



ลองบอกหน่อยสิ

# คำสั่งวนซ้ำ

## คืออะไร??????

:

.

(159) Amazing ice cream production process in modern  
factories - YouTube



ลองบอกหน่อยสิ

จากคลิปนักเรียนพบการวนซ้ำ  
เกิดขึ้นในขั้นตอนใดบ้าง  
และคิดว่ามีประโยชน์อย่างไร

# คำสั่งวนซ้ำ

คือ คำสั่งที่ใช้กำหนดให้เกิดการทำงานซ้ำ ณ ส่วนใดส่วนหนึ่งของโปรแกรมหลายๆ ครั้ง ตาม **เงื่อนไข** ที่กำหนด

คำสั่งควบคุม แบบวนซ้ำ  
อาจเรียกว่า **คำสั่งลูป(Loop)**

# คำสั่งวนซ้ำ

**while**

**do-while**

**for**

# ทำไมต้องใช้คำสั่ง ควบคุมการทำงานแบบวนซ้ำ?

- เหตุการณ์ที่เกิดขึ้นหลายรอบ เช่น
  - โปรแกรมแสดง ชื่อ 20 ครั้ง
- เหตุการณ์ที่เกิดขึ้นหลายรอบ โดยมีการเปลี่ยนแปลงค่าหรือมีเงื่อนไข เช่น
  - แสดงผลเลข 0,1,2,...,10
  - แสดงผลรวมของ 1,2,3,...,99
  - แสดงชื่อไปเรื่อยๆ จนกว่าตัวแปร  $x$  จะมีค่ามากกว่า 30

## ลองพิจารณาคำสั่งดังต่อไปนี้

```
#include<stdio.h>
int main()
{
    printf("1\t");
    printf("2\t");
    printf("3\t");
    printf("4\t");
    printf("5\t");
    ...
    ...
    printf("10\t");
    return 0;
}
```

```
#include<stdio.h>
int main()
{
    int count = 1;
    printf("%d\t",count++);
    printf("%d\t",count++);
    printf("%d\t",count++);
    printf("%d\t",count++);
    printf("%d\t",count++);
    ...
    ...
    printf("%d\t",count++);
    return 0;
}
```

### Input-Output

1 2 3 4 5 6 7 8 9 10

# ลองพิจารณาคำสั่งดังต่อไปนี้

```
#include<stdio.h>
int main()
{
    int count = 1;
    printf("%d\t",count++);
    printf("%d\t",count++);
    printf("%d\t",count++);
    printf("%d\t",count++);
    printf("%d\t",count++);
    ...
    ...
    printf("%d\t",count++);
    return 0;
}
```

```
#include<stdio.h>
int main()
{
    int count = 1;
    while (count <= 10) {
        printf("%d\t",count++);
    }
    return 0;
}
```

## Input-Output

1 2 3 4 5 6 7 8 9 10



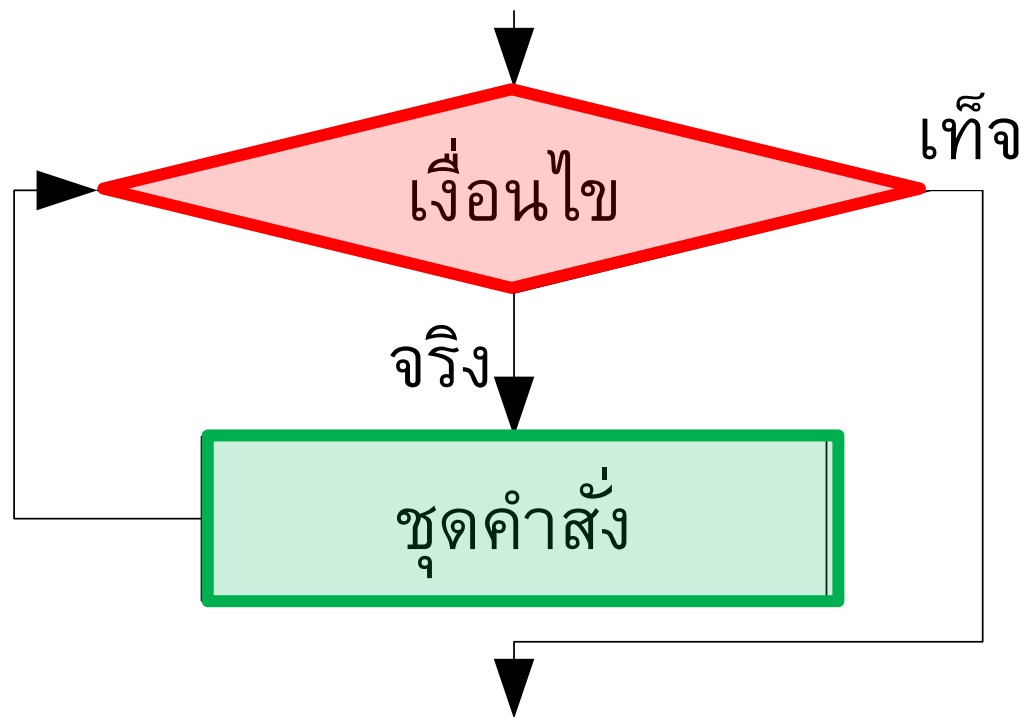
# 1) คำสั่ง while



# การทำงานของคำสั่ง while

```
while( เงื่อนไข ){  
    ชุดคำสั่ง;  
}
```

1. ตรวจสอบ**เงื่อนไข**
2. เมื่อเงื่อนไขเป็น**จริง** ทำ**ชุดคำสั่ง**ซ้ำ
3. เมื่อเงื่อนไขเป็น**เท็จ** ออกจากการวนซ้ำ



# การทำงานของคำสั่ง while

## รูปแบบคำสั่ง While

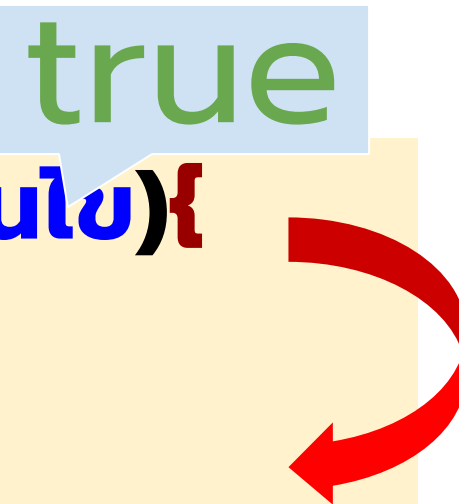
```
while (เงื่อนไข) {  
    คำสั่ง ;  
}
```

**เงื่อนไข** หมายถึง นิพจน์ทางตรรกศาสตร์ ที่ใช้ควบคุมการทำงานของคำสั่ง while

**ถ้าเงื่อนไขเป็นจริง**(ค่าของนิพจน์เป็น **true**) จะเข้าไปทำคำสั่งภายในลูป while

# รูปแบบคำสั่ง While

```
while(เงื่อนไข){  
    คำสั่ง;  
}
```

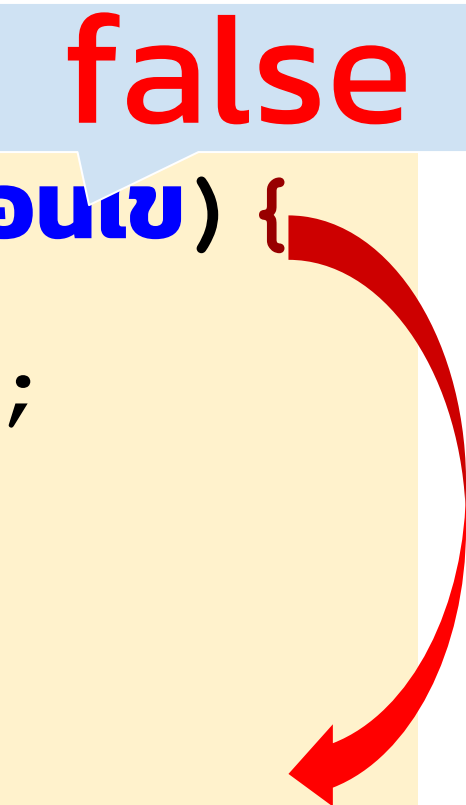
A diagram illustrating the 'while' loop structure. It shows the code 'while(เงื่อนไข){ คำสั่ง; }' on a yellow background. A blue box with the word 'true' in green text is positioned above the opening curly brace. A red curved arrow points from the closing curly brace back to the opening curly brace, indicating the loop's repetition.

**เงื่อนไข** หมายถึง นิพจน์ทางตรรกศาสตร์ ที่ใช้ควบคุมการทำงานของคำสั่ง 'while'

ถ้าเงื่อนไขเป็น**จริง**(ค่าของนิพจน์เป็น **true**) จะเข้าไปทำคำสั่งภายในลูป **while** แล้ววนกลับมาเช็คเงื่อนไขอีกครั้ง

# รูปแบบคำสั่ง While

```
while (เงื่อนไข) {  
    คำสั่ง ;  
}
```



**เงื่อนไข** หมายถึง นิพจน์ทางตรรกศาสตร์ ที่ใช้ควบคุมการทำงานของคำสั่ง while

ถ้าเงื่อนไขเป็น**เท็จ** (ค่าของนิพจน์เป็น **false**) จะไม่ทำคำสั่งภายในลูป while

# ตัวดำเนินการเชิงสัมพันธ์ (Relational Operators)

เครื่องหมาย	ความหมาย	ตัวอย่างการใช้
>	มากกว่า	$A > B$
<	น้อยกว่า	$A < B$
>=	มากกว่าหรือเท่ากับ	$A >= B$
<=	น้อยกว่าหรือเท่ากับ	$A <= B$
==	เท่ากับ	$A == B$
!=	ไม่เท่ากับ	$A != B$

# ตัวดำเนินการทางตรรกะ (Logical Operator)

เครื่องหมาย	ความหมาย	ตัวอย่างการใช้
&&	AND (และ)	A && B
	OR (หรือ)	A    B
!	NOT (นิเสธ)	!A



# ตัวดำเนินการทางตรรกะ (Logical Operator)

ตัวแปร		ผลลัพธ์		
A	B	A && B	A    B	!A
1	1	1	1	0
1	0	0	1	0
0	1	0	1	1
0	0	0	0	1





# ตัวดำเนินการทางตรรกะ (Logical Operator)



**!( 8 < 10 )   ||   !( 10 == 8 )**

**!( 8 < 10 )**

**!( 10 == 8 )**

!True

||

!False

False

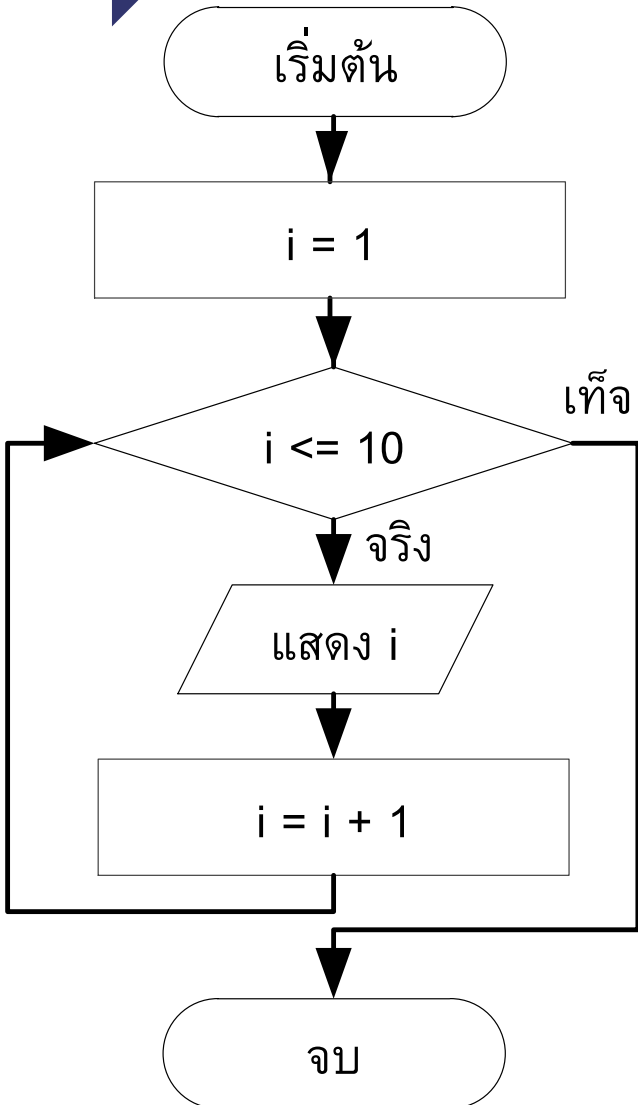
||

True



True

## ตัวอย่าง จงบอกผลลัพธ์ของโปรแกรมต่อไปนี้

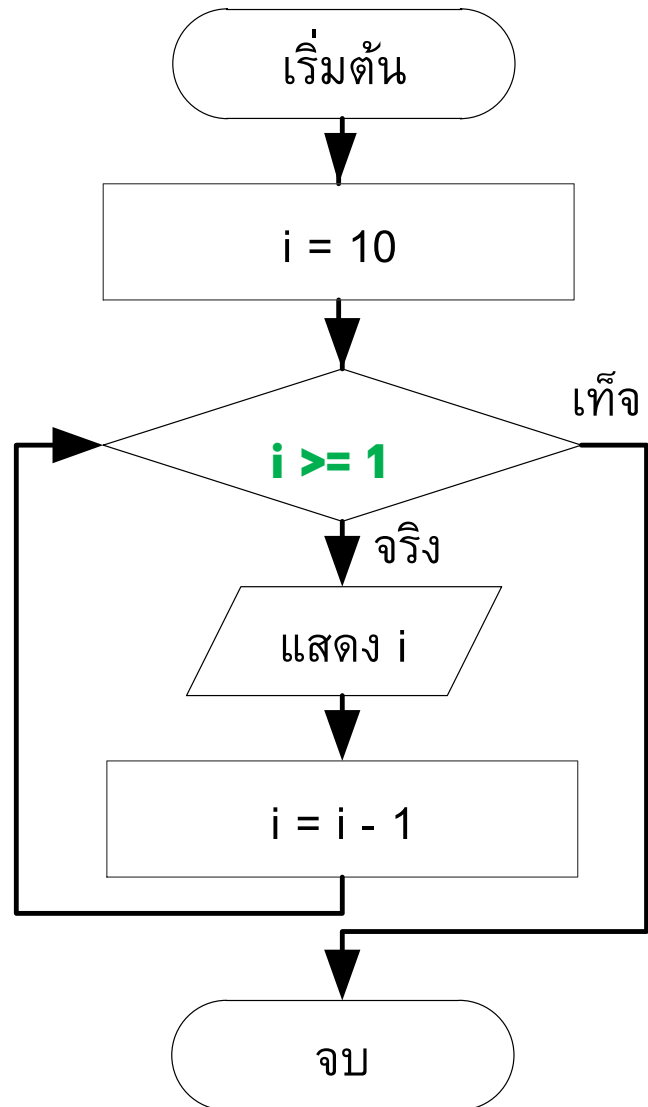


```
#include<stdio.h>
int main() {
    int i = 1;
    while (i<=10) {
        printf("%d\t", i);
        i = i + 1;
    }
    return 0;
}
```

