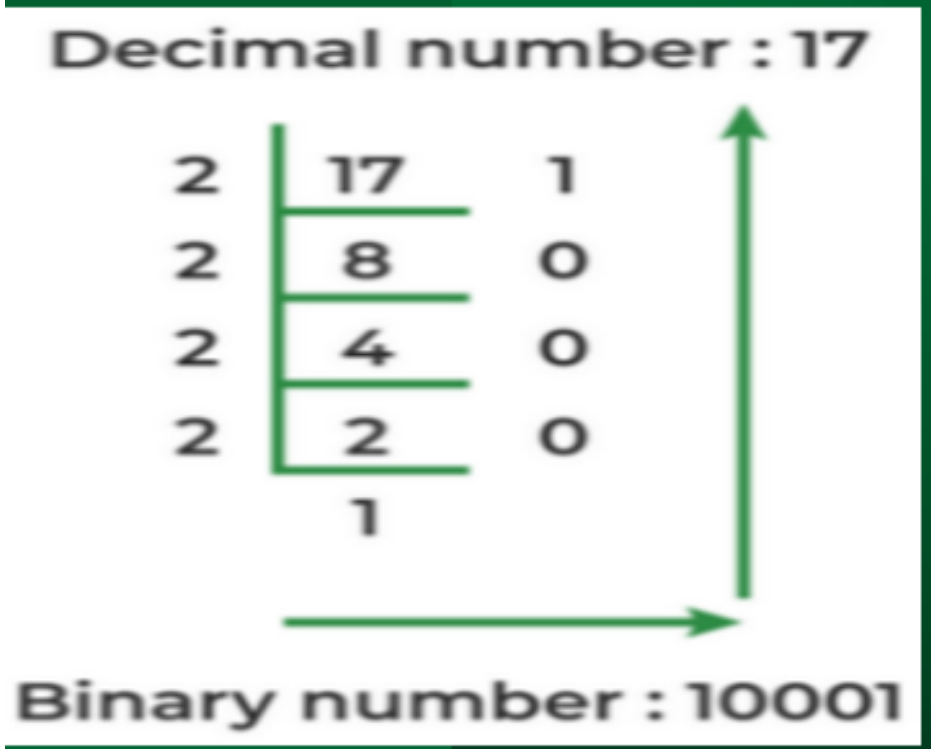
main.pas

```pascal
program InputUntilMinus999;
var
  x: Integer;

begin
  Write('Input (-999) x = ');
  ReadLn(x);

  while x <> -999 do
  begin
    WriteLn('x = ', x);
    Write('Input x (-999) = ');
    ReadLn(x);
  end;

end.
```

