

# คำสั่งควบคุมแบบมี ทางเลือกหรือเงื่อนไข

ความหมาย ประเภทของคำสั่งควบคุม  
และ นิพจน์ตรรกศาสตร์

# OUTLINE

## 1. ลำดับการทำงานของคำสั่งในโปรแกรม

- ทำงานแบบเรียงลำดับ
- ทำงานแบบมีเงื่อนไข
- ทำงานแบบวนซ้ำ

## 2. คำสั่งควบคุม

- ประเภทคำสั่งควบคุม
- ความหมายของเงื่อนไข

# OUTLINE

## 3. นิพจน์ตรรกศาสตร์

- ข้อมูลชนิด boolean
- ตัวดำเนินการเชิงสัมพันธ์
- ตัวดำเนินการตรรกะ

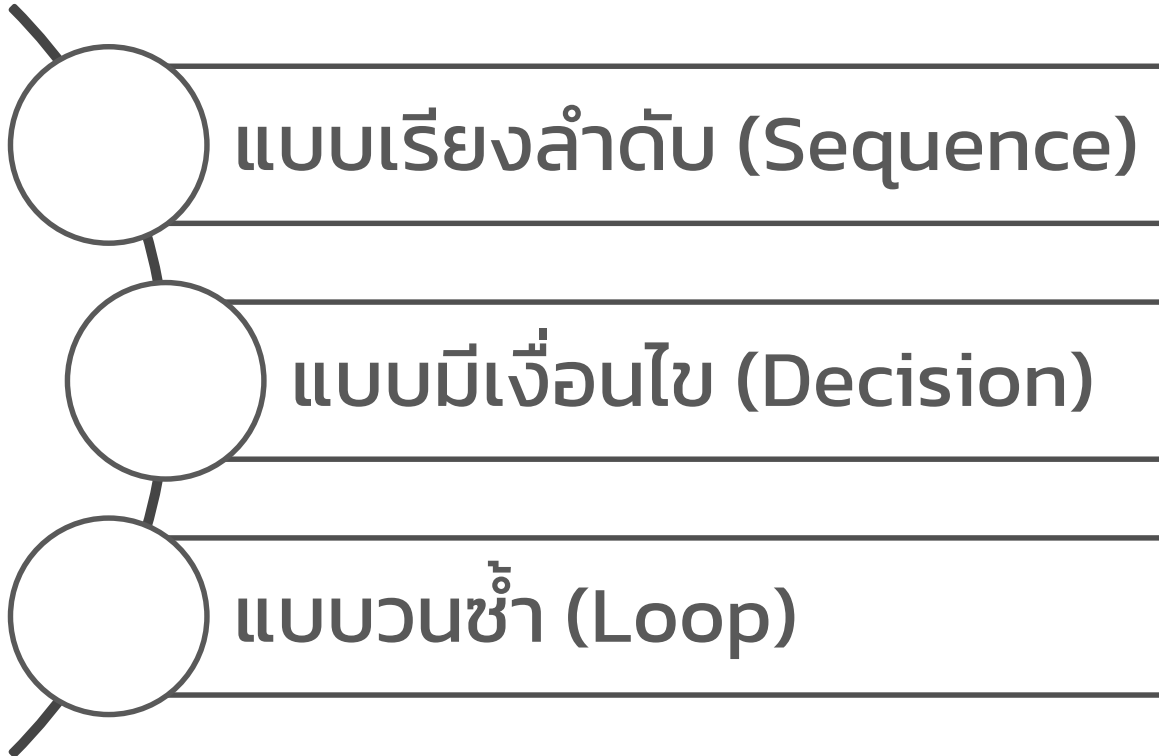
## 4. คำสั่งการทำงานแบบมีทางเลือกหรือเงื่อนไข