### Input-Output

1 2 3 4 5 6 7 8 9 10

## ตัวอย่าง จงเขียนโปรแกรมแสดงตัวเลข 10-1



```
#include<stdio.h>
int main() {
    int i=10;
    while (i>=1) {
        printf("%d\t",i) ;
        i=i-1;
    }
    return 0;
}
```

### Input-Output

10 9 8 7 6 5 4 3 2 1

# จากโปรแกรมต่อไปนี้ เกิดอะไรขึ้น?

```
#include<stdio.h>
int main() {
    int i=10;
    while(i>=1) {
        printf("%d\t",i);
    }
    return 0;
}
```

```
#include<stdio.h>
int main() {
    int i=10;
    while(i>=1) {
        printf("%d\t",i);
        i=i-1;
    }
    return 0;
}
```

**Infinite loop**

# ตัวอย่าง จงเขียนโปรแกรมแสดงตัวเลข 5 , 10 , 15 ,..., 100

```
#include<stdio.h>
int main() {
    int i=5;
    while (i<=100) {
        printf ("%d\n", i) ;
        i=i+5;
    }
    return 0;
}
```

ตัวอย่าง จงเขียนโปรแกรมหาผลบวกของตัวเลข 5 , 10 , 15 , ..., 100

```
#include<stdio.h>
int main(){
    int i=5;
    int sum = 0;
    while (i<=100){
        sum = sum+i;
        i=i+5;
    }
    printf ("%d\t",sum) ;
    return 0;
}
```

Input-Output

1050

## ตัวอย่าง จงเขียนโปรแกรมหาผลบวกของตัวเลข 1-100

```
#include<stdio.h>
int main() {
    int i=1;
    int sum = 0;
    while (i<=100) {
        sum = sum+i;
        i=i+1;
    }
    printf ("%d\t",sum) ;
    return 0;
}
```

Input-Output

5050

# ไม่ใส่ปีกกาหลังคำสั่ง while ได้หรือไม่?

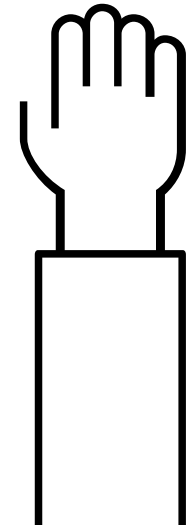
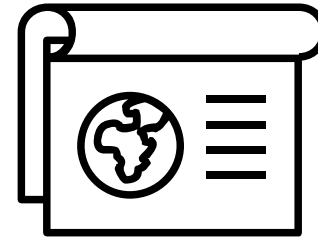
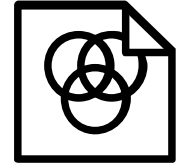
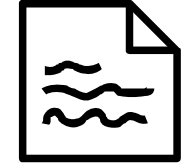
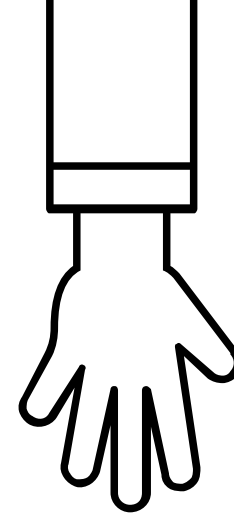
```
#include<stdio.h>
int main() {
    int i=1;
    int sum = 0;
    while(i<=100)
        sum = sum+i;
        i=i+1;
    printf("%d\t", sum);
    return 0;
}
```



```
#include<stdio.h>
int main() {
    int i=1;
    int sum = 0;
    while(i<=100)
        sum = sum+i++;
    printf("%d\t", sum);
    return 0;
}
```







# การเขียนโปรแกรม แบบวนซ้ำกับ ตัวอักษร (character)



**ตัวอย่าง** จงเขียนโปรแกรมรับค่าตัวอักษร  
ภาษาอังกฤษพิมพ์ใหญ่หนึ่งตัว จากนั้นให้พิมพ์  
ตัวอักษรภาษาอังกฤษตั้งแต่ A จนถึงตัวที่รับเข้ามา

### Input-Output

Please Enter Character: D

A B C D

### Input-Output

Please Enter Character: O

A B C D E F G H I J K L M N O



**ตัวอย่าง** จงเขียนโปรแกรมรับค่าตัวอักษรภาษาอังกฤษพิมพ์ใหญ่  
หนึ่งตัว จากนั้นให้พิมพ์ตัวอักษรภาษาอังกฤษตั้งแต่ A จนถึงตัวที่  
รับเข้ามา

```
#include <stdio.h>
int main()
{
    int c = 65;
    int lastc;
    printf("Please Enter Character : ");
    scanf("%c",&lastc);
    while (c<=lastc) {
        printf("%c ",c);
        c++;
    }
    return 0;
}
```

**ตัวอย่าง** จงเขียนโปรแกรมรับค่าตัวอักษรภาษาอังกฤษพิมพ์ใหญ่  
หนึ่งตัว จากนั้นให้พิมพ์ตัวอักษรภาษาอังกฤษตั้งแต่ A จนถึงตัวที่  
รับเข้ามา

```
#include <stdio.h>
int main()
{
    char c = 'A';
    char lastc;
    printf("Please Enter Character : ");
    scanf("%c",&lastc);
    while(c<=lastc) {
        printf("%c ",c);
        c++;
    }
    return 0;
}
```

# การเขียนโปรแกรม แบบวนซ้ำและ กำหนดเงื่อนไข



# การเขียนโปรแกรมแบบวนซ้ำ และกำหนดเงื่อนไข

การเขียนโปรแกรมที่มีความซับซ้อนมากขึ้นจำเป็นต้องอาศัยการเขียนโปรแกรมแบบวนซ้ำร่วมกับการเขียนโปรแกรมแบบมีเงื่อนไข

โดยการเพิ่มเงื่อนไขการทำงานในส่วนของการวนรอบ หรือมีการตรวจสอบเงื่อนไขว่าจะให้โปรแกรมมีการวนรอบอย่างไร



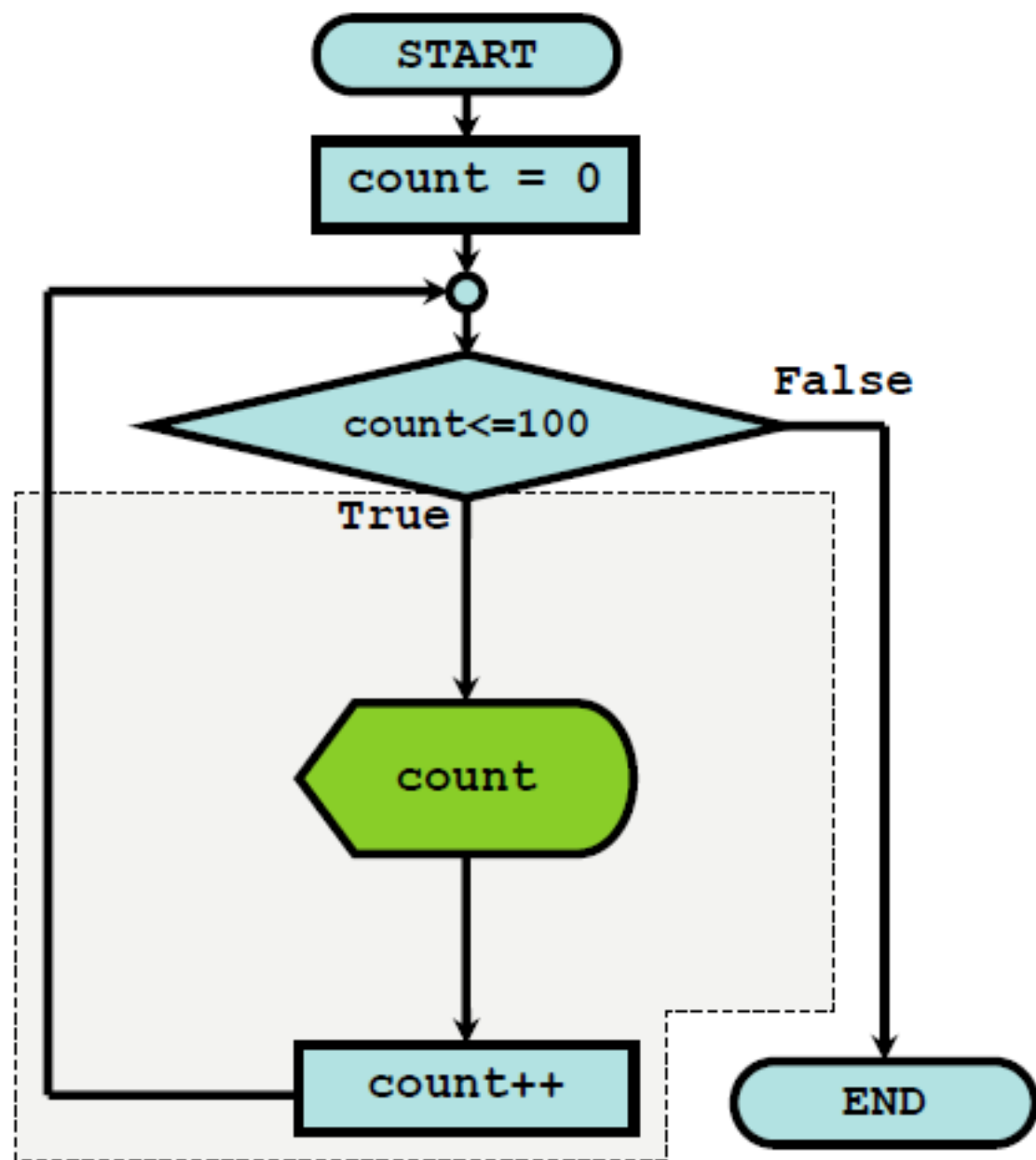
## ตัวอย่าง จงเขียนโปรแกรมแสดงเลข 0-100 โดยใช้ while

```
#include <stdio.h>
int main() {
    int count=0;
    while (count<=100) {
        printf("%d ", count++);
    }
    return 0;
}
```