Convert to Pascal



จงเขียนโปรแกรมรับข้อมูลเลขจำนวนเต็มบวก
และแปลงให้เป็นฐาน 2



Input ? : 17

1

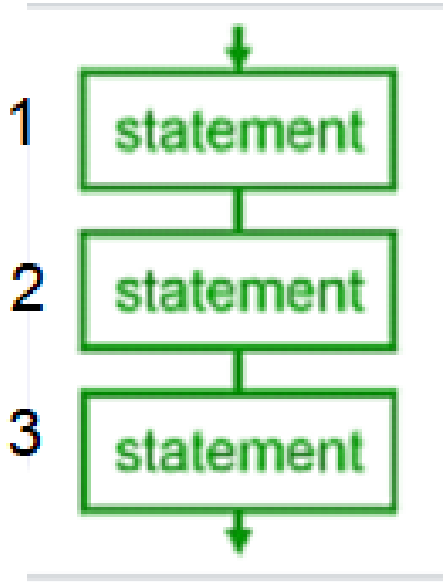
0

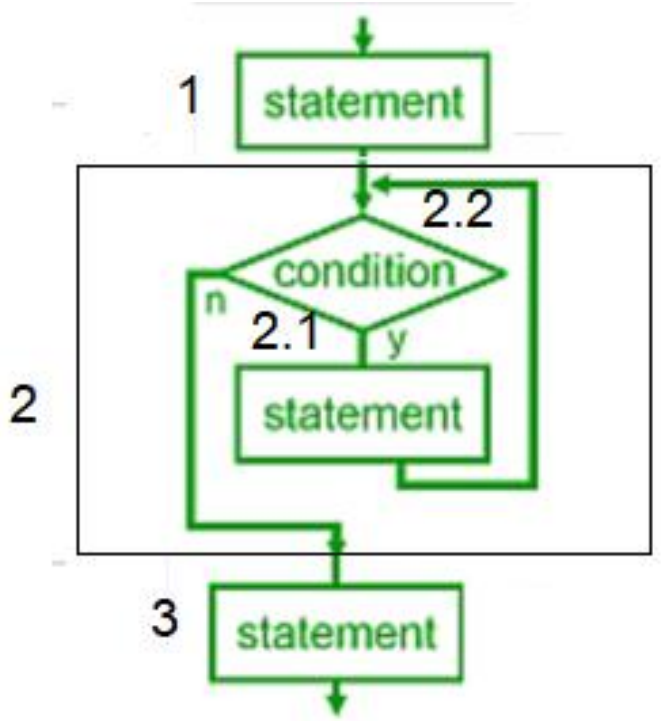
0

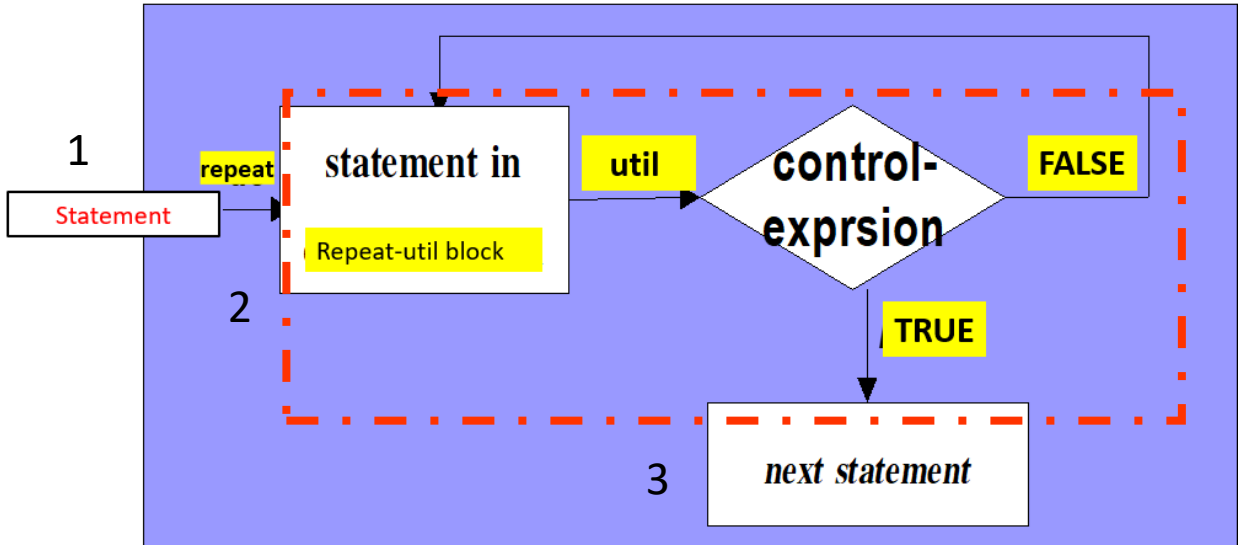
0

1

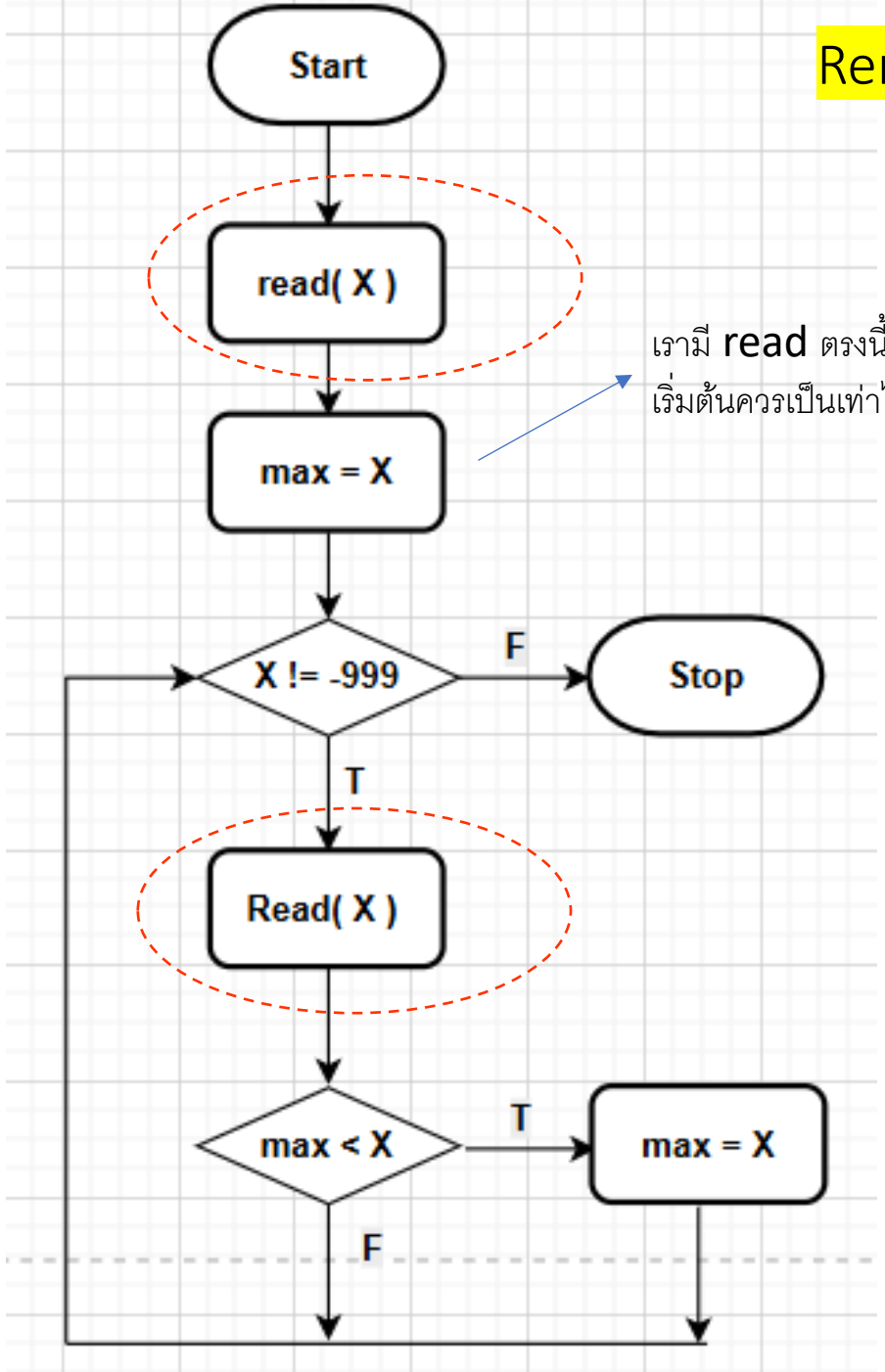
Repeat-util – Stm : unknown Repetition



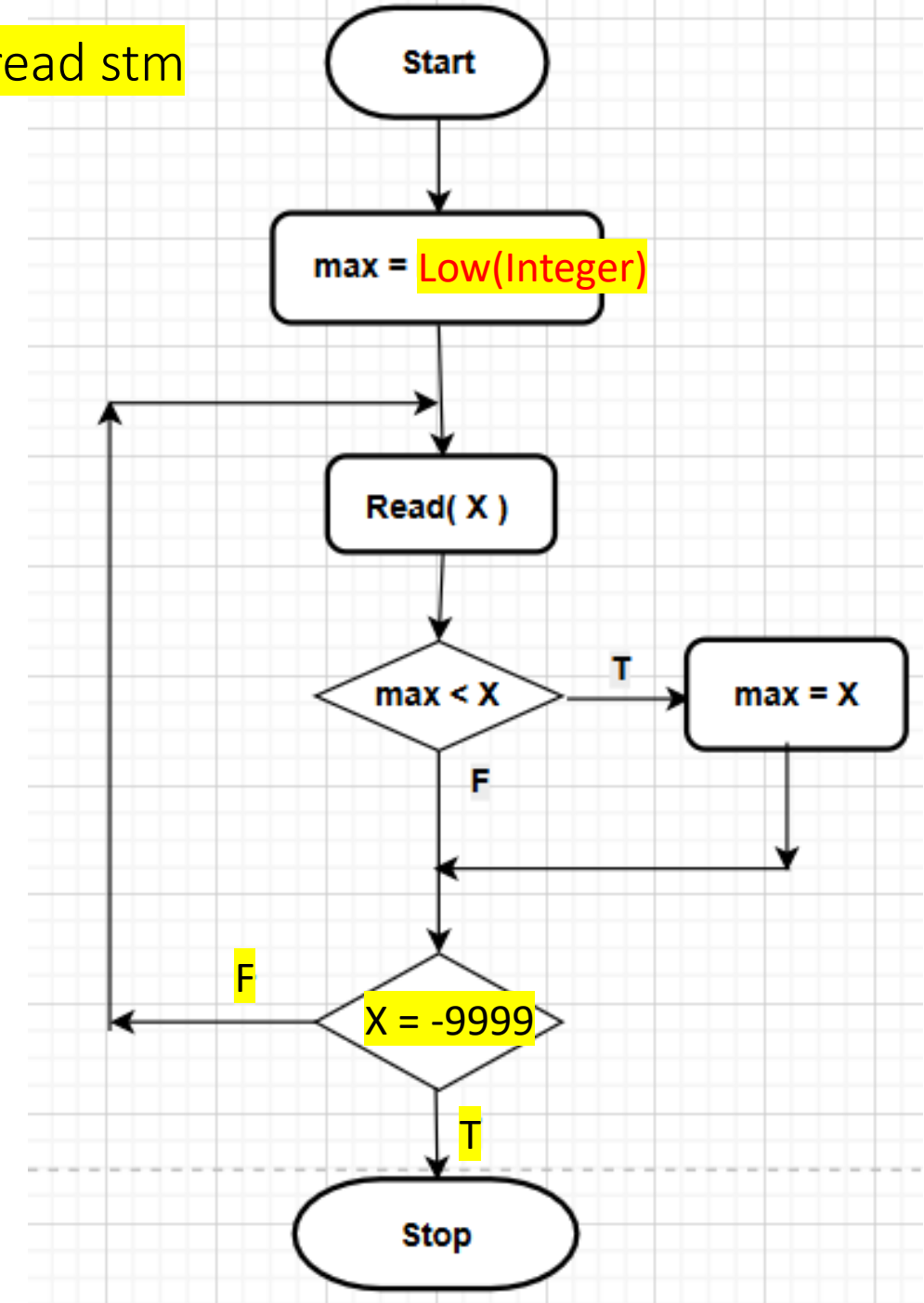




Remove the duplicate read stm



เรามี read ตรงนี้ เพราะไม่รู้ว่า max เริ่มต้นควรเป็นเท่าไร



6.2. repeat - util

- *A repeat-util loop repeats a statement within its block as long as a control expression is false at end of the loop.*

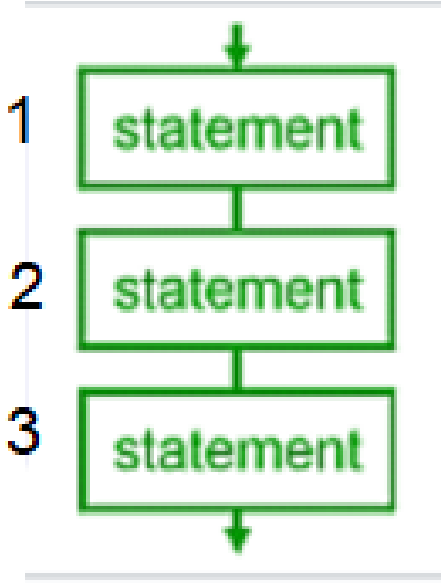
- *Statement in repeat-util block must be executed at least one time.*

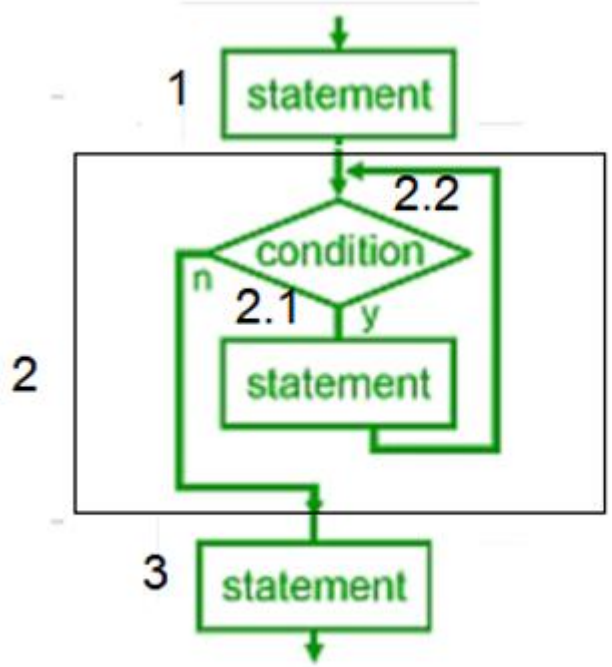
repeat

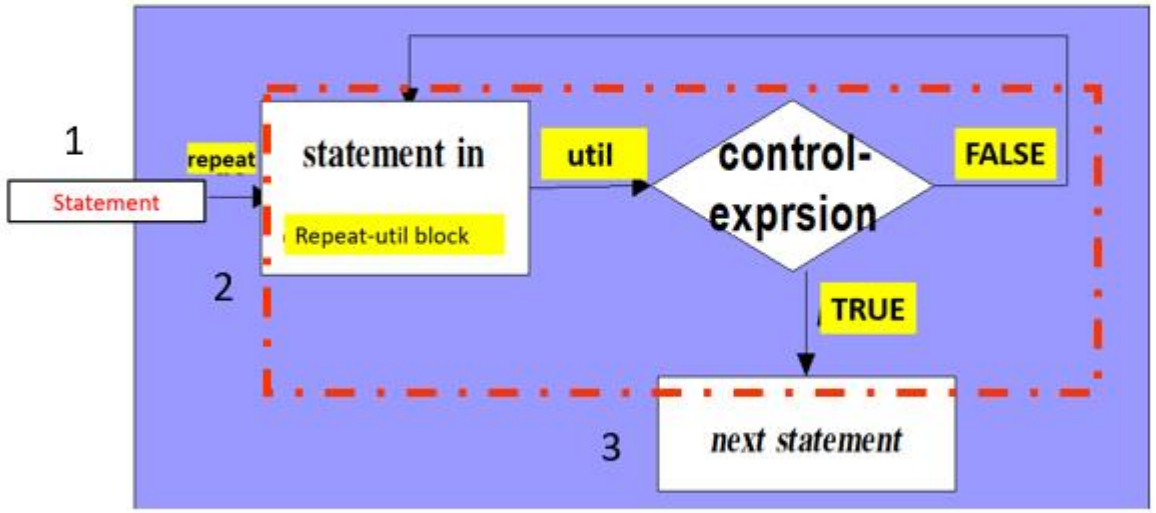
statements in repeat-util block

util <control-expression> ;