- คำสั่ง if-else
- คำสั่ง switch-case



**ลำดับการทำงานของคำสั่งในโปรแกรม**

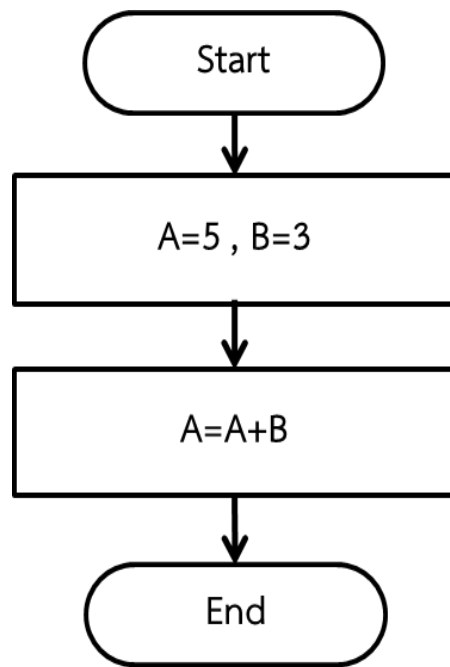
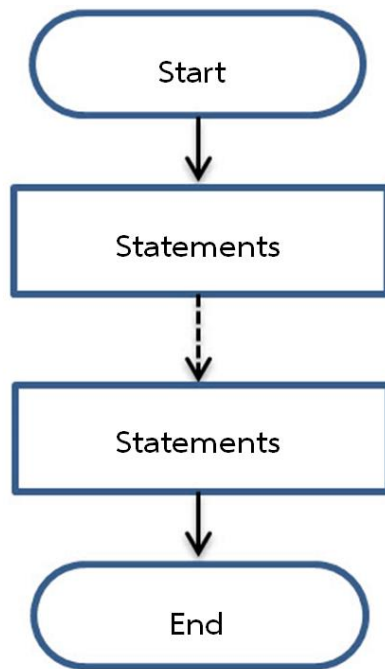
# 1 ลำดับการทำงานของคำสั่งในโปรแกรม



## 1.1

# การทำงานแบบเรียงลำดับ (Sequence)

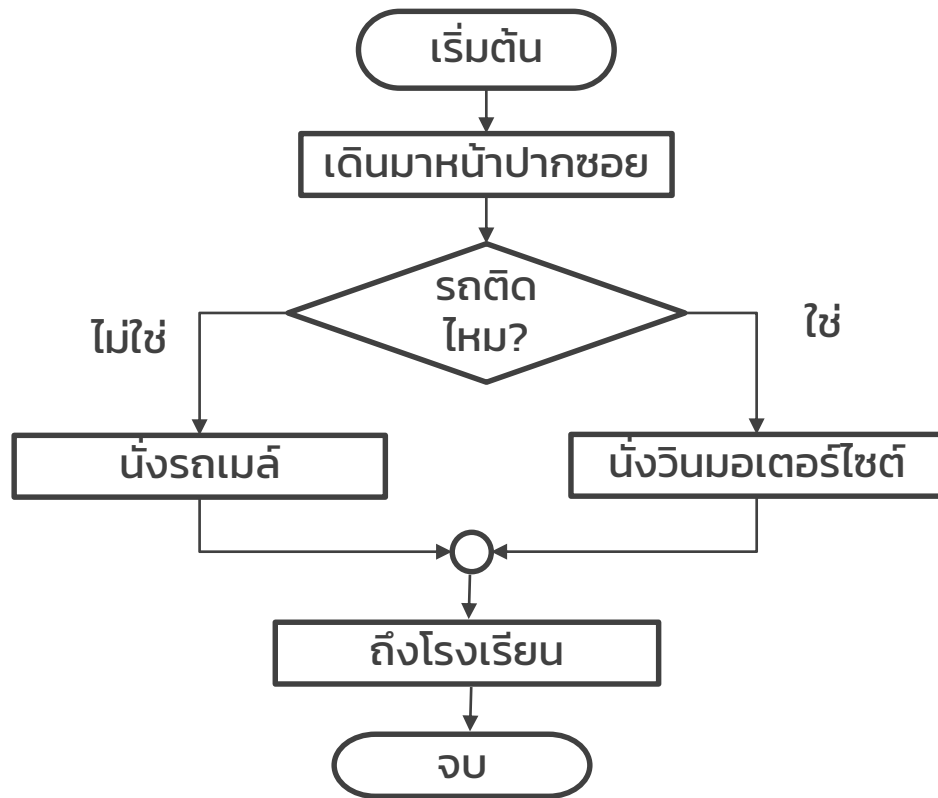
เป็นรูปแบบการเขียนโปรแกรมที่ง่ายที่สุด และไม่มีซับซ้อน มีลำดับการทำงานจากบนลงล่าง มีการทำงานทีละคำสั่งจนจบการทำงาน