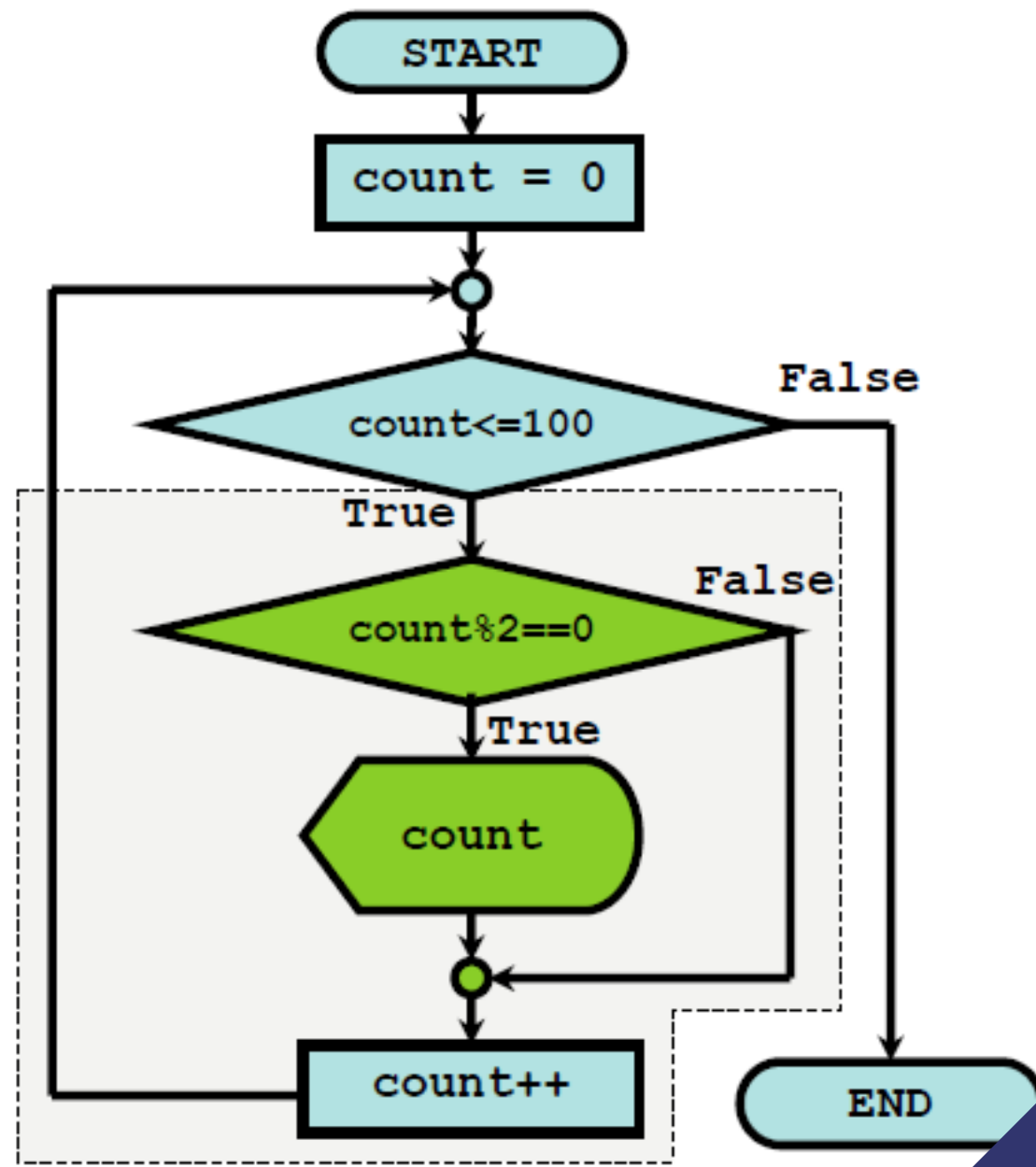
## ตัวอย่าง จงเขียนโปรแกรมแสดงเลข 0-100 เฉพาะเลขคู่โดยใช้ while

```
#include <stdio.h>
int main() {
    int count=0;
    while (count<=100) {
        if (count%2==0)
            printf("%d ", count);
        count++;
    }
    return 0;
}
```

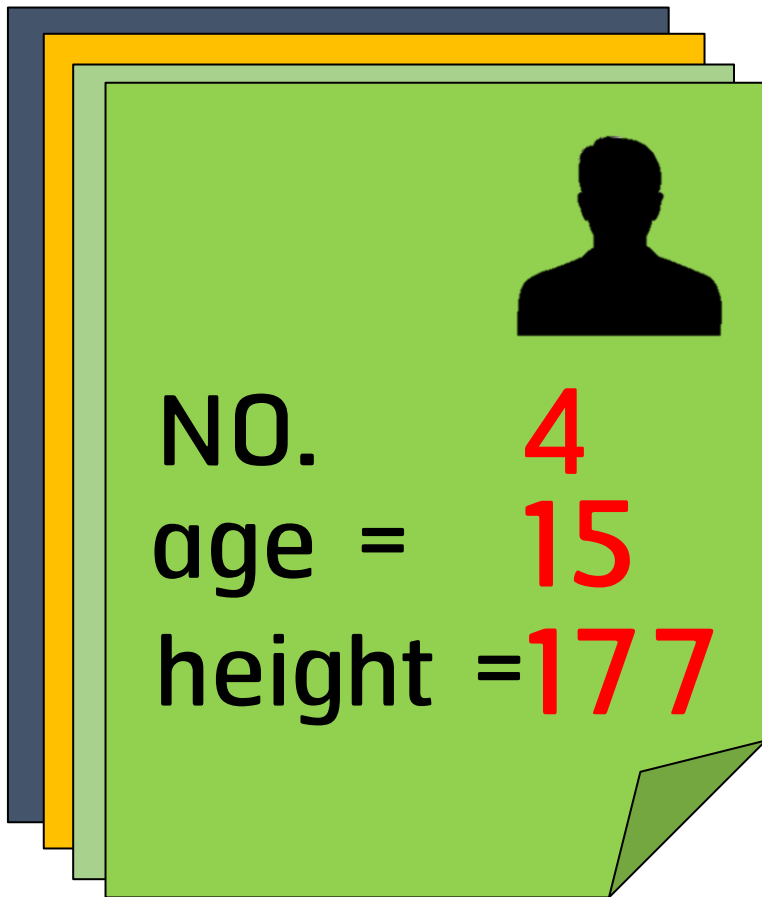
(แสดงเลข 0 – 100)



(แสดงเลขคู่ 0 – 100)

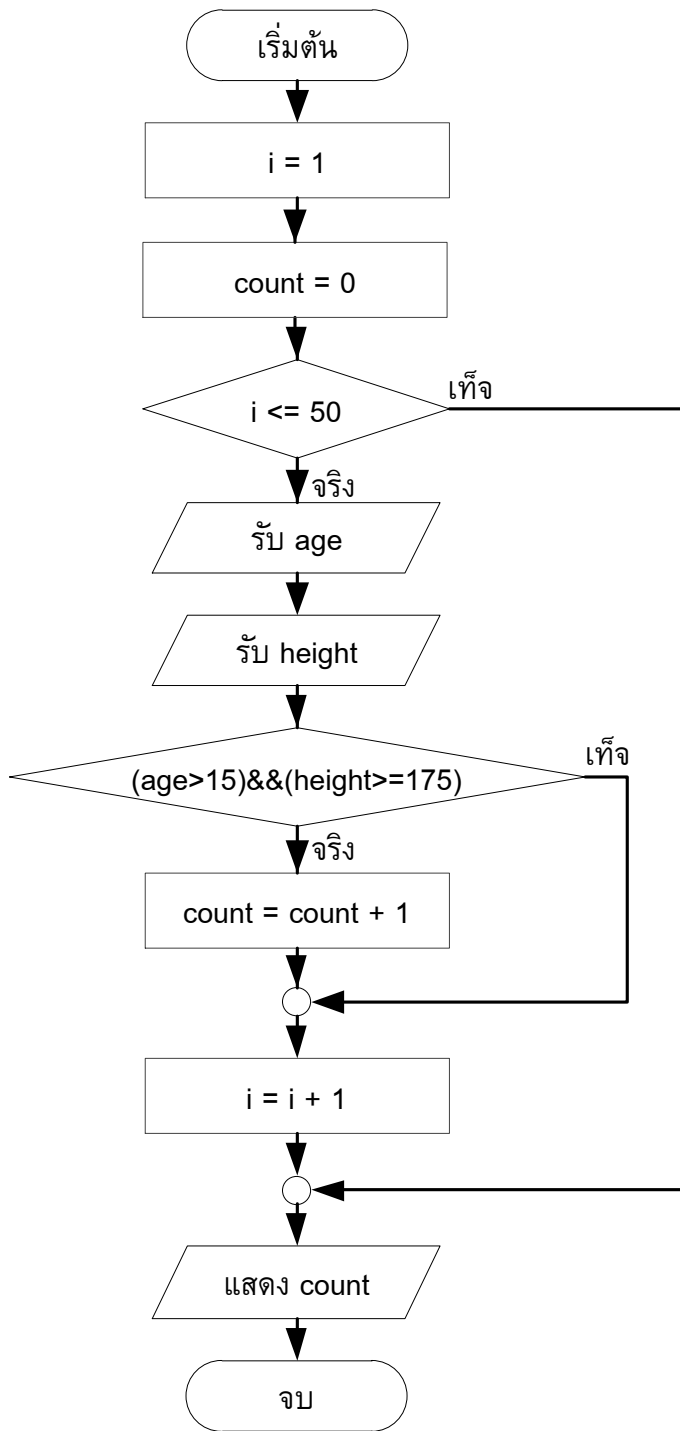


ตัวอย่าง จงเขียนโปรแกรมรับค่าความสูงและอายุของพนักงาน  
บริษัททั้งหมด 50 คน แล้วให้นับว่าผู้ที่มีอายุมากกว่า 15 ปี และ  
มีความสูงตั้งแต่ 175 cm มีทั้งหมดกี่คน



count

0



```
#include<stdio.h>
int main(){
    int i=1;
    int count=0;
    int age;
    int height;
    while (i<=50) {
        printf("Enter age:");
        scanf ("%d", &age);
        printf("Enter height:");
        scanf ("%d", &height);
        if ( (age>15) && (height>=175) ) {
            count=count+1;
        }
        i++;
    }
    printf ("count=%d", count);
    return 0;
}
```

ตัวอย่าง จงเขียนโปรแกรมรับค่าตัวเลข  
จำนวน 5 ตัว แล้วหาค่าสูงสุด

Input-Output

5

17

55

150

12

The largest number is 150

## ตัวอย่าง จงเขียนโปรแกรมรับค่าตัวเลขจำนวนเต็ม บวก 5 ตัว แล้วหาค่าสูงสุด

```
#include <stdio.h>
int main()
{
    int num,i=0,max=0;
    while (i<5) {
        scanf ("%d" , &num) ;
        if (max<num)
            max=num;
        i++;
    }
    printf("The largest number is %d",max) ;
    return 0;
}
```

# คำสั่ง break

- คือคำสั่งที่ใช้ในการออกจากวนลูป

```
#include <stdio.h>
int main() {
    int i=0;
    while(i<10) {
        i++;
        if(i==5)
            break;
        printf("%d ", i);
    }
    printf("\ni'm done ");
    return 0;
}
```

## Input-Output

```
1 2 3 4
i'm done
```



# คำสั่ง return

- คือคำสั่งที่ใช้ในการออกจากฟังก์ชันหรือโปรแกรม

```
#include <stdio.h>
int main() {
    int i=0;
    while(i<10) {
        i++;
        if(i==5)
            return 0;
        printf("%d ", i);
    }
    printf("\nI'm done ");
    return 0;
}
```

## Input-Output

1 2 3 4

# คำสั่ง continue

- คือคำสั่งที่กลับไปตรวจสอบเงื่อนไขของลูปอีกครั้ง ถ้าเงื่อนไขเป็นจริงก็ดำเนินการทำคำสั่งในลูปต่อไป

```
#include <stdio.h>
int main(){
    int i=0;
    while(i<10){
        i++;
        if(i==5)
            continue;
        printf("%d",i);
    }
    return 0;
}
```

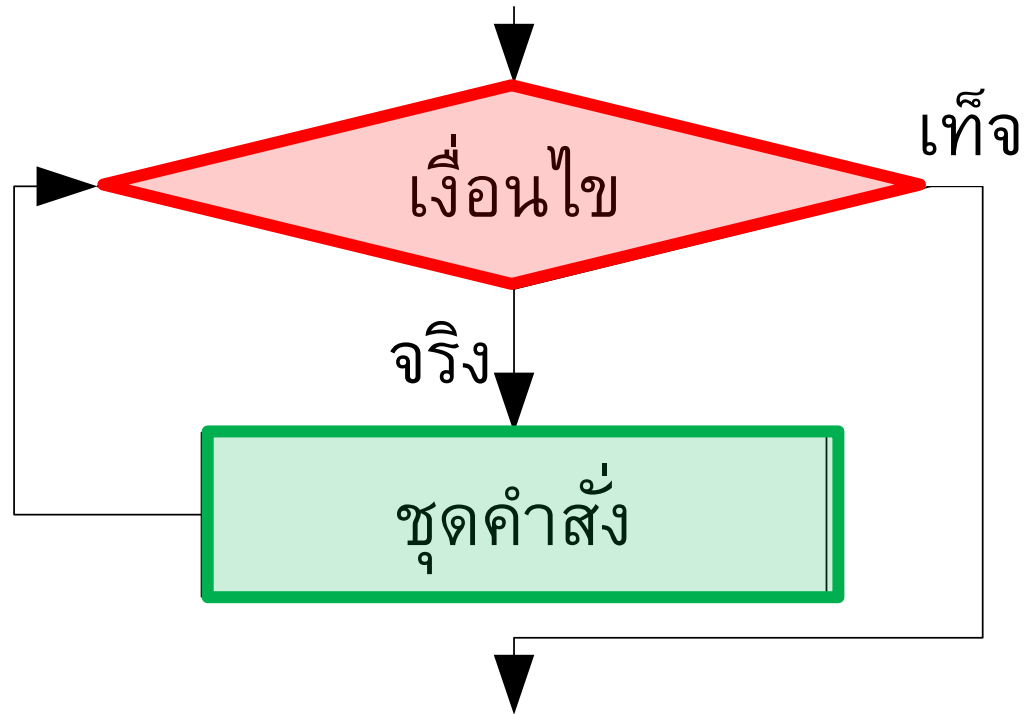
## Input-Output

1 2 3 4 6 7 8 9 10

# สรุปการทำงานของคำสั่ง while

```
while( เงื่อนไข ){  
    ชุดคำสั่ง;  
}
```

1. ตรวจสอบ**เงื่อนไข**
2. เมื่อเงื่อนไขเป็น**จริง** ทำ**ชุดคำสั่ง**ซ้ำ
3. เมื่อเงื่อนไขเป็น**เท็จ** ออกจากการวนซ้ำ