next statement





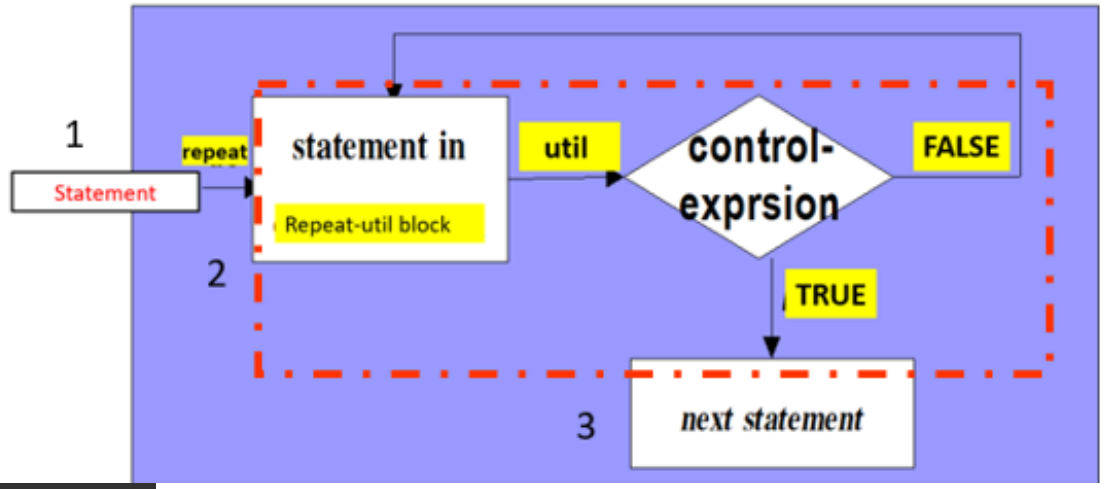


main.pas

```
1  program SquareRepeatUntil;
2
3  var
4    num, squared: Real;
5
6  begin
7    repeat
8      Write('Enter a number (0 to quit): ');
9      ReadLn(num);
10     squared := num * num;
11     WriteLn(num:0:2, ' squared is ', squared:0:2);
12   until num = 0;
13
14   WriteLn('Program ended.');
15 end.
```

SquareRep.pas

```
Compiling main.pas
Linking a.out
15 lines compiled, 0.0 sec
Enter a number (0 to quit): 4
4.00 squared is 16.00
Enter a number (0 to quit): 2
2.00 squared is 4.00
Enter a number (0 to quit): 1
1.00 squared is 1.00
Enter a number (0 to quit): 0
0.00 squared is 0.00
Program ended.
```



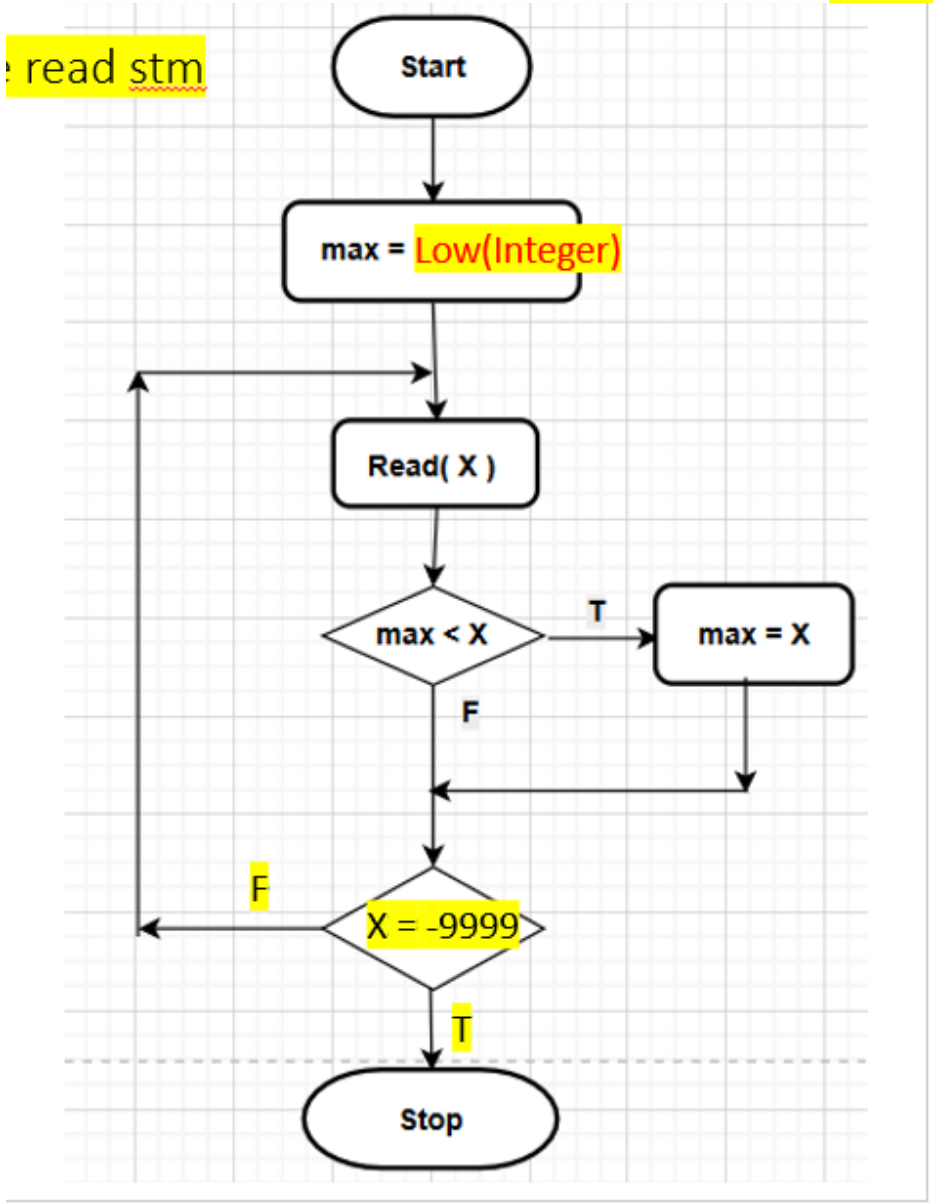
repeat

statements in repeat-util block

util <control-expression> ;

next statement

Convert to PASCAL



main.pas

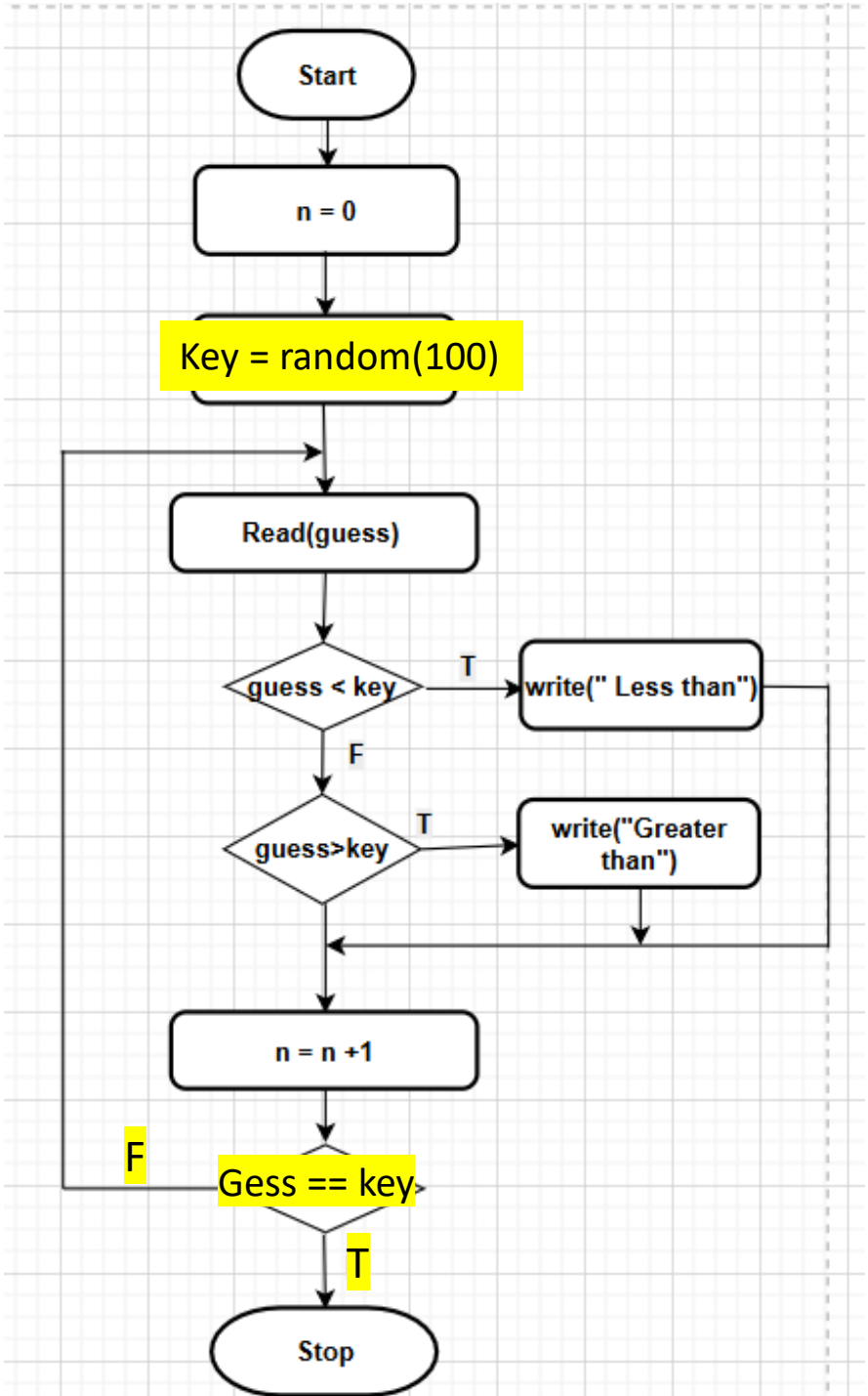
```pascal
program MinIntExample;
begin

  WriteLn('Min Integer = ', Low(Integer));
  WriteLn('Max Integer = ', High(Integer));

end.
```