## 1.2

# การทำงานแบบมีเงื่อนไข (Decision)

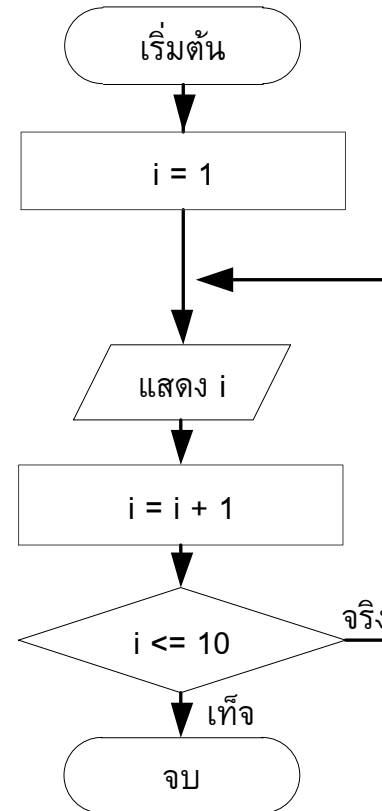
เป็นรูปแบบการเขียนโปรแกรมที่มีทางเลือกเพื่อตัดสินใจซึ่งโปรแกรมจะตรวจสอบเงื่อนไขเพื่อเลือกทิศทางการทำงานของโปรแกรมโดย**เลือกทางเลือกใดทางเลือกหนึ่ง**จากสองทางเลือกเท่านั้นคือ



### 1.3

## การทำงานแบบวนซ้ำ (Loop)

การทำซ้ำ (Loop) เป็นรูปแบบการเขียนโปรแกรมที่มีการทำงานใน **ขั้นตอนเดิมซ้ำกันหลายรอบ** ซึ่งการทำงานของโปรแกรมจะมีการตรวจสอบเงื่อนไข เพื่อกำหนดให้เข้าทำงานในลูปหรือออกจากลูปการทำงาน







## คำสั่งควบคุม (Control Flow Statements)

## 2

# คำสั่งควบคุม (Control Flow Statements)

คือ กลุ่มคำสั่งที่ใช้ควบคุมการเปลี่ยนแปลงลำดับการทำงาน  
ของชุดคำสั่งในโปรแกรม แบ่งได้ 2 ประเภท

2.1 คำสั่งควบคุมที่ *ใช้เงื่อนไข* เป็นองค์ประกอบของคำสั่ง

2.2 คำสั่งควบคุมที่ *ไม่ใช้เงื่อนไข* เป็นองค์ประกอบของคำสั่ง

## 2

# คำสั่งควบคุม (Control Flow Statements)

## 2.1 คำสั่งควบคุมที่ *ใช้เงื่อนไข* เป็นองค์ประกอบของคำสั่ง

- คำสั่งควบคุมแบบทางเลือก if, if-else, switch-case
- คำสั่งควบคุมแบบทำซ้ำ while, do-while, for

## 2.2 คำสั่งควบคุมที่ *ไม่ใช้เงื่อนไข* เป็นองค์ประกอบของคำสั่ง

- กลุ่มคำสั่งที่ใช้ร่วมกับคำสั่งควบคุมแบบใช้เงื่อนไข เช่น break, continue และ label
- กลุ่มคำสั่งอื่นๆ เช่น return, exception

# ความหมายของเงื่อนไข

**เงื่อนไข** หมายถึง นิพจน์ทางตรรกศาสตร์ ที่ให้ **ค่าความจริง** ( boolean ซึ่งมีค่าเป็น *true* หรือ *false* ) แก่คำสั่งควบคุม และคำสั่งควบคุมจะใช้ค่าที่ได้ กำหนดเส้นทางทำงานของโปรแกรม

# ตัวอย่างเงื่อนไข

```
int x = 1;  
if (x <= 4) {  
    printf("%d", x);  
}  
x++;
```

ถ้าตรวจเงื่อนไขแล้วเป็นจริง หมายถึง

ค่าของนิพจน์เป็น **true** จะทำให้ทำคำสั่งอย่างหนึ่ง

ถ้าตรวจเงื่อนไขแล้วเป็นเท็จ หมายถึง

ค่าของนิพจน์เป็น **false** จะทำให้ทำคำสั่งอีกอย่างหนึ่ง



นิพนธ์ตรรกศาสตร์

มีค่าเป็น **true** หรือ **false** ซึ่งอาจเกิดได้จาก

1. ข้อมูลชนิด boolean
2. การดำเนินการเปรียบเทียบ (Comparison Operators)  
โดยตัวดำเนินการเชิงสัมพันธ์ (Relational Operators)
3. การดำเนินการทางตรรกะ โดยตัวดำเนินการตรรกะ  
(Logical operators)

## 3.1 ข้อมูลชนิด boolean

Boolean เป็นประเภทข้อมูลที่มีได้เพียงสองค่าคือจริง true และเท็จ false เก็บค่าที่เป็นไปได้เพียงสองค่า