**พักผ่อนน้ำ  
เข้าห้องน้ำ  
ปิดร่างกาย**

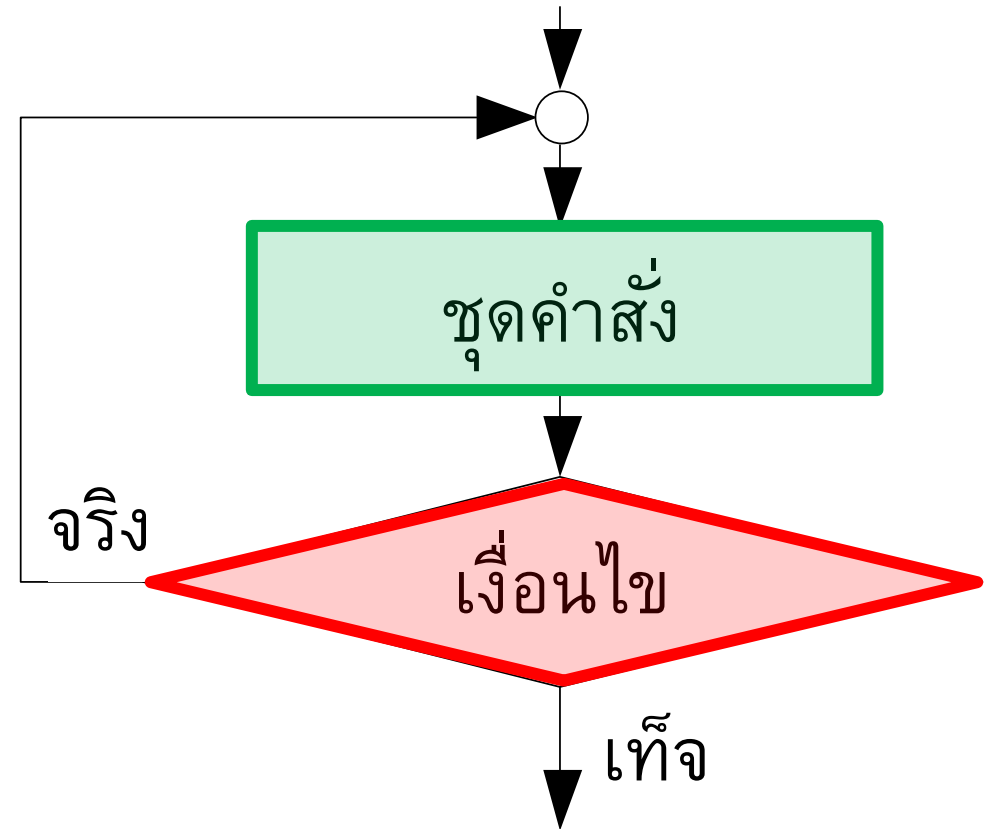
**เริ่ม 10.35**

## 2. คำสั่ง do-while



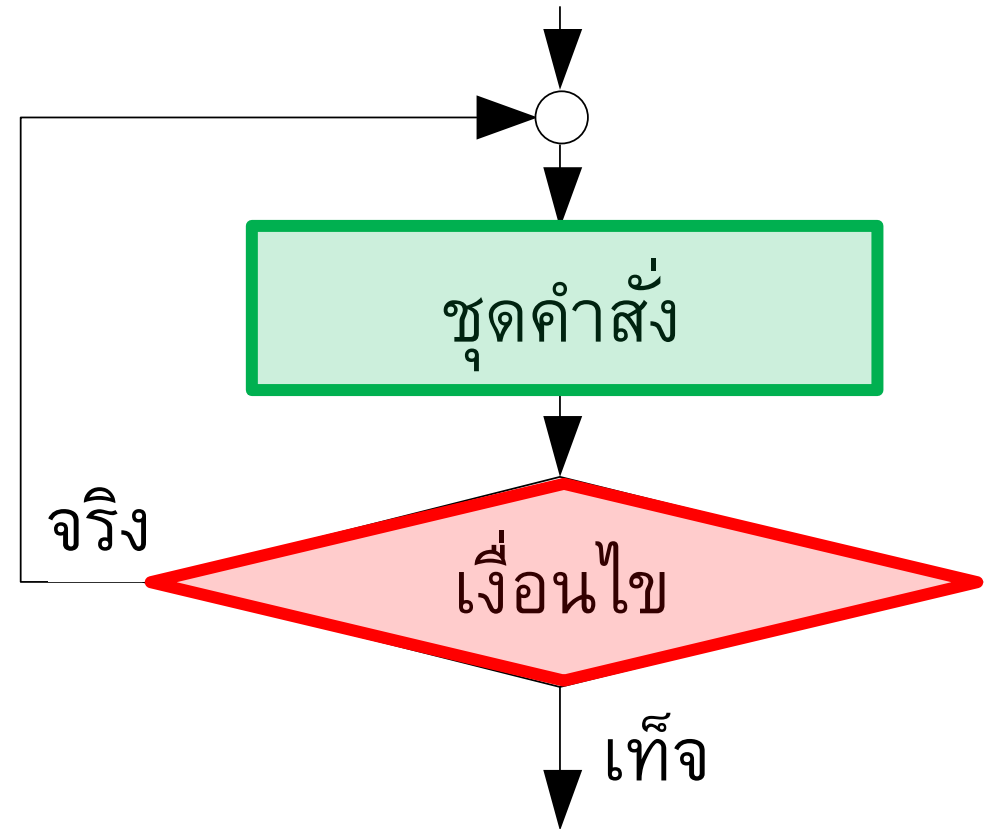
# การทำงานของคำสั่ง do-while

```
do{  
    ชุดคำสั่ง;  
}  
while( เงื่อนไข );
```



# การทำงานของคำสั่ง do-while

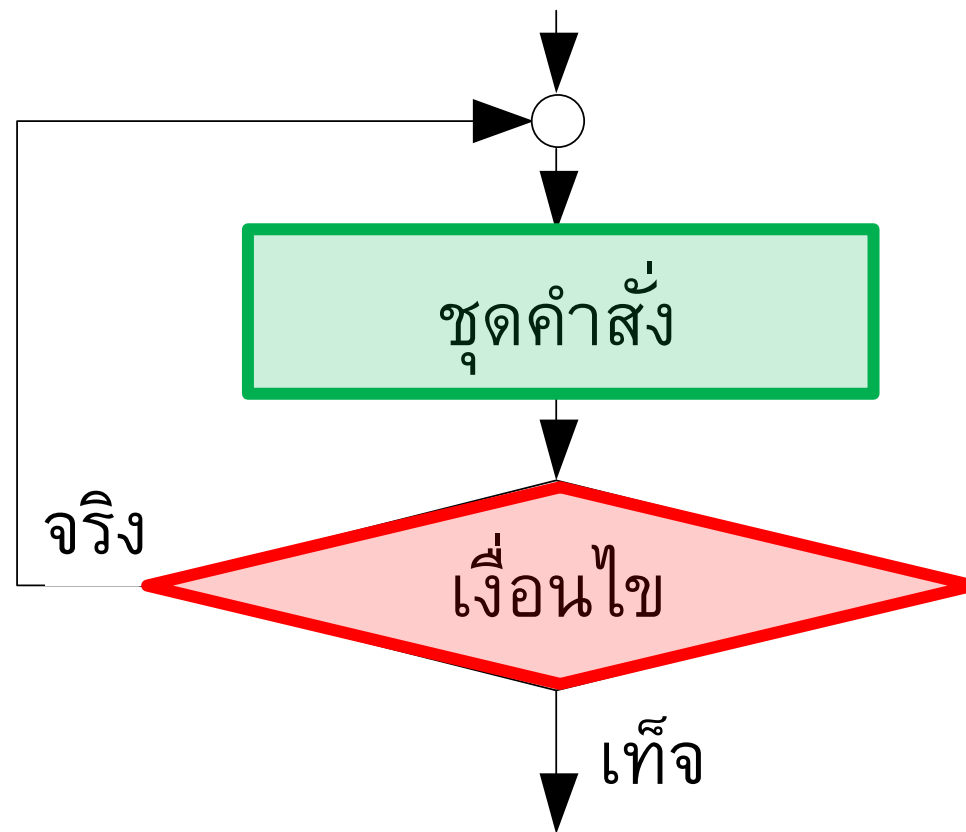
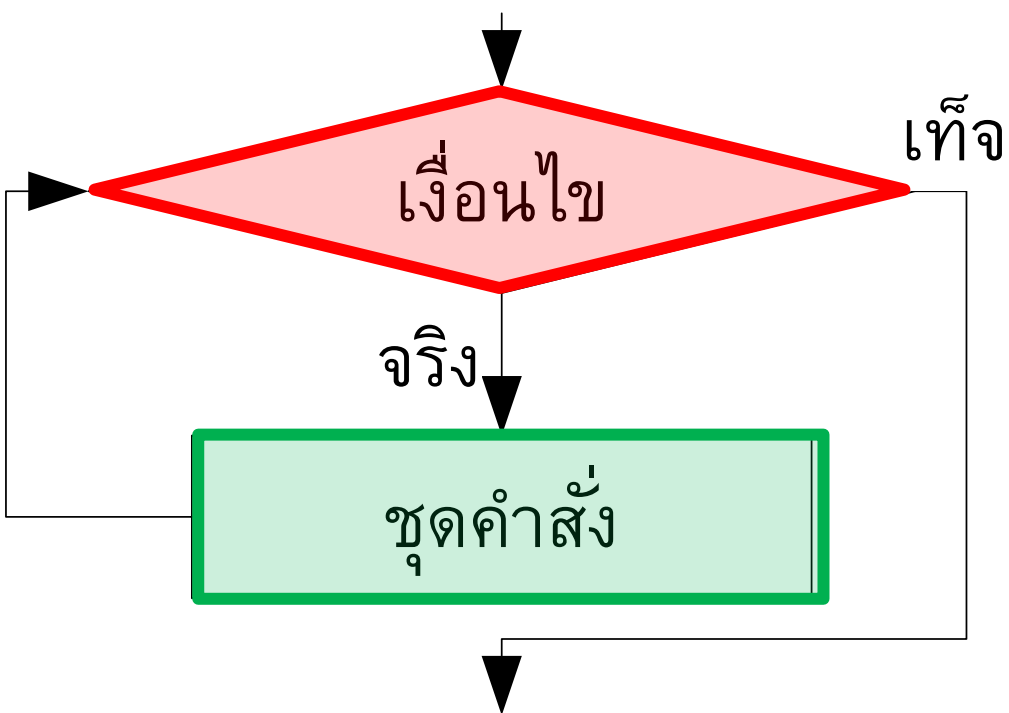
```
do{  
    ชุดคำสั่ง;  
}  
while( เงื่อนไข );
```



## สรุป

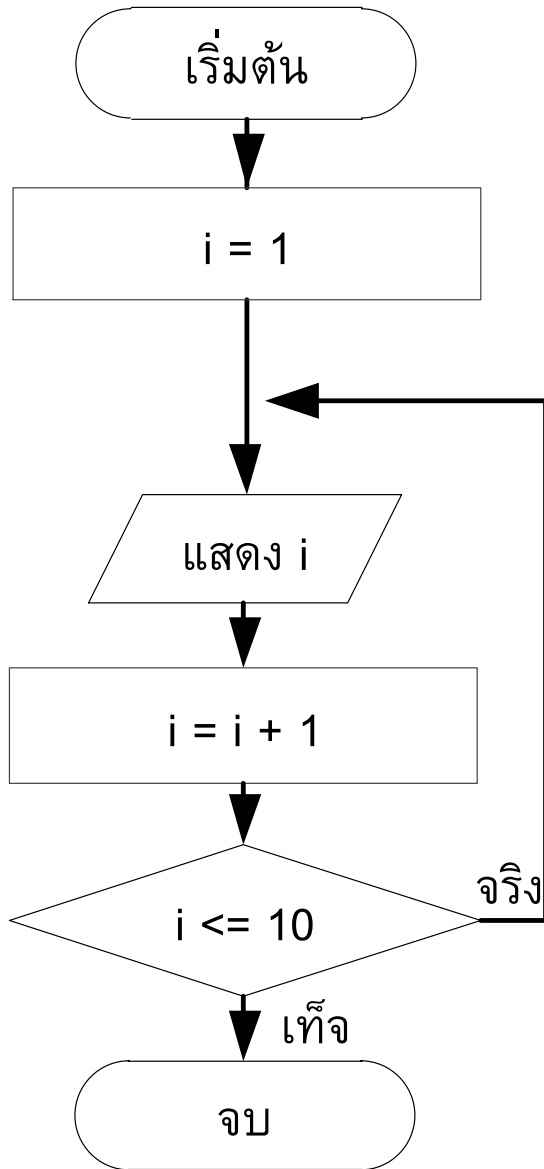
1. ทำชุดคำสั่งอย่างน้อย 1 ครั้ง
2. ตรวจสอบเงื่อนไข
  - เมื่อเงื่อนไขเป็นจริง กลับไปทำชุดคำสั่ง
  - เมื่อเงื่อนไขเป็นเท็จ ออกจากการวนซ้ำ

# เปรียบเทียบ while และ do-while





# ตัวอย่าง จงเขียนโปรแกรมแสดงตัวเลข 1-10



```
#include<stdio.h>
int main() {
    int i=1;
    do{
        printf("%d ",i);
        i=i+1;
    }
    while(i<=10);
    return 0;
}
```