minmax.pas

main.pas

```
1  program MinIntExample;
2  begin
3
4    WriteLn('Min Integer = ', Low(Integer));
5    WriteLn('Max Integer = ', High(Integer));
6
7  end.
8
```

minmax.pas



Convert to PASCAL

คำสั่งที่ใช้ในการสุ่มเลขของภาษา PASCAL

main.pas

```
1  program RandomExample;
2
3  begin
4    Randomize;                 // Seed the generator
5
6    WriteLn('Random integer 0..99: ', Random(100));
7    WriteLn('Random integer 0..high(integer) : ', Random( high(integer)));
8  end.
9
```

Ran.pas

Free-PASCAL

break & continue stm.

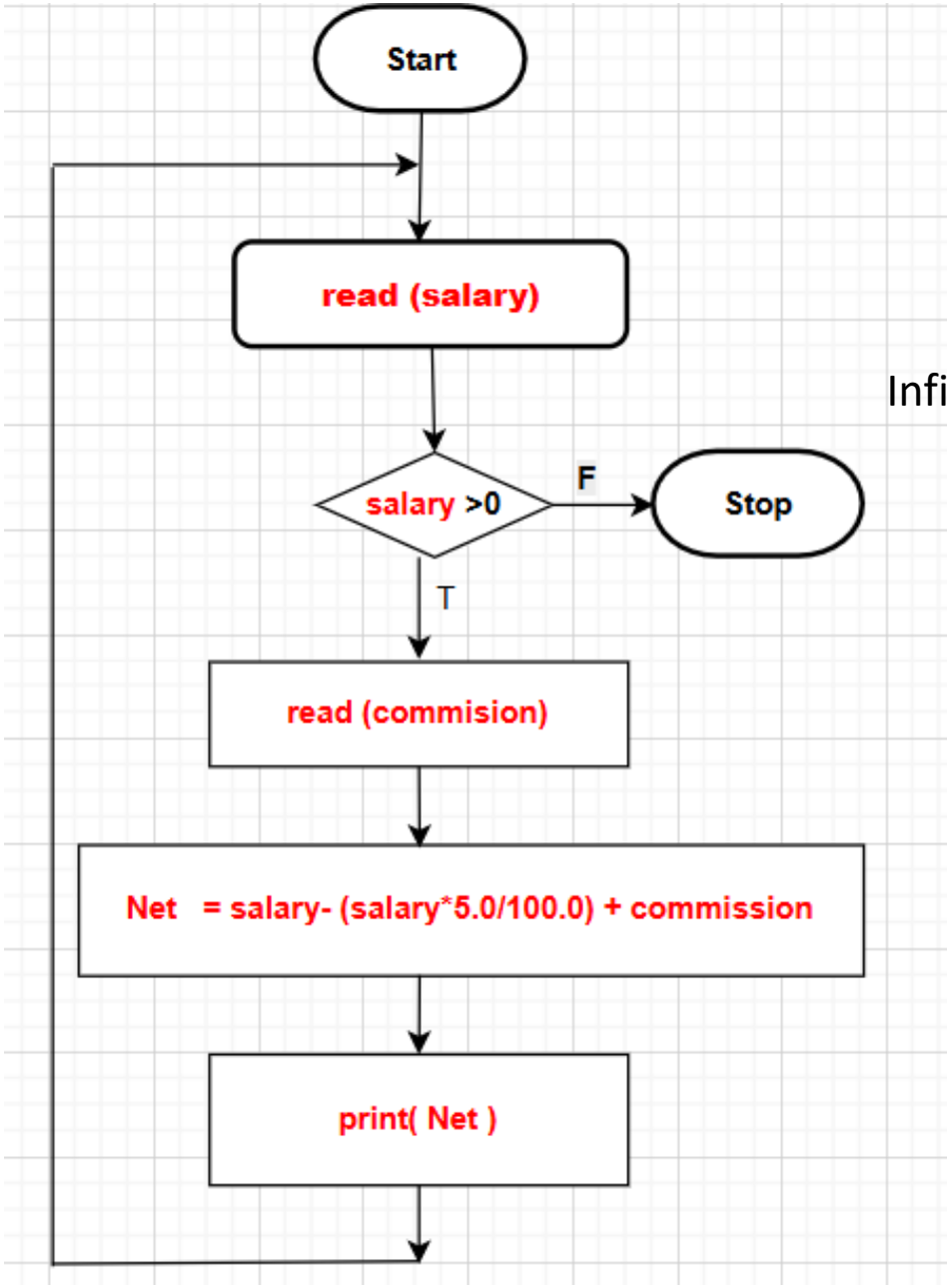
Break statement

เขียนโปรแกรมรับข้อมูลของพนักงานเพื่อคำนวณเงินเดือนสุทธิของพนักงาน โดยมีละเอียดข้อมูลดังนี้

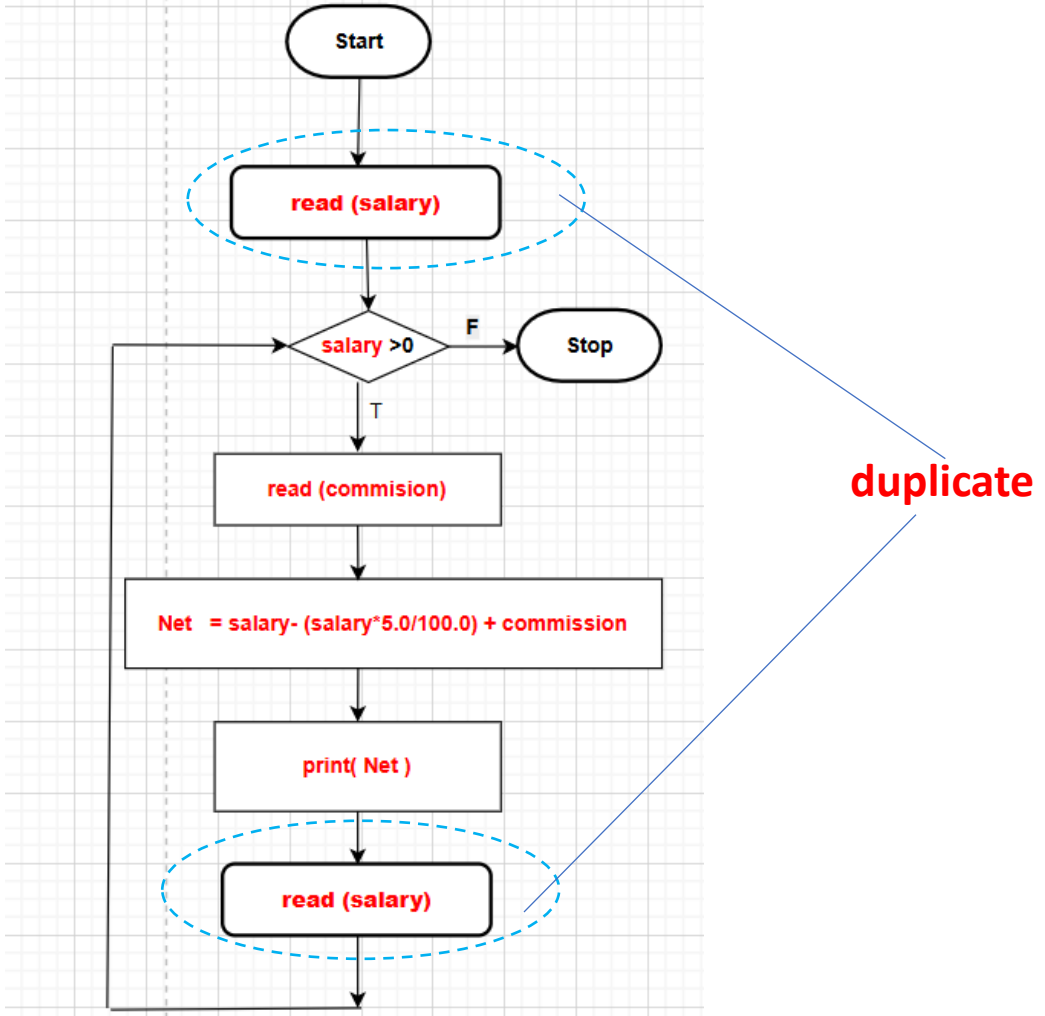
- รับเงินเดือนพื้นฐาน (salary)
- หักภาษี 5% (salary* 5.0/100.0)
- รับค่าคอมมิชชั่นเพิ่มเติม (commission)
- แสดงเงินเดือนสุทธิ = (เงินเดือน – ภาษี) + ค่าคอมมิชชั่น