- 1 หรือ true สำหรับค่าที่เป็นจริง
- 0 หรือ false สำหรับค่าที่เป็นเท็จ

```
1 #include<stdio.h>
2 #include <stdbool.h>
3 int main(){
4     bool x = true;
5     printf("%d\n", x);
6     printf(x ? "true" : "false");
7     return 0;
8 }
```

ผลการรัน :

```
1
true
```



## 3.2

# ตัวดำเนินการเชิงสัมพันธ์ (Relational Operators)

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

# ตัวอย่าง เมื่อให้ $x = 5$ , $y = 10$

นิพจน์	ค่าความจริง
$x == y$	เท็จ
$x > y$	เท็จ
$x >= y$	เท็จ
$x <= y$	จริง
$x != y$	จริง
$x * x < y * y$	จริง
$x + y >= x * y$	เท็จ

# ตัวอย่างโปรแกรม

## ตัวดำเนินการเชิงสัมพันธ์

```
1  #include<stdio.h>
2  int main(){
3      int x=5 ,y=10;
4      printf("%d\n",x>y);
5      printf("%d\n",x<y);
6      printf("%d\n",x>=y);
7      printf("%d\n",x<=y);
8      printf("%d\n",x==y);
9      printf("%d\n",x!=y);
10     return 0;
11 }
```

ผลการรัน :

0  
1  
0  
1  
0  
1

# ลำดับในการดำเนินการ



( )	ตัวดำเนินการทางคณิตศาสตร์
++ - + - !	
* / %	
+ -	
< <= > >=	Relation Operators
== !=	
&&	Logical Operators
= += -= *= /= %=	

นิพจน์ต่อไปนี้เป็นจริงหรือเท็จ

$$2+3+5-1 > 4+5$$

$$9 > 9$$

**FALSE**

นิพจน์ต่อไปนี้เป็นจริงหรือเท็จ

1

$$2 + 6/2 - 3 + 4*2 == 2*2 + 6$$

2

$$2 + 3 - 3 + 8 == 4 + 6$$

3

10

==

10

TRUE

### 3.3

## ตัวดำเนินการตรรกะ (Logical Operators )

เป็นตัวดำเนินการที่ใช้ประมวลผลนิพจน์ตรรกะในรูปแบบพีชคณิตบูลีน

ตัวดำเนินการ	ความหมาย	ตัวอย่าง
&&	และ (and)	x && y
	หรือ (or)	x    y
!	ไม่ (not)	!x

# ตัวดำเนินการ **and** และ การประมวลผล

นิพจน์A	นิพจน์B	A&&B
false	false	false
false	true	false
true	false	false
true	true	?



# อธิบายตัวดำเนินการ **and**

ถ้านิพจน์ A, B มีค่าเป็นจริงทั้งสองตัว  
จะให้ค่า**เป็นจริง**

ถ้านิพจน์ตัวใดตัวหนึ่งเป็นเท็จหรือเป็นเท็จทั้ง  
สองตัว จะให้ค่า**เป็นเท็จ**

# ตัวดำเนินการ **Or** และ การประมวลผล

นิพจน์ A	นิพจน์ B	$A \parallel B$
false	false	?
false	true	true
true	false	true
true	true	true

# อธิบายตัวดำเนินการ or

ถ้า นิพจน์ A, B มีค่าเป็นเท็จทั้งสองตัว  
จะให้ค่า เป็นเท็จ

ถ้า นิพจน์ตัวใดตัวหนึ่ง หรือทั้งสองตัว  
มีค่าเป็นจริง จะให้ค่า เป็นจริง

# ตัวดำเนินการ **not** และ การประมวลผล

นิพจน์ A	นิพจน์ B	!A	!B
false	false	true	true
false	true	true	false
true	false	false	true
true	true	false	false

# อธิบายตัวดำเนินการ !

ตัวดำเนินการ “!” จะให้ค่าตรงกันข้ามกับค่าความจริงของนิพจน์นั้นๆ เช่น

ถ้า A มีค่า **เป็น true** แล้ว

**!A** จะมีค่า **เป็น false**

# ตารางสรุปการประมวลผล

## ของตัวดำเนินการตรรกะ

นิพจน์A	นิพจน์B	A&&B	A  B	!A	!B
false	false	false	false	true	true
false	true	false	true	true	false
true	false	false	true	false	true
true	true	true	true	false	false

กำหนดค่า  $A = 5$  ,  $B = 7$   
ให้ผู้เรียนหาผลลัพธ์ของนิพจน์ต่อไปนี้

ลำดับ	นิพจน์	ผลลัพธ์
1	$A == B$	False
2	$A != B$	True
3	$A >= B$	False
4	$A > B$	False
5	$A < B$	True
6	$A <= B$	True
7	$(A > B) \&\& (A != B)$	False
8	$(A < B) \parallel (A >= B)$	True
9	$!(A <= B) \&\& (A == B)$	False
10	$!((A == B) \parallel (A > B))$	True