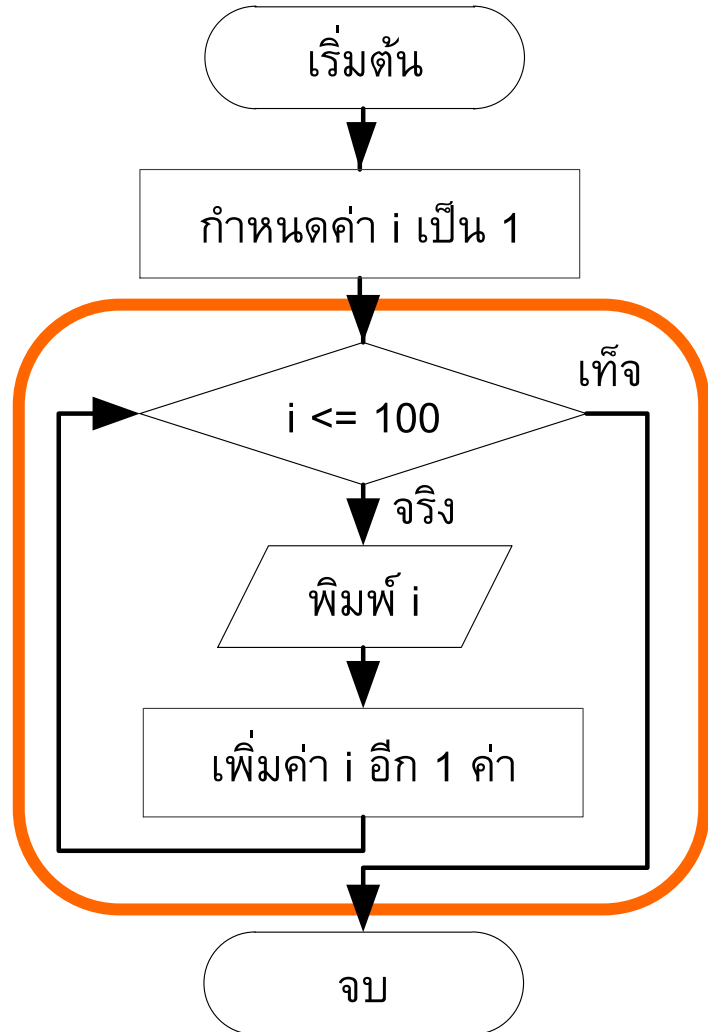
# เปรียบเทียบการทำงานของ while และ do-while

```
#include<stdio.h>
int main(){
    int i = 1; ①
    while (i<=10) ②
    {
        printf("%d ",i);
        i = i + 1; ③
    }
    return 0;
}
```

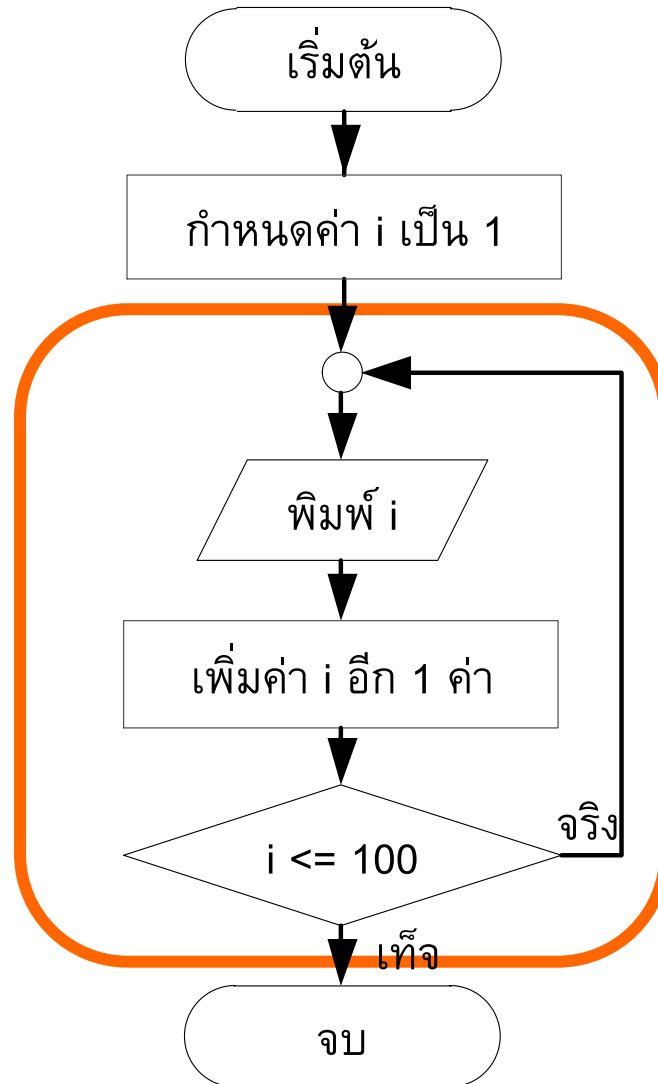
```
#include<stdio.h>
int main(){
    int i=1; ①
    do{
        printf("%d ",i);
        i=i+1; ③
    }
    while (i<=10); ②
    return 0;
}
```

# เปรียบเทียบการทำงานของ while และ do-while

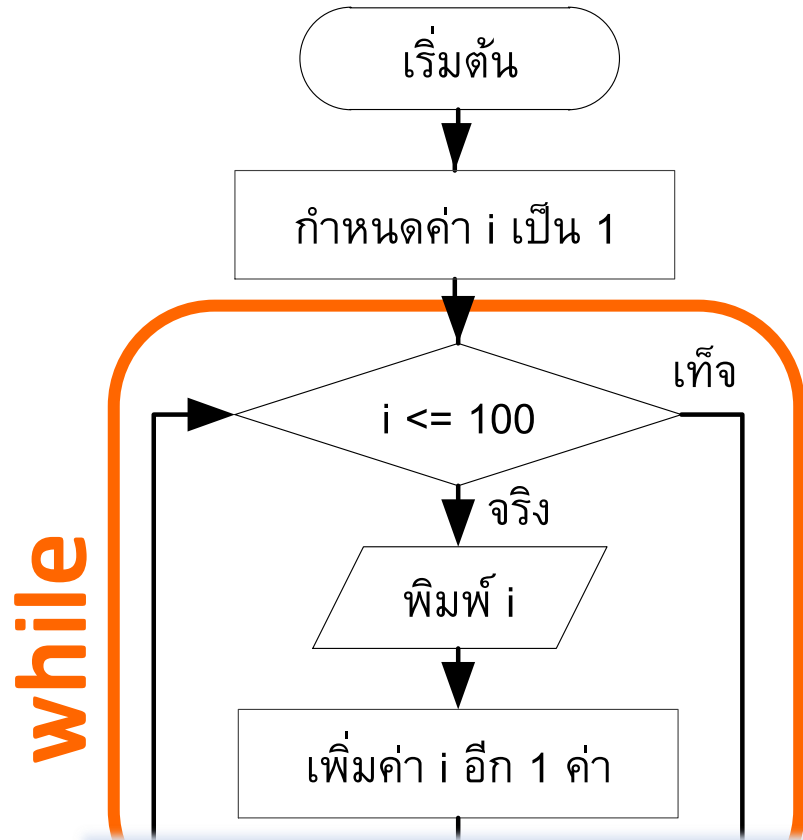
while



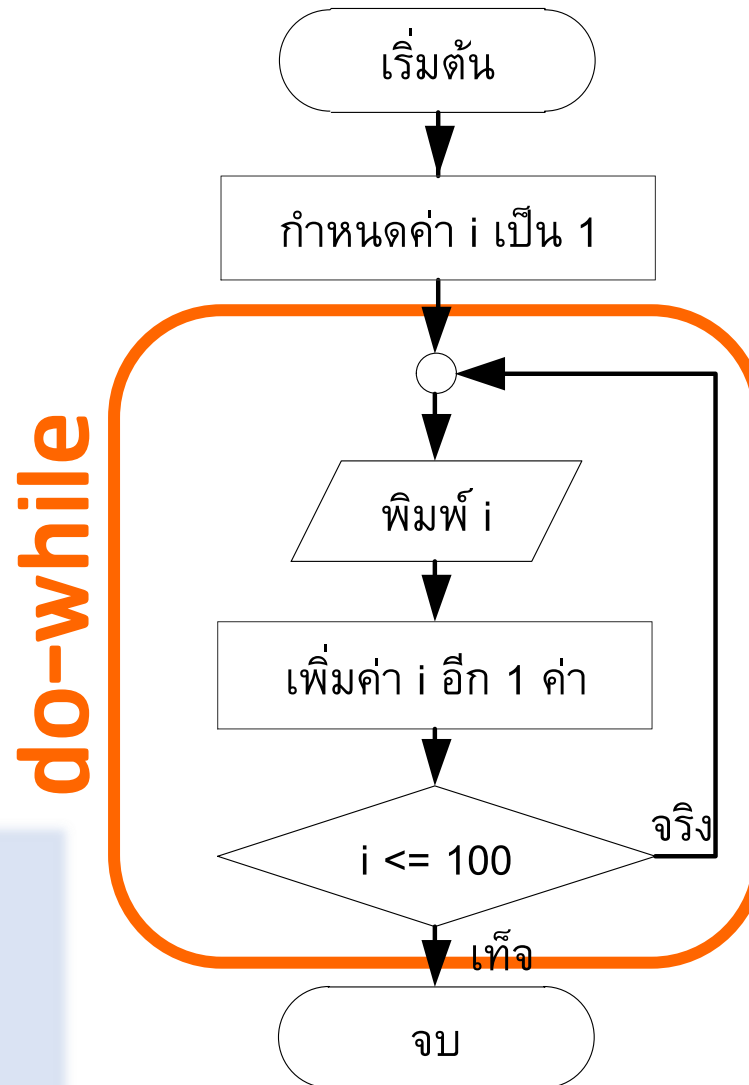
do-while



# เปรียบเทียบการทำงานของ while และ do-while



```
while (i<=10) {  
    printf("%d ",i);  
    i=i+1;  
}
```



```
do{  
    printf("%d ",i);  
    i=i+1;  
}  
while (i<=10);
```

**ตัวอย่าง** จงเขียนโปรแกรมรับจำนวนเต็ม 1 จำนวน แล้วคำนวณหาผลบวกตัวเลขตั้งแต่เลข 1 ถึง จำนวนที่รับมา โดยใช้คำสั่ง while และ do-while ตามลำดับ แสดงผลดังตัวอย่าง

### Input-Output

```
Enter a positive number : 10
```

```
while loop
```

```
Summation from 1 to 10 : 55
```

```
do-while loop
```

```
Summation from 1 to 10 : 55
```

```
#include <stdio.h>
```

```
int main() {
```

```
    int i=1,num,sum=0;
```

```
    printf("Enter a positive number : ");
```

```
    scanf("%d",&num);
```

```
    printf("\nwhile loop\n");
```

```
    while(i<=num) {
```

```
        sum=sum+i++;
```

```
    }
```

```
    printf("Summation from 1 to %d : %d\n",num,sum);
```

```
    //-----
```

```
    i=1;
```

```
    sum=0;
```

```
    printf("\ndo-while loop\n");
```

```
    do{
```

```
        sum=sum+i++;
```

```
    }while(i<=num);
```

```
    printf("Summation from 1 to %d : %d\n",num,sum);
```

```
    return 0;
```

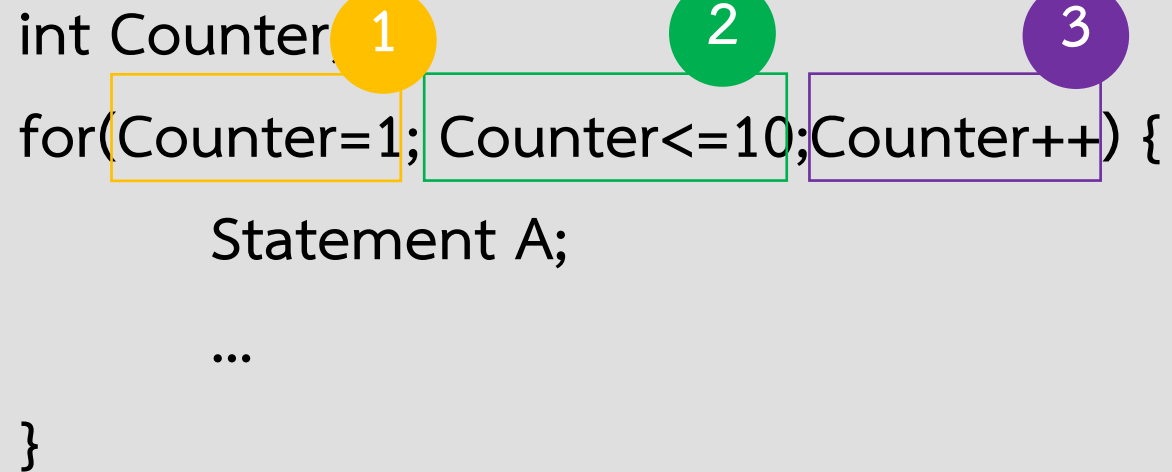
```
}
```

### 3. คำสั่ง for



# คำสั่ง for ประกอบด้วยตัวตรวจสอบเงื่อนไข 3 ส่วน

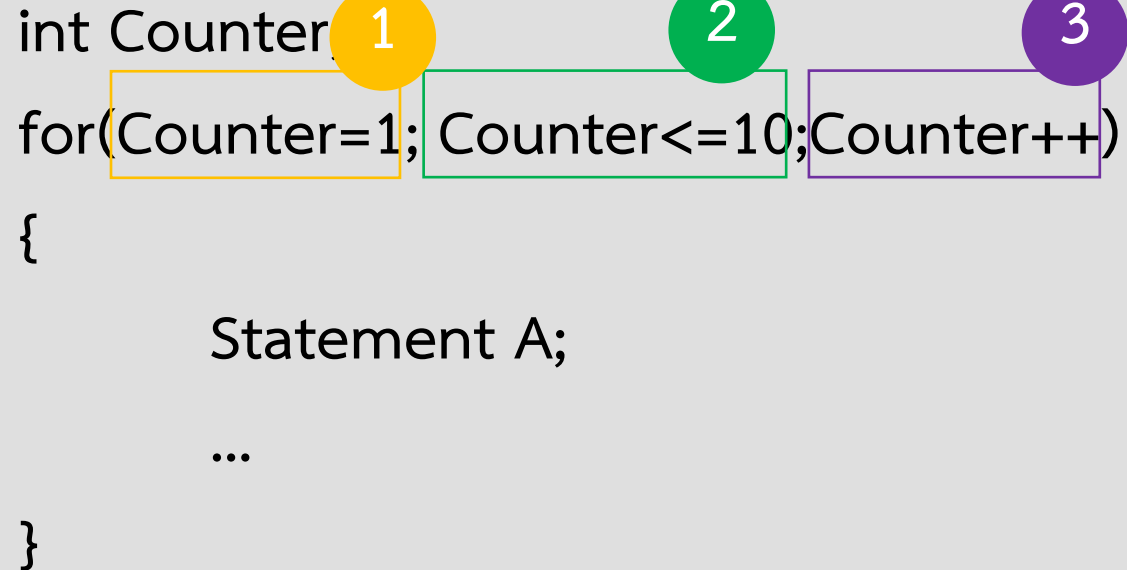
```
int Counter;
for(Counter=1; Counter<=10;Counter++) {
    Statement A;
    ...
}
```