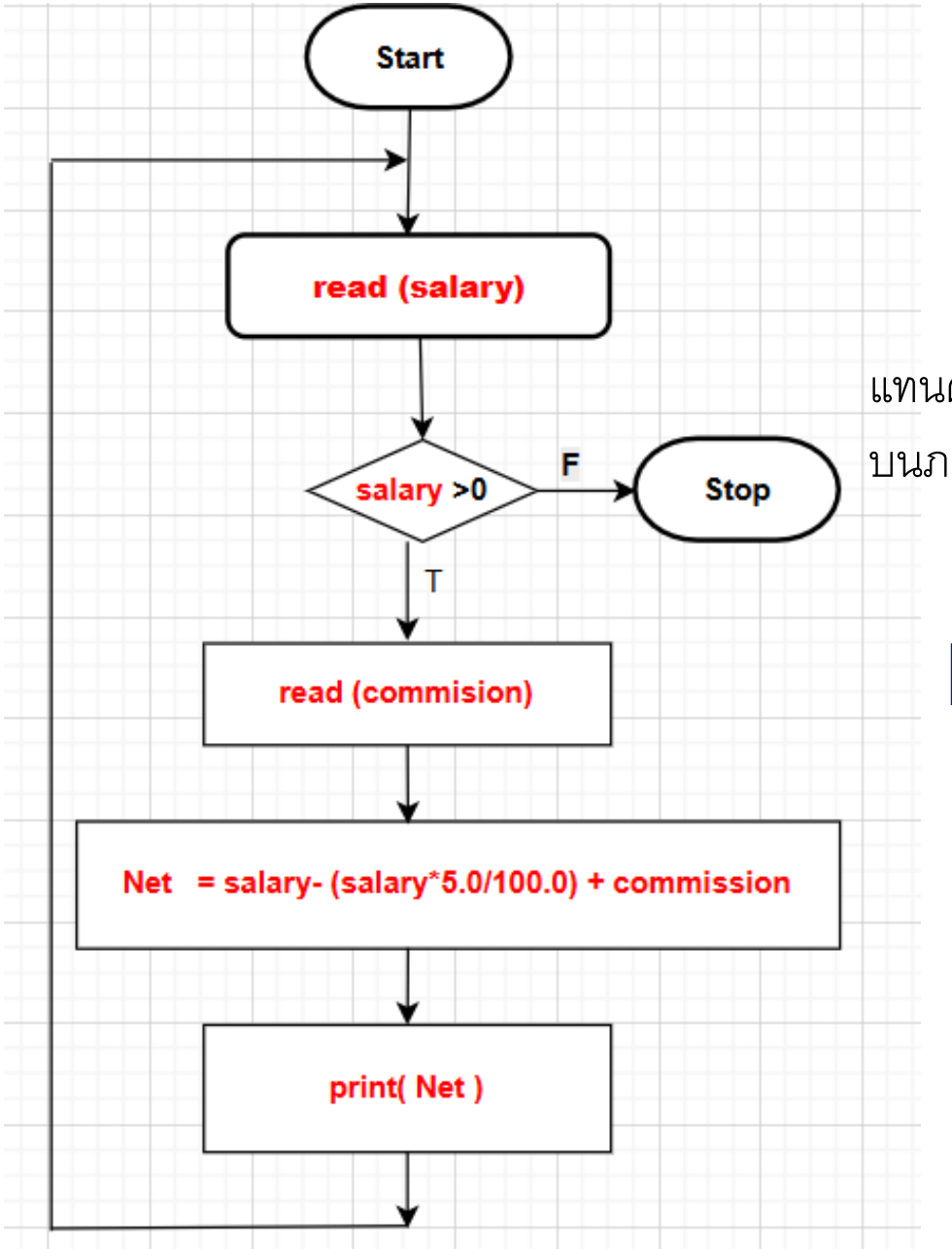
Net = salary- (salary* 5.0/100.0) + commission

โดยรับข้อมูลจนกระทั่งค่าเงินเดือนพื้นฐาน (salary) มีค่าน้อยกว่า 0



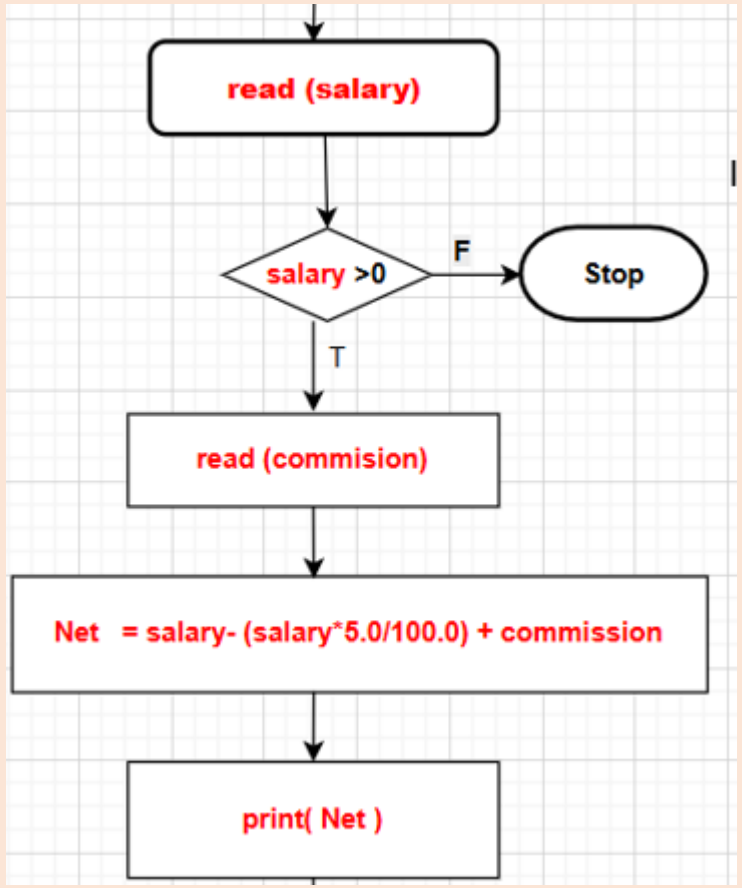
Infinite loop





แทนด้วยคำสั่ง while .true.
บนภาษาโปรแกรม

While (True)
Begin



End;

break Statement

most lang. such as C/C++,java,
the **break** statement terminates the loop when it is encountered.

breakfor.pas

```
1  program BreakContinueExample;
2
3  var
4    i: Integer;
5
6  begin
7
8    for i := 1 to 10 do
9    begin
10
11     if i = 3 then
12       break;
13
14     writeln('i : ' , i );
15   end;
16   writeln('end : ' , i );
17 end.
18
```

main.pas

breakwhile.pas

```
1  program BreakContinueExample;
2  var
3    i: Integer;
4  begin
5    i := 1;
6    while( true ) do
7    begin
8
9      if i = 3 then
10       break;
11
12     writeln('i : ' , i );
13     i := i+1;
14
15   end;
16   writeln('end : ' , i );
17
18 end.
19
```

break Statement-for

most lang. such as C/C++,java,
the **break** statement terminates the loop when it is encountered.

```
1  program BreakContinueExample;
2
3  var
4    i: Integer;
5
6  begin
7
8    for i := 1 to 10 do
9    begin
10
11     if i = 3 then
12       break;
13
14     writeln('i : ' , i );
15   end;
16   writeln('end : ' , i );
17 end.
18
```

breakfor.pas

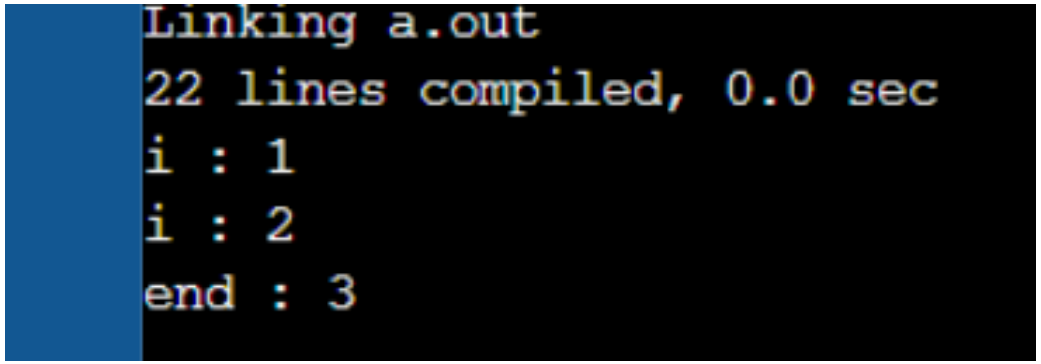
```
Free Pascal Compiler version
Copyright (c) 1993-2021 by F
Target OS: Linux for x86-64
Compiling main.pas
Linking a.out
17 lines compiled, 0.0 sec
i : 1
i : 2
end : 3
```

break Statement - while

most lang. such as C/C++,java,
the **break** statement terminates the loop when it is encountered.

main.pas

```
1  program BreakContinueExample;
2  var
3    i: Integer;
4  begin
5    i := 1;
6    while( true ) do
7    begin
8
9      if i = 3 then
10       break;
11
12     writeln('i : ' , i );
13     i := i+1;
14
15   end;
16   writeln('end : ' , i );
17
18 end.
19
```