**คำสั่งการทำงานแบบมีทางเลือกหรือเงื่อนไข**



## คำสั่งการทำงานแบบมีทางเลือกหรือเงื่อนไข

### 4.1 คำสั่ง if-else

- คำสั่งเลือกทำแบบทางเดียว (if)
- คำสั่งเลือกทำอย่างใดอย่างหนึ่ง (if-else)
- คำสั่งเลือกทำเชิงซ้อน (nested-if statement)

### 4.2 คำสั่ง switch-case

## 4.1

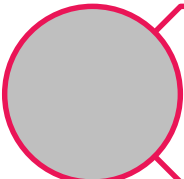
# คำสั่ง if-else



1. คำสั่งเลือกทำแบบทางเดียว (if)



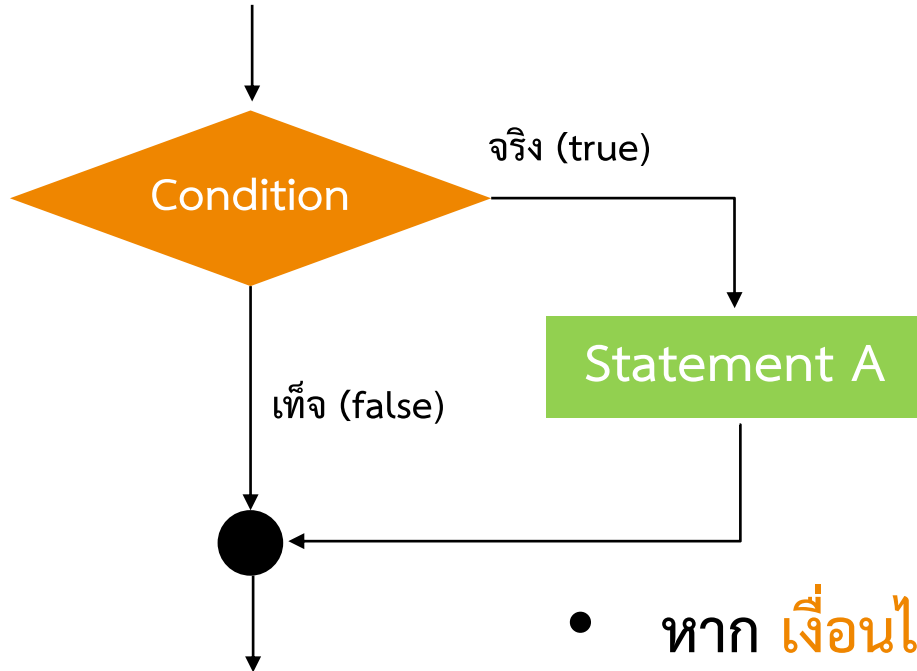
2. คำสั่งเลือกทำอย่างใดอย่างหนึ่ง (if-else)



3. คำสั่งเลือกทำเชิงซ้อน (nested-if statement)

## 4.1.1

# คำสั่งเลือกทำแบบทางเดียว (if)



```
if (condition)
{
    statement A ;
}
```

- หาก **เงื่อนไข** เป็นจริง (ไม่เป็นศูนย์) ทำคำสั่งใน {}
- หากมีคำสั่งเดียวไม่จำเป็นต้องใช้วงเล็บปีกกา