# คำสั่ง for ประกอบด้วยตัวตรวจสอบ เงื่อนไข 3 ส่วน

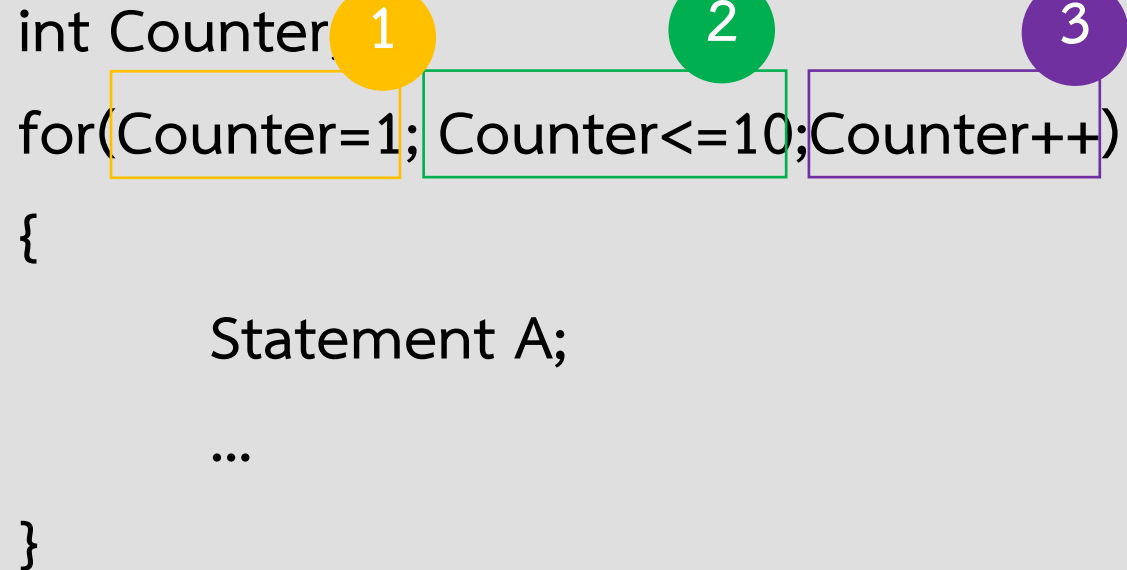
```
int Counter;
for(Counter=1; Counter<=10; Counter++)
{
    Statement A;
    ...
}
```

A diagram showing a C++ for loop. The loop is enclosed in a light gray rounded rectangle with a dashed orange border. The code is: `int Counter;` followed by `for(Counter=1; Counter<=10; Counter++)` on the next line, then a block `{`, `Statement A;`, `...`, and `}`. The three parts of the for loop are highlighted with colored boxes and numbered circles: 1. `Counter=1` is highlighted with an orange box and a yellow circle with the number 1. 2. `Counter<=10` is highlighted with a green box and a green circle with the number 2. 3. `Counter++` is highlighted with a purple box and a purple circle with the number 3.

**1. ส่วนกำหนดค่าเริ่มต้นให้กับตัวแปร** เป็นการกำหนดค่าเริ่มต้นให้กับตัวแปร ใช้เป็นตัวควบคุมเพื่อบอกให้ทราบว่า จะทำซ้ำตั้งแต่ค่าของตัวแปรนี้เป็นค่าอะไร

# คำสั่ง for ประกอบด้วยตัวตรวจสอบเงื่อนไข 3 ส่วน

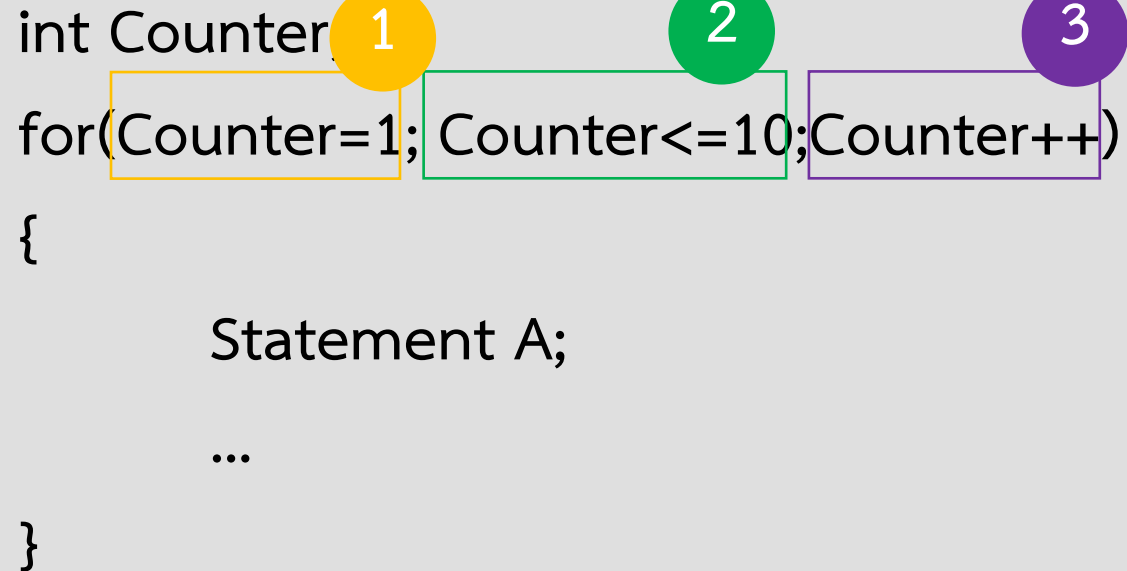
```
int Counter;
for(Counter=1; Counter<=10; Counter++)
{
    Statement A;
    ...
}
```

A diagram showing a C++ for loop. The loop is enclosed in a light gray rounded rectangle with a dashed orange border. The code is: `int Counter;` followed by `for(Counter=1; Counter<=10; Counter++)` on the next line, then an opening curly brace `{`, then `Statement A;` on the next line, then `...` on the next line, and finally a closing curly brace `}`. The three parts of the for loop are highlighted with colored boxes and numbered: 1. `Counter=1` is highlighted with a yellow box and a yellow circle with the number 1. 2. `Counter<=10` is highlighted with a green box and a green circle with the number 2. 3. `Counter++` is highlighted with a purple box and a purple circle with the number 3.

**2. ส่วนเงื่อนไขที่ต้องการตรวจสอบ** เป็นเงื่อนไขในการทำซ้ำของลูป โดยหากเงื่อนไขนี้เป็นจริงก็ยังคงทำงานในลูปต่อไป แต่ถ้าหากเงื่อนไขนี้เป็นเท็จ ก็จะออกจากการทำงานของลูป ไปทำคำสั่งที่อยู่นอกลูปต่อไป

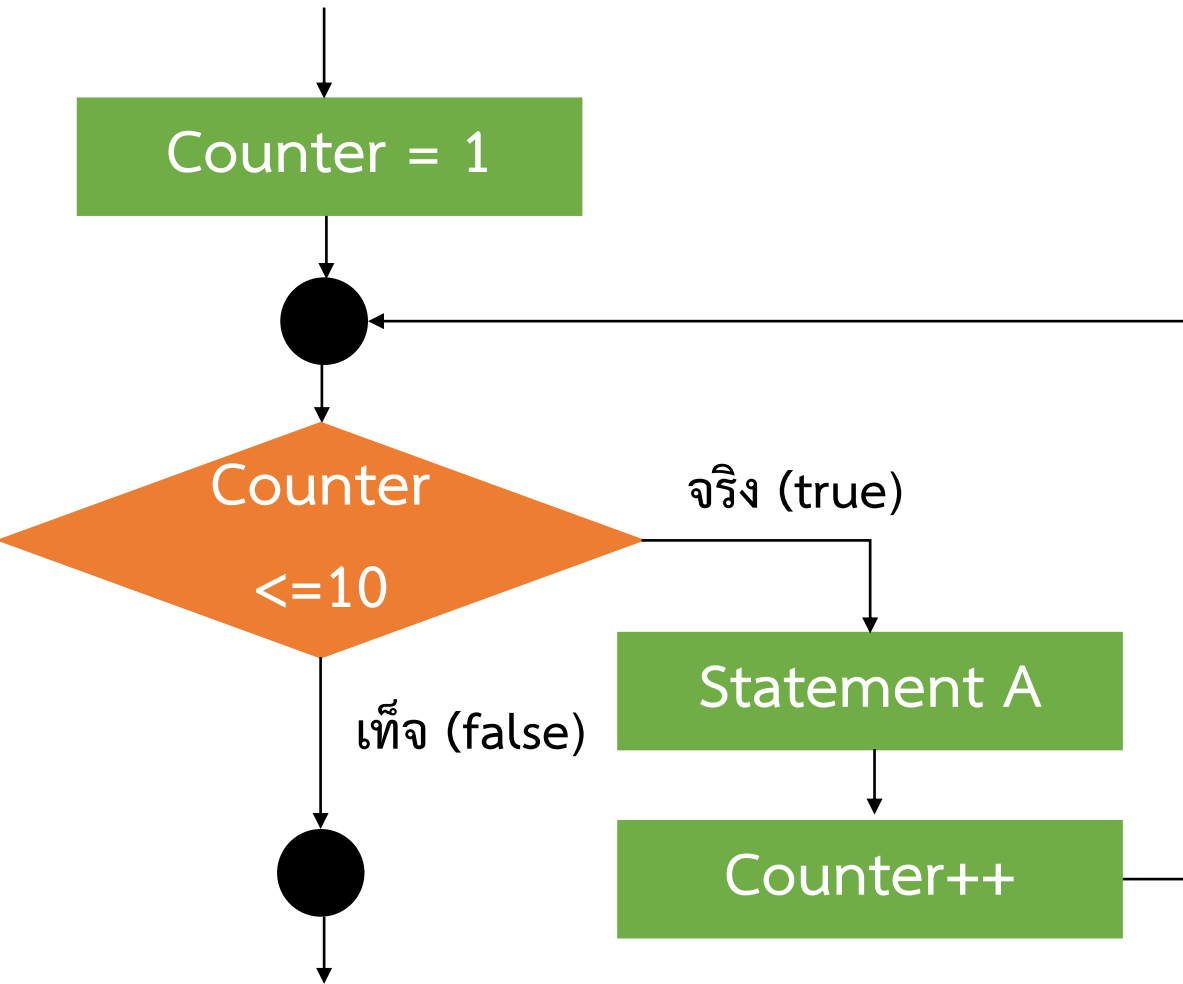
# คำสั่ง for ประกอบด้วยตัวตรวจสอบเงื่อนไข 3 ส่วน

```
int Counter;
for(Counter=1; Counter<=10; Counter++)
{
    Statement A;
    ...
}
```