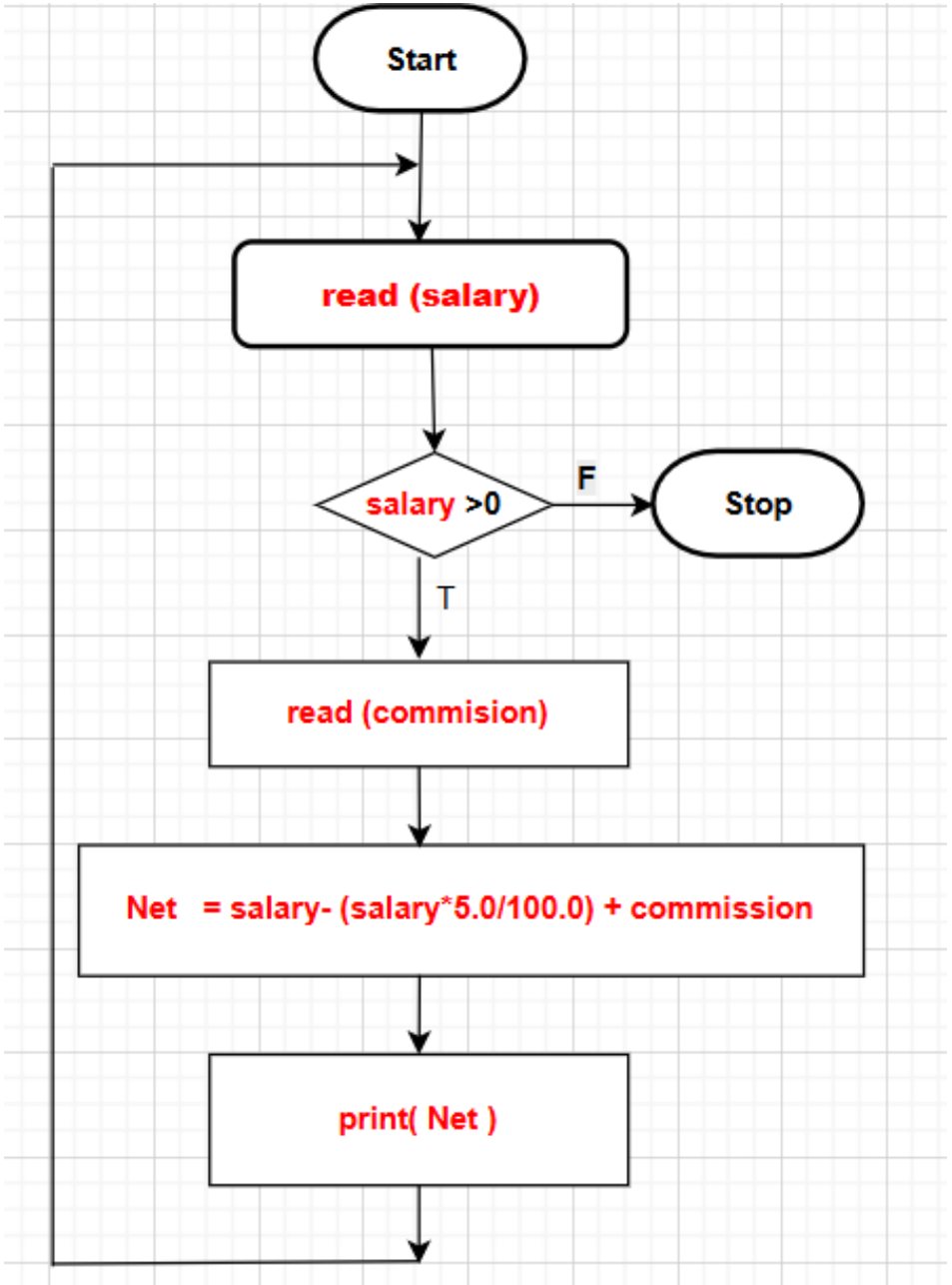
main.pas

```pascal
program NestedForBreakStructured;
var
  i, j: Integer;

begin
  for i := 1 to 3 do
  begin

    for j := 1 to 4 do
    begin
      if j = 3 then
      begin
        Break;  // ❌ not standard Pascal, so we
      end;
      WriteLn('inner -> i = ', i, ', j = ', j);
    end;
    WriteLn('outer -> i = ', i, ', j = ', j);
  end;
end.
```

ForNestBreak.pas

```
Linking a.out
25 lines compiled, 0.0 sec
inner -> i = 1, j = 1
inner -> i = 1, j = 2
outer -> i = 1, j = 3
inner -> i = 2, j = 1
inner -> i = 2, j = 2
outer -> i = 2, j = 3
inner -> i = 3, j = 1
inner -> i = 3, j = 2
outer -> i = 3, j = 3
```

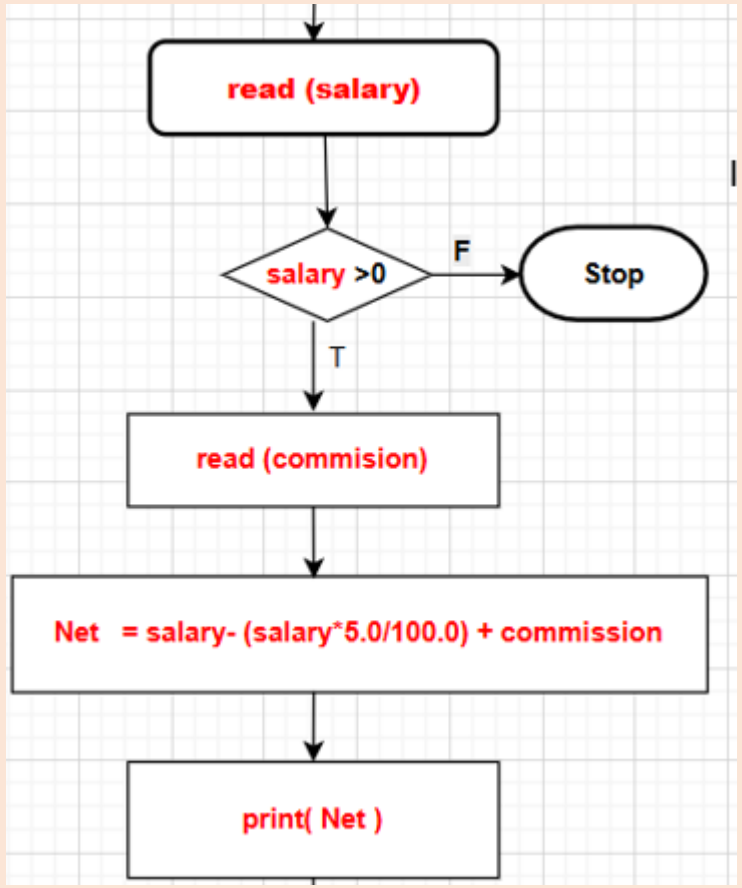
จงแทนด้วยภาษา PASCAL



แทนด้วยคำสั่ง

บนภาษาโปรแกรม

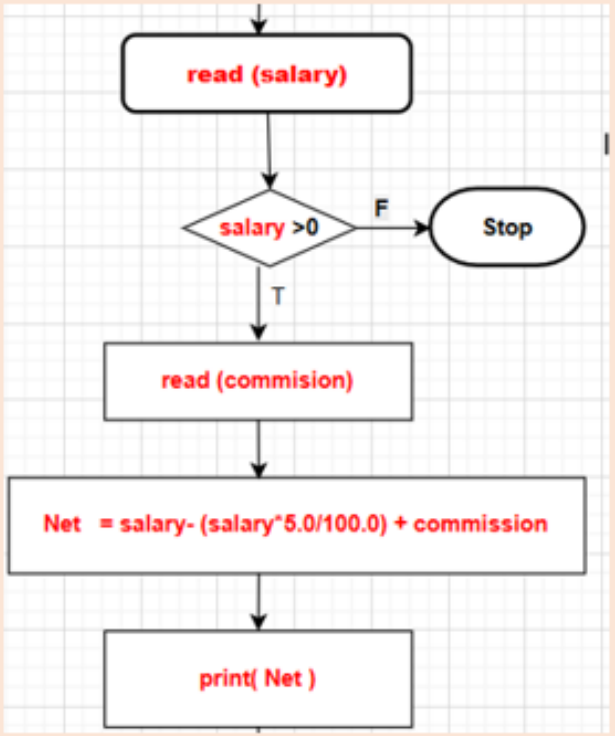
While (True) DO
Begin



End;