# ตัวอย่าง

```
if (age >= 18) {  
    printf ("Teenager\n");  
}  
printf ("good luck");
```

ถ้าค่าในตัวแปร age เป็นค่าต่าง ๆ จะทำให้แสดงผลลัพธ์ดังต่อไปนี้

age = 25

Teenager  
good luck

age = 14

good luck

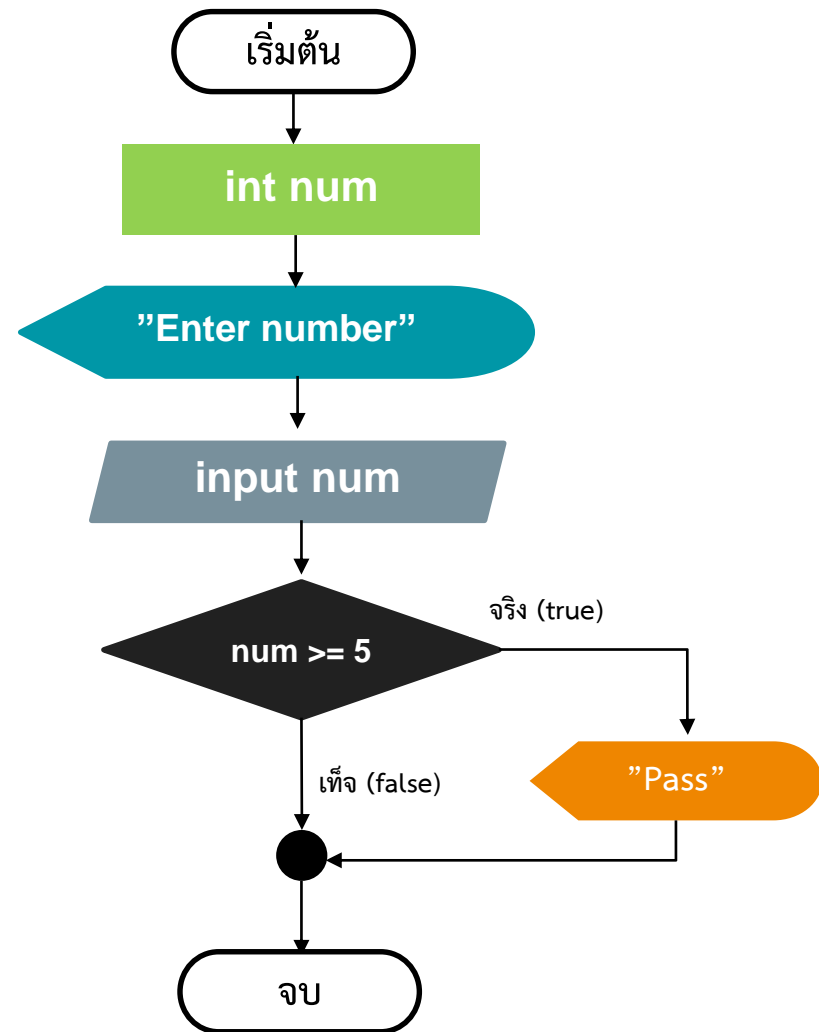
age = 18

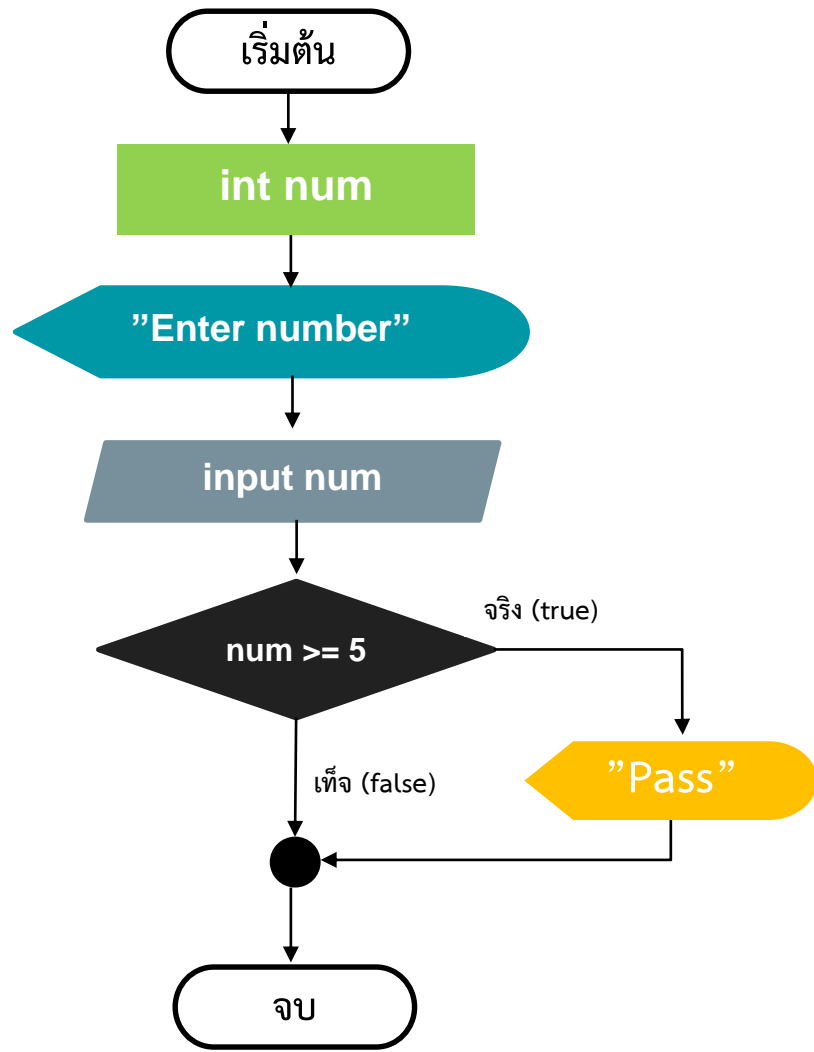
Teenager  
good luck

**ปัญหา :** จงออกแบบขั้นตอนวิธีและเขียนโปรแกรมเพื่อตรวจสอบว่า ถ้าคะแนนมากกว่าหรือเท่ากับ 5 ให้แสดงข้อความว่า "Pass"

Input-Output