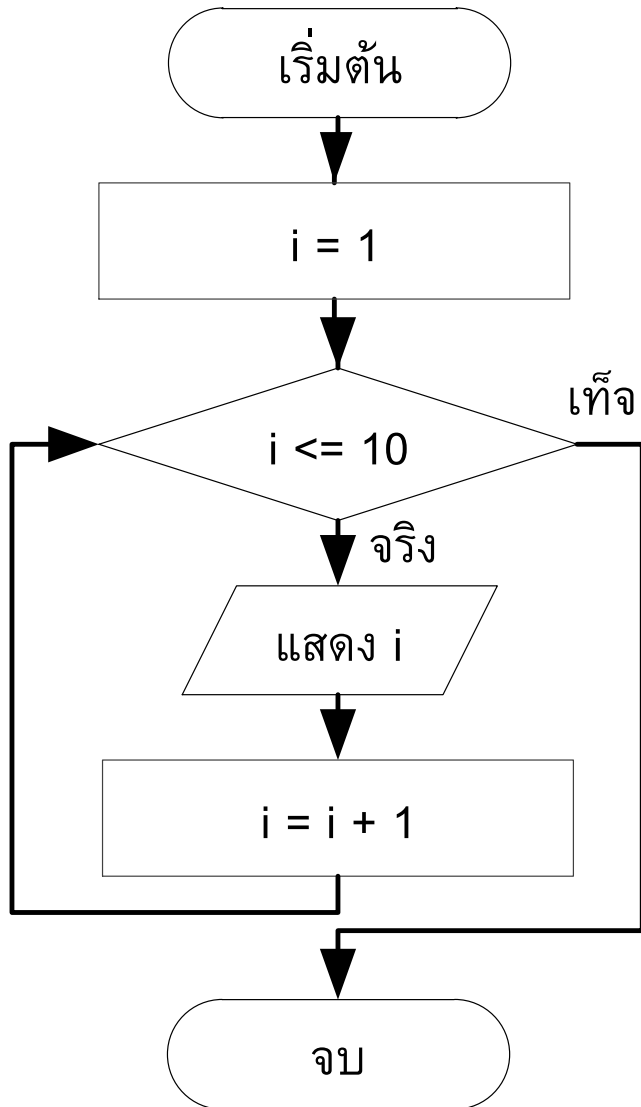
**3. ส่วนของการปรับค่าตัวแปร** เป็นส่วนที่ปรับเปลี่ยนค่าของตัวแปรที่ใช้ในลูป โดยอาจเป็นการเพิ่มหรือลดค่าของตัวแปรก็ได้ ซึ่งส่วนนี้จะต้องส่งผลให้เงื่อนไขที่ทำการตรวจสอบในส่วนที่ 2 มีโอกาสเป็นเท็จด้วย ไม่เช่นนั้นอาจออกจากการทำงานของลูปไม่ได้ เกิดการทำงานของลูปที่ไม่รู้จบ (Infinite loop)

# ลำดับการทำงานของ คำสั่ง for



```
int Counter;  
for(Counter=1; Counter<=10;Counter++)  
{  
    Statement A;  
    ...  
}
```

# ตัวอย่าง จงเขียนโปรแกรมแสดงตัวเลข 1-10



```
#include<stdio.h>
int main() {
    int i;
    for (i=1;i<=10;i++) {
        printf("%d\t",i) ;
    }
    return 0;
}
```

# ตัวอย่าง

```
#include<stdio.h>
int main(){
    int n = 5;
    for( ; n > 0; n--)
        printf("%d  ",n) ;
    return 0;
}
```

สแตตเมนต์ว่าง

Input-Output

5 4 3 2 1

# จงหาค่าต่างๆต่อไปนี้จากคำสั่ง for ที่กำหนดให้

	จำนวนรอบที่ทำงาน	ค่า i สุดท้ายใน loop	ค่า i เมื่อจบ loop
--	------------------	----------------------	--------------------

1. for(i=0;i<10;i++)			
----------------------	--	--	--

1. for(i=0;i<10;i++)	10		
----------------------	----	--	--

1. for(i=0;i<10;i++)	10	9	
----------------------	----	---	--

1. for(i=0;i<10;i++)	10	9	10
----------------------	----	---	----

2. for(i=1;i<=100;i++)			
------------------------	--	--	--

2. for(i=1;i<=100;i++)	100		
------------------------	-----	--	--

2. for(i=1;i<=100;i++)	100	100	
------------------------	-----	-----	--

2. for(i=1;i<=100;i++)	100	100	101
------------------------	-----	-----	-----

3. for(i=50;i>0;i-=2)			
-----------------------	--	--	--

3. for(i=50;i>0;i-=2)	25		
-----------------------	----	--	--

3. for(i=50;i>0;i-=2)	25	2	
-----------------------	----	---	--

3. for(i=50;i>0;i-=2)	25	2	0
-----------------------	----	---	---

4. for(i=10;i+5<100;i*=2)			
---------------------------	--	--	--

4. for(i=10;i+5<100;i*=2)	4		
---------------------------	---	--	--

4. for(i=10;i+5<100;i*=2)	4	80	
---------------------------	---	----	--

4. for(i=10;i+5<100;i*=2)	4	80	160
---------------------------	---	----	-----

# เปรียบเทียบการทำงานของ for และ while

```
#include<stdio.h>
int main() {
    int i; 1
    for (i=1; i<=10; i++) 2
    {
        printf("%d\t", i); 3
    }
    return 0;
}
```

```
#include<stdio.h>
int main() {
    int i = 1; 1
    while (i<=10) 2
    {
        printf("%d\t", i);
        i = i + 1; 3
    }
    return 0;
}
```

Input-Output

1 2 3 4 5 6 7 8 9 10



# จงแปลงคำสั่ง for เป็น while

```
int i;  
for(i=10;i<20;i++) {  
    printf("%d ",i);  
}
```

```
int i=10;  
while(i<20) {  
    printf("%d ",i);  
    i++;  
}
```

## Input-Output

10 11 12 13 14 15 16 17 18 19

# แปลงคำสั่ง for เป็น while

```
int k,m;  
for (k=0,m=10 ; k<10 || m<50 ; k+=2,m*=2)  
{  
    printf("%d ",k) ;  
    printf("%d\n",m) ;  
}
```

```
int k=0,m=10;  
while(k<10 || m<50) {  
    printf("%d ",k) ;  
    printf("%d\n",m) ;  
    k+=2 ;  
    m*=2 ;  
}
```

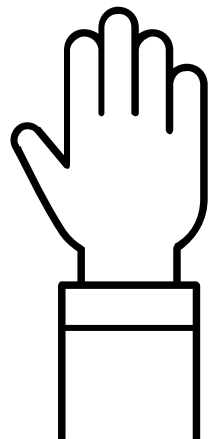
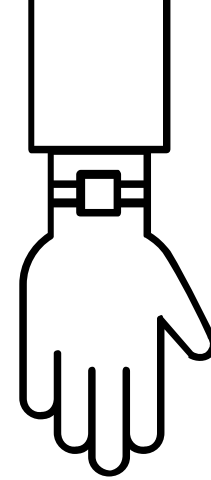
## Input-Output

6 80

8 160

# สรุป

- คำสั่ง **while**
  - ใช้กับลูปที่ไม่ทราบจำนวนทำซ้ำที่แน่นอน
  - ทำซ้ำโดยตรวจสอบเงื่อนไขก่อนเข้าลูป
- คำสั่ง **do while**
  - ทำซ้ำโดยตรวจสอบเงื่อนไขหลังจากทำงานในลูปไปแล้วหนึ่งครั้ง
- คำสั่ง **for**
  - ใช้กับลูปที่ทราบรอบการทำงานแน่นอน
  - ทำซ้ำจนกว่าเงื่อนไขที่กำหนดเป็นเท็จแล้วจึงออกนอกลูป



# แบบฝึกหัด



# การบ้าน

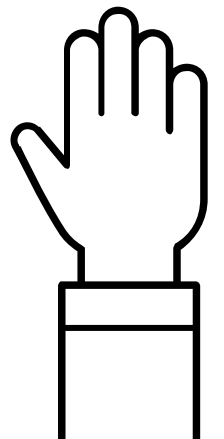
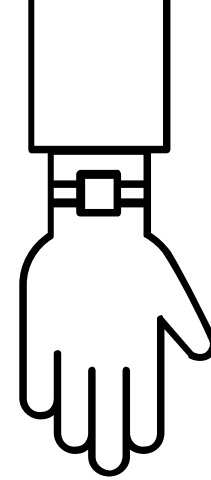
## (ส่งใน Google Classroom)

- 3.1 โปรแกรมรับค่าตัวเลขจำนวน 10 ตัว แล้วหาค่าเฉลี่ย
- 3.2 โปรแกรมรับค่าตัวเลขจำนวนเต็ม 10 ตัว แล้วหาค่าต่ำสุดและสูงสุด
- 3.3 โปรแกรมรับค่าตัวเลขจำนวนเต็มบวก แล้วบอกว่าเป็นจำนวนเฉพาะหรือไม่
- 3.4 โปรแกรมรับค่าตัวเลข แล้วบอกว่ตั้งแต่เลข 1 ถึงเลขที่รับเข้ามา มีจำนวนเฉพาะกี่ตัว และมีเลขอะไรบ้าง
- 3.5 โปรแกรมรับค่าจำนวนเต็มบวก 2 ตัว จากนั้นให้แสดงจำนวนที่หารด้วยจำนวนที่รับมาทั้งคู่ลงตัว ตั้งแต่ 1-1000
- 3.6 โปรแกรมรับค่าตัวอักษรภาษาอังกฤษ 1 ตัว จากนั้นให้พิมพ์ตัวอักษรภาษาอังกฤษตั้งแต่ตัวที่รับเข้ามา ถอยหลังจนถึง A (หากใส่ตัวพิมพ์เล็กให้แปลงเป็นตัวพิมพ์ใหญ่)

3.7 เขียนโปรแกรมรับค่าจำนวนเต็ม 1 จำนวน (n) และ  
พิมพ์ผลลัพธ์  $x^2$  ตั้งแต่ 1 – n จำนวน

- สามารถแสดงเป็นทศนิยมได้
- ถ้าเป็นจำนวนเต็มลบให้เริ่มต้นที่ 1 .....-n

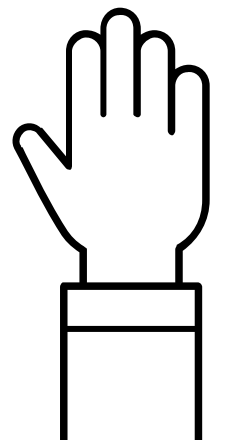
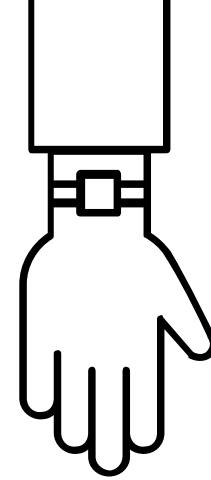
Input :	1		3		5	
Output :	1	1	1	1	1	1
			2	4	2	4
			3	9	3	9
					4	16
					5	25



3.8. เขียนโปรแกรมรับค่าจำนวนเต็ม 1 จำนวน (n) และ  
พิมพ์ผลลัพธ์  $2^n$  ตั้งแต่ 1 – n จำนวน

- สามารถแสดงเป็นทศนิยมได้
- ถ้าเป็นจำนวนเต็มลบให้เริ่มต้นที่ 1 .....-n

Input :	1		3		5	
Output :	1	2	1	2	1	2
			2	4	2	4
			3	8	3	8
					4	16
					5	32



## 3.9 จงเขียนโปรแกรมโดยใช้ loop เพื่อให้แสดงผลดังต่อไปนี้

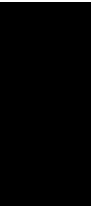
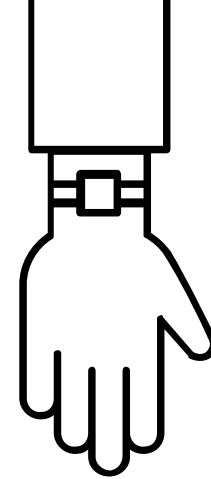
\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

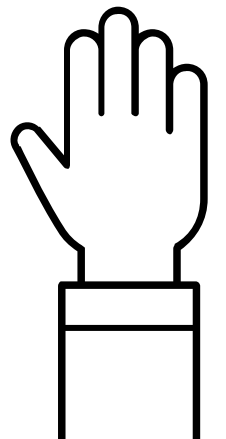
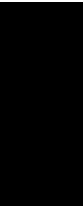
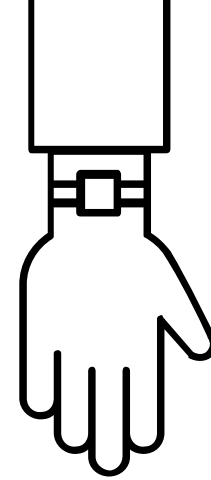
\*\*\*\*\*





## 3.10 จงเขียนโปรแกรมโดยใช้ loop เพื่อให้แสดงผลดังต่อไปนี้

```
*  
**  
***  
****  
*****
```



### 3.11 จงเขียนโปรแกรมทำปฏิทิน โดยอินพุตที่วันที่ 1 ของเดือน และจำนวนวันในเดือน

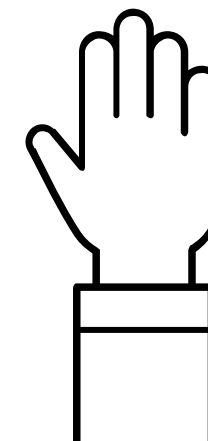
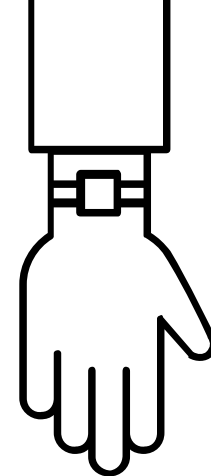
(1 = อาทิตย์ 2=จันทร์ 3=อังคาร 4=พุธ 5=พฤหัสบดี 6=ศุกร์ 7=เสาร์)