CODE

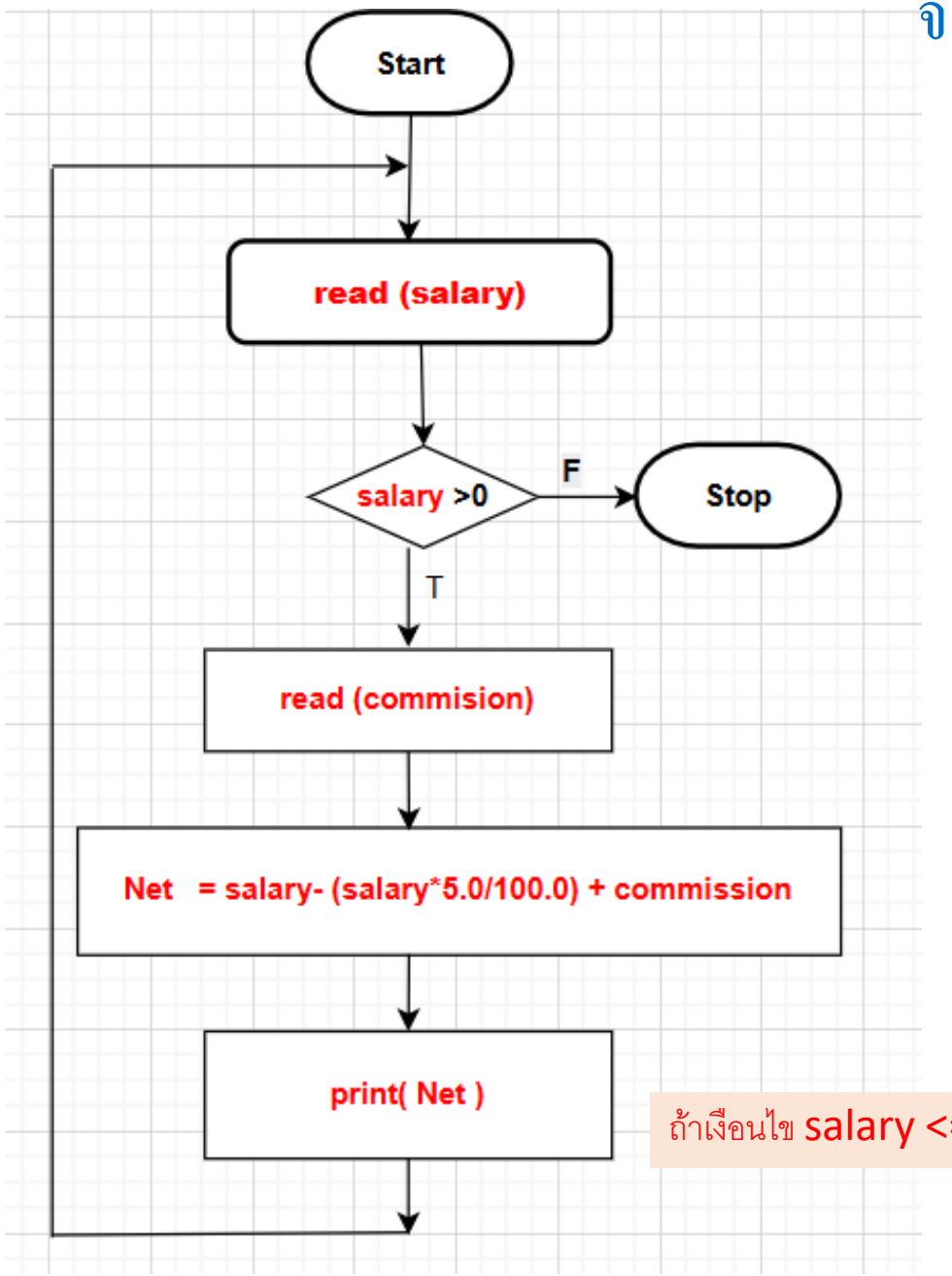
While (True) DO
Begin



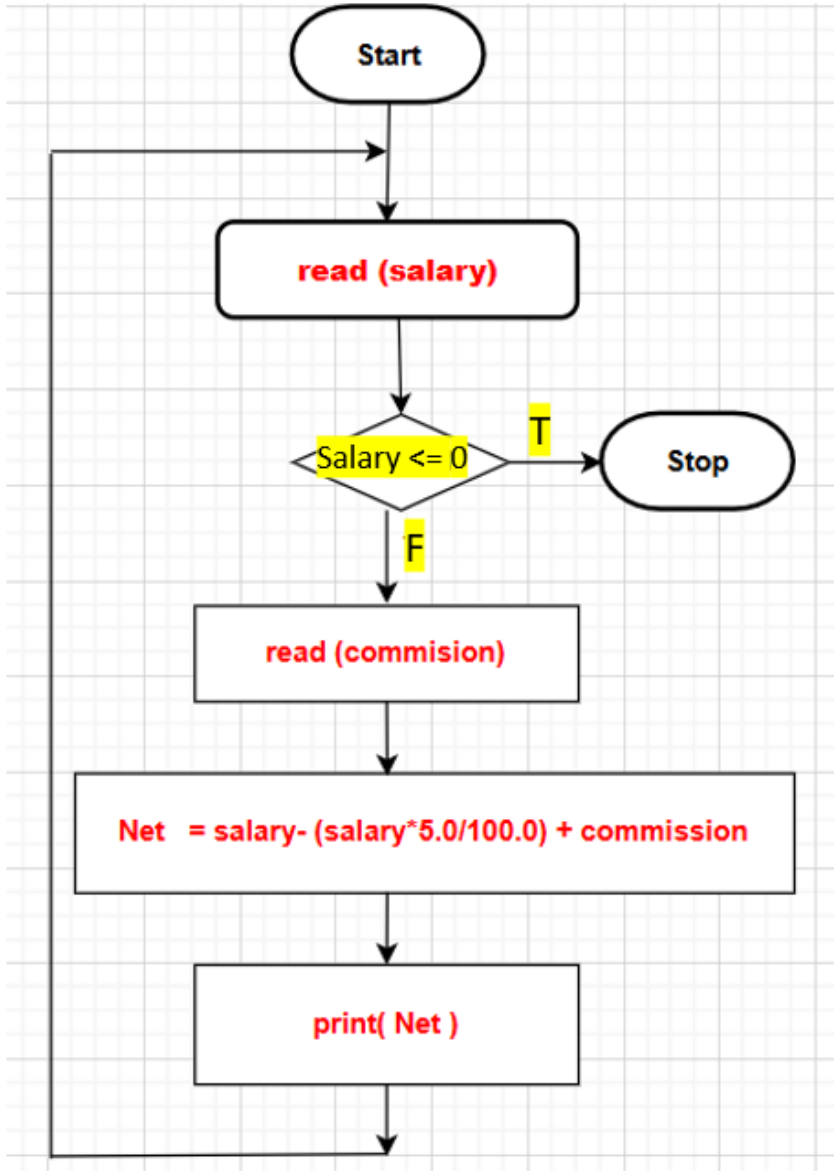
End;

```
1
2  program salaryWTIfElse;
3  var
4   salary,commission,net:real;
5  begin
6
7    while ( true) do
8    begin
9       write( 'salary (-999)= ');
10      read( salary);
11      if (salary > 0 ) then
12        begin
13          write( 'commission = ');
14          read( commission );
15          net := salary-(salary*0.5/100.0)+ commission;
16          writeln( 'net = ',net:0:2);
17        end
18      else
19        break;
20   end;
21   write('end app');
22 end.
23
```

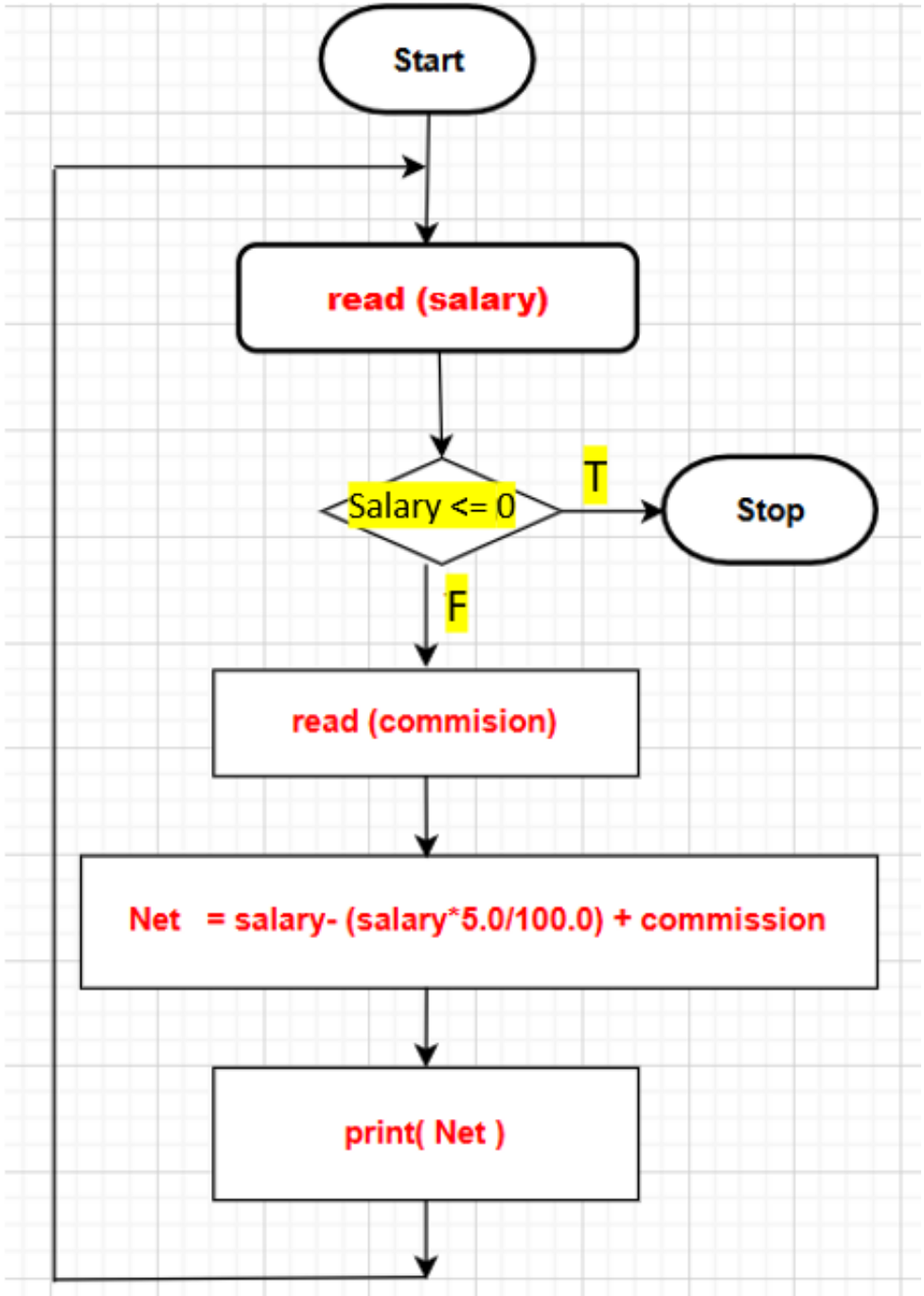
จงแทนด้วยภาษา PASCAL



Change logic



ถ้าเงื่อนไข salary <= 0 เป็นจริง จะออกจาก loop



Convert to while&break

continue Statement

The "**continue**" statement is used to skip the current iteration of the loop and the control of the program goes to the next iteration