Enter number: 8  
Pass





```
#include<stdio.h>
int main() {
    int num;
    printf("Enter number: ");
    scanf("%d", &num);
    if (num >= 5) {
        printf("Pass");
    }
    return 0;
}
```

## 4.1.2

# คำสั่งเลือกทำอย่างใดอย่างหนึ่ง (if-else)

```
if (condition)
```

```
{
```

```
    statement 1 ;
```

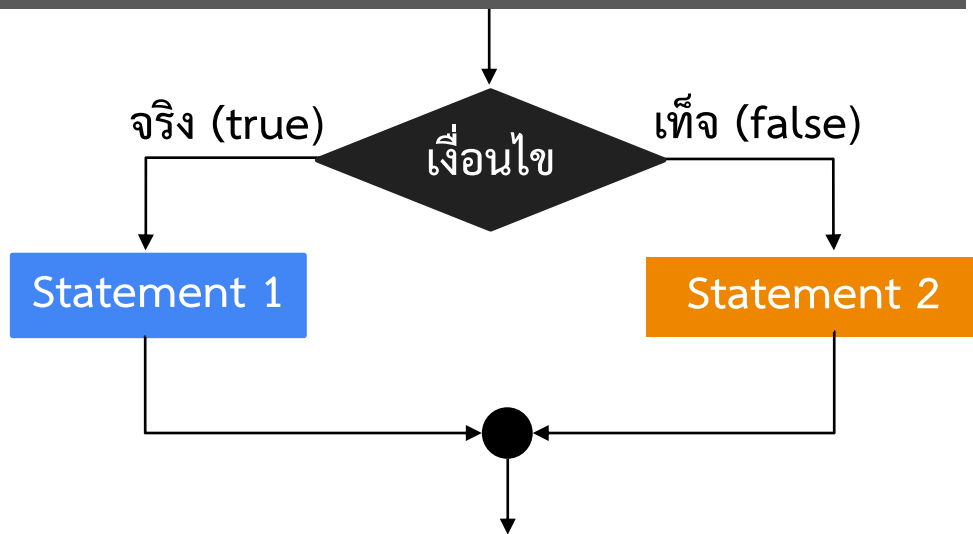
```
}
```

```
else
```

```
{
```

```
    statement 2 ;
```

```
}
```