ตัวอย่าง

INPUT 3  
30  
OUTPUT

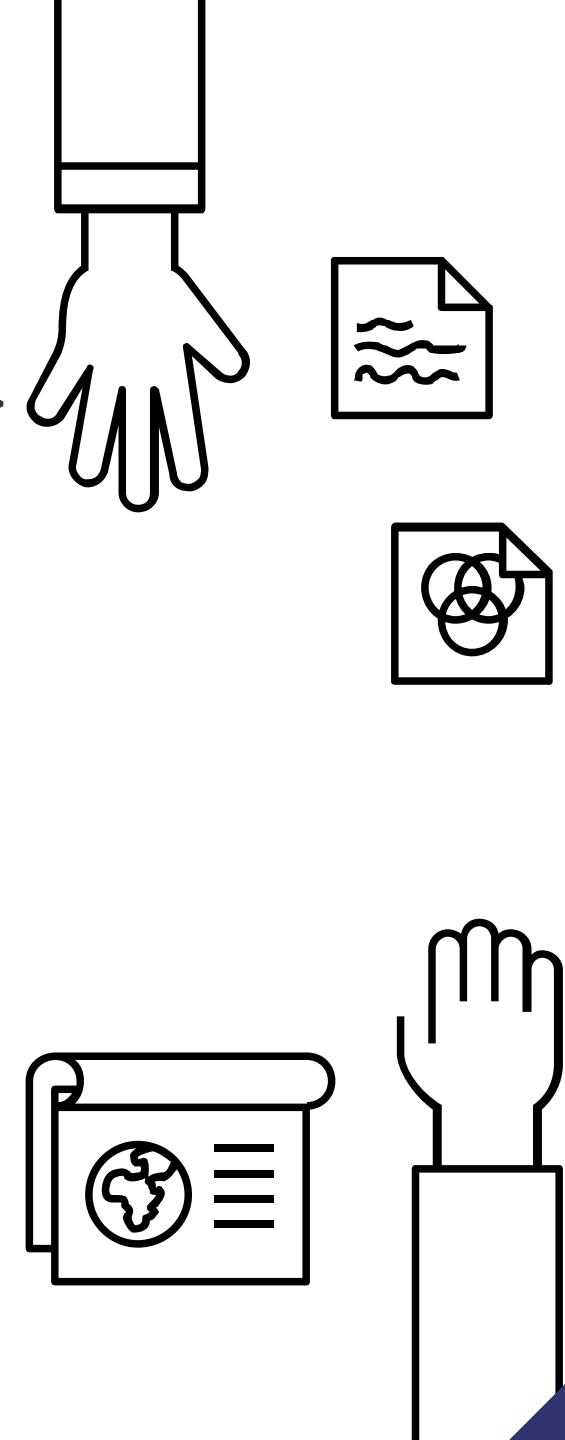
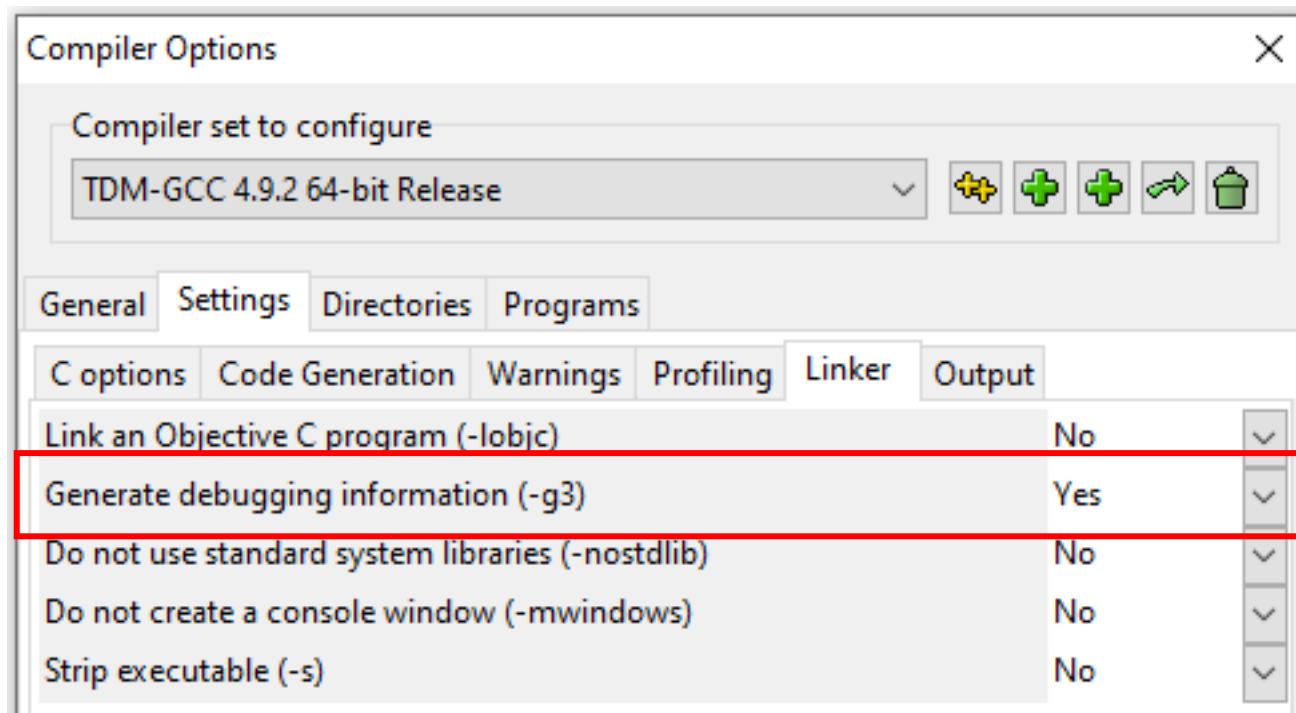
	1	2	3	4	5	
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

3.12 จงพัฒนาให้แสดงต่อ อีก 3 เดือน  
(โดยให้แต่ละเดือนมี จำนวนวันเท่ากับเดือนแรก)



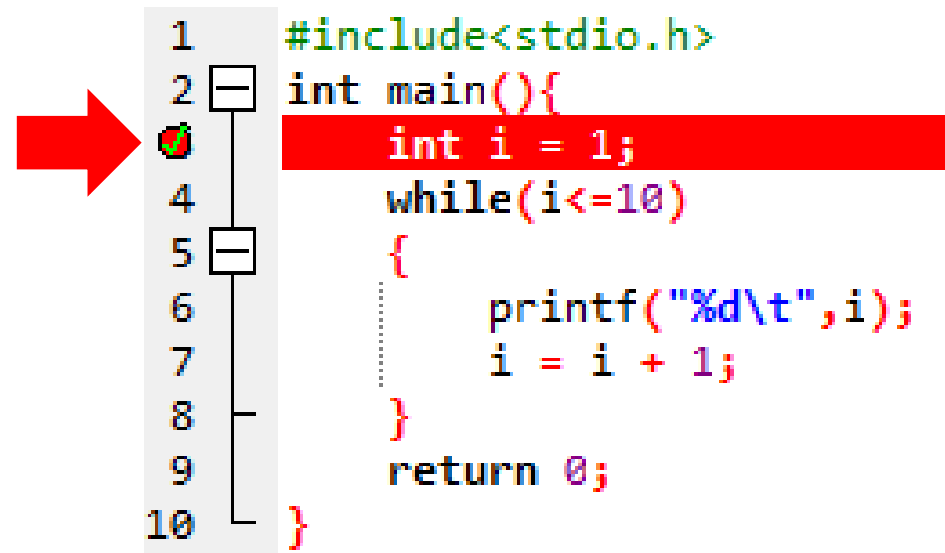
# Debugging in Dev-C++

1. กำหนดค่าให้ linker สร้างข้อมูลการดีบั๊ก โดยเลือกเมนู  
Tools -> Compiler Options -> Settings -> Linker ->  
Generate Debugging Information



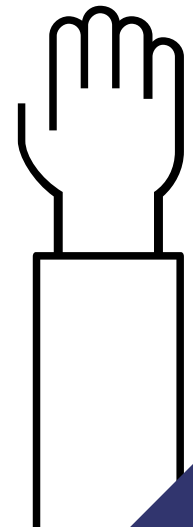
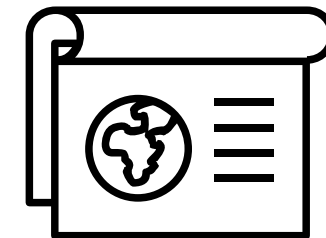
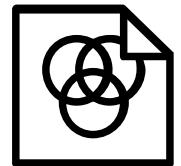
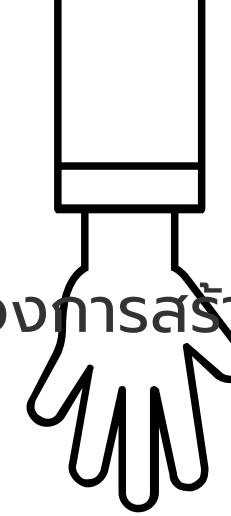
## Debugging in Dev-C++

2. ทดลองสร้างโปรแกรมตัวอย่าง แล้วคลิกที่จุดที่ต้องการสร้าง Breakpoint โดยคลิกที่ขอบทางซ้ายมือ



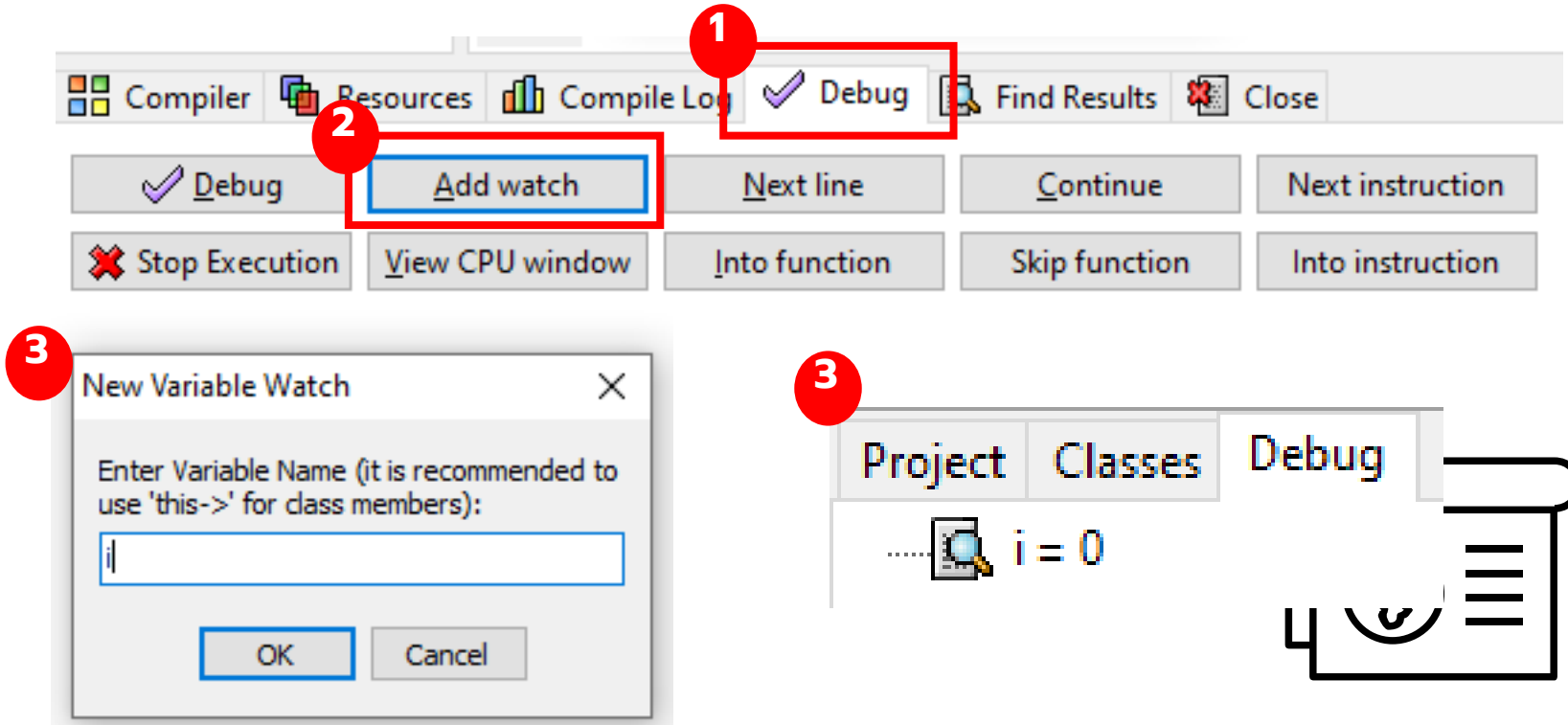
3. คอมไพล์โปรแกรมใหม่อีกครั้ง

4. จากนั้นเลือกเมนู  Debug (F5)



# Debugging in Dev-C++

5. ในขณะนี้จะอยู่ในกระบวนการดีบั๊ก สามารถดูค่าของตัวแปรใดๆได้ โดยการเลื่อนเมาส์ไปวางที่ตัวแปรที่สนใจ หรือ Add watch ชื่อตัวแปรที่ต้องการ



# Debugging in Dev-C++

6. จากนั้นเราสามารถเลือก Next line เพื่อดีบั๊กแต่ละขั้นตอนของโปรแกรม พร้อมๆกันดูค่าของตัวแปรที่สนใจได้
7. สามารถหยุดการดีบั๊ก โดยเลือกเมนู Stop Execution