ain.pas

```
1  program BreakContinueExample;
2
3  var
4    i: Integer;
5
6  begin
7
8    for i := 1 to 10 do
9    begin
10
11     if i >= 3 then
12       continue;
13
14     writeln('i : ' , i );
15   end;
16   writeln('end : ' , i );
17 end.
```

```
1  program BreakContinueExample;
2
3  var
4    i: Integer;
5
6  begin
7    i := 0;
8    while i <> 10  do
9    begin
10     i := i +1;
11     if i >= 4 then
12       continue;
13     writeln('i : ' , i );
14   end;
15   writeln('end : ' , i );
16 end.
17
```

continue Statement

most lang. such as C/C++,java,
the **continue** statement terminates the loop when it is encountered.

ain.pas

```
1  program BreakContinueExample;
2
3  var
4    i: Integer;
5
6  begin
7
8    for i := 1 to 10 do
9    begin
10
11     if i >= 3 then
12       continue;
13
14     writeln('i : ' , i );
15   end;
16   writeln('end : ' , i );
17 end.
```

BreakCon.pas

```
Linking a.out
17 lines compiled, 0.0 sec
i : 1
i : 2
end : 10
```

```
1  program BreakContinueExample;
2
3  var
4    i: Integer;
5
6  begin
7    i := 0;
8    while i <> 10  do
9    begin
10     i := i +1;
11     if i >= 4 then
12       continue;
13     writeln('i : ' , i );
14   end;
15   writeln('end : ' , i );
16 end.
17
```

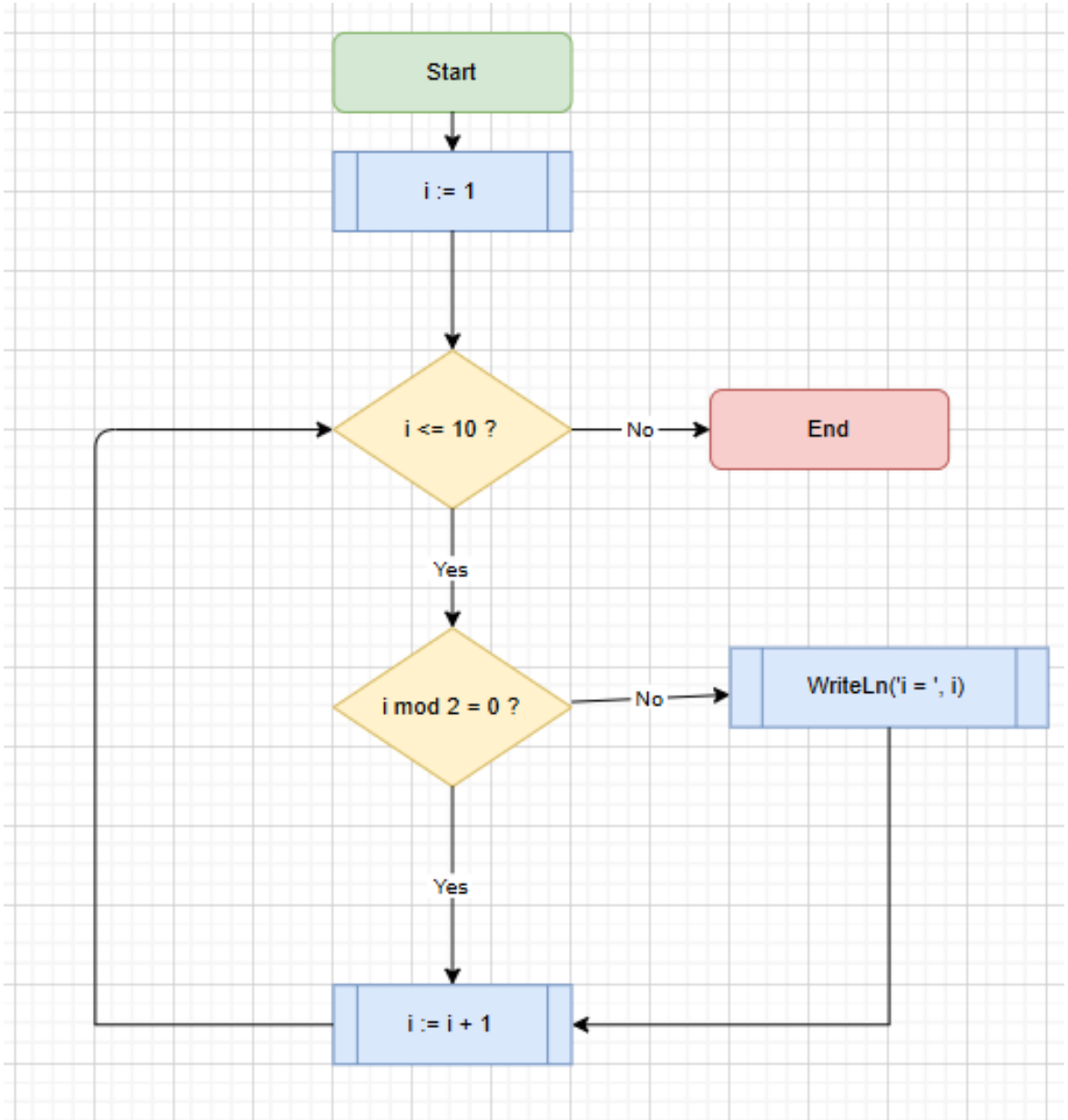
ContinueEx.pas

```
16 lines compiled, 0.0 sec
i : 1
i : 2
i : 3
end : 10
```

NestedForBreak.pas

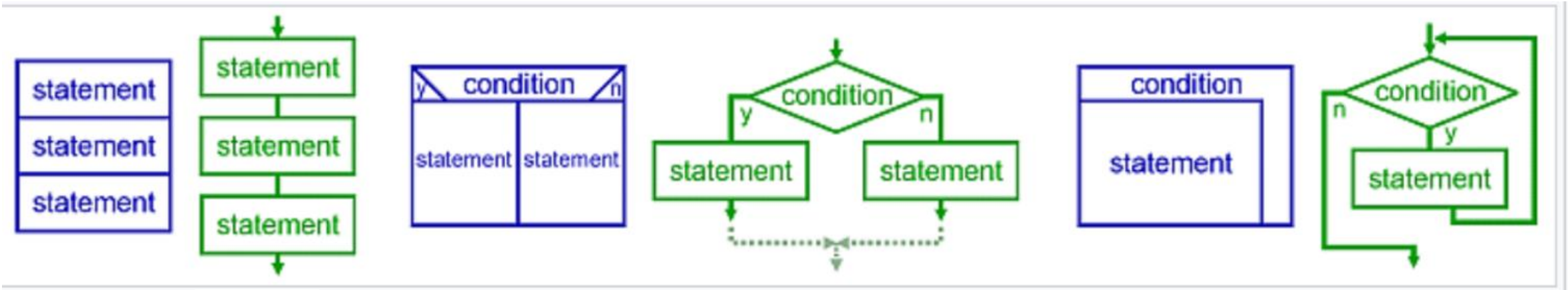
```pascal
program NestedForBreakStructured;
var
  i, j: Integer;

begin
  for i := 1 to 3 do
  begin

    for j := 1 to 4 do
    begin
      if j >= 3 then
      begin
        continue;  // ✗ not standard Pascal, so we'll handle manually
      end;
      WriteLn('inner -> i = ', i, ', j = ', j);
    end;
    WriteLn('outer -> i = ', i, ', j = ', j);
  end;
end.
```

```
Target OS: Linux for x86-64
Compiling main.pas
Linking a.out
25 lines compiled, 0.0 sec
inner -> i = 1, j = 1
inner -> i = 1, j = 2
outer -> i = 1, j = 4
inner -> i = 2, j = 1
inner -> i = 2, j = 2
outer -> i = 2, j = 4
inner -> i = 3, j = 1
inner -> i = 3, j = 2
outer -> i = 3, j = 4
```


Start
i := 1
i <= 10 ?
No
End
Yes
i mod 2 = 0 ?
No
WriteLn('i = ', i)
Yes
i := i + 1

Structure Programming

Structured programming is a programming paradigm aimed at improving the clarity, quality, and development time of a computer program. A code block is structured, as shown in the figure. **In flow-charting condition, a box with a single entry point and single exit point are structured**. Structured programming is a method of making it evident that the program is correct.



sequence

If-then-else

repetition

- NS diagrams (blue) and flow charts (green).

Structure Programming