ถ้าเงื่อนไขเป็นจริงจะทำ  
statement 1 แต่ถ้าเงื่อนไขเป็น  
เท็จจะทำ statement 2

**ปัญหา :** จงออกแบบขั้นตอนวิธีและเขียนโปรแกรมเพื่อตรวจสอบว่า

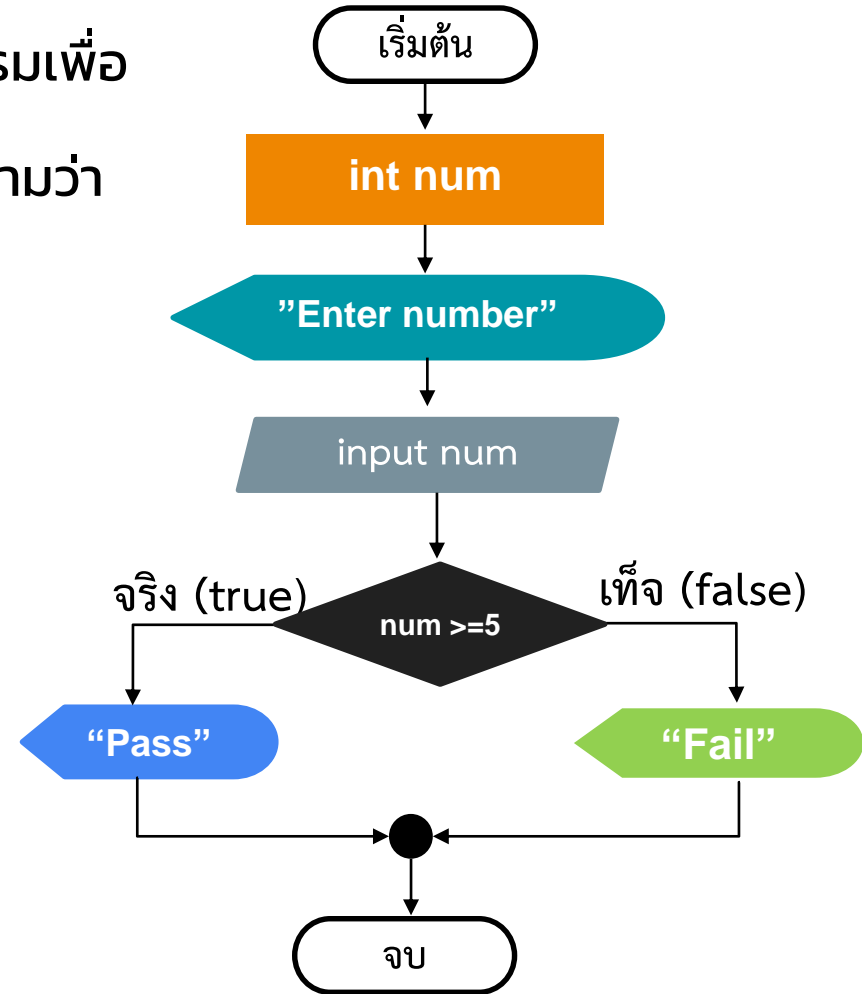
- ถ้าคะแนนมากกว่าหรือเท่ากับ 5 ให้แสดงข้อความว่า "Pass"
- ถ้าน้อยกว่า 5 ให้แสดงข้อความว่า "Fail"

Input-Output

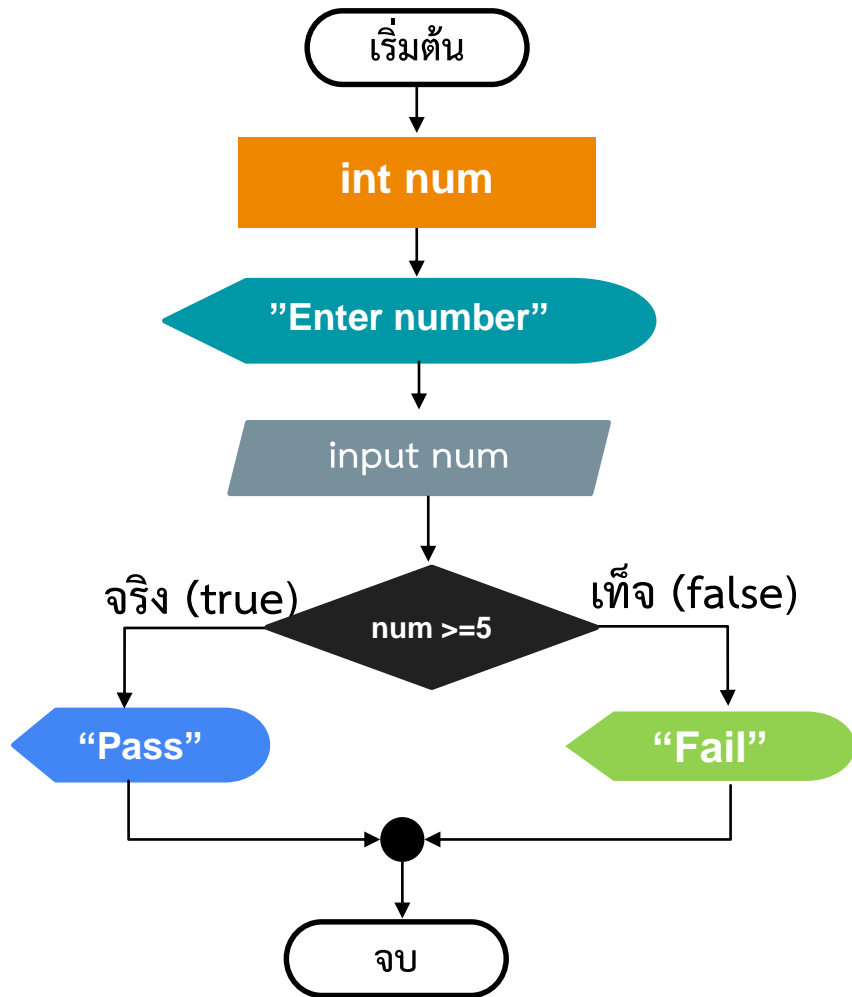
Enter number: 8  
Pass

Input-Output

Enter number: 3  
Fail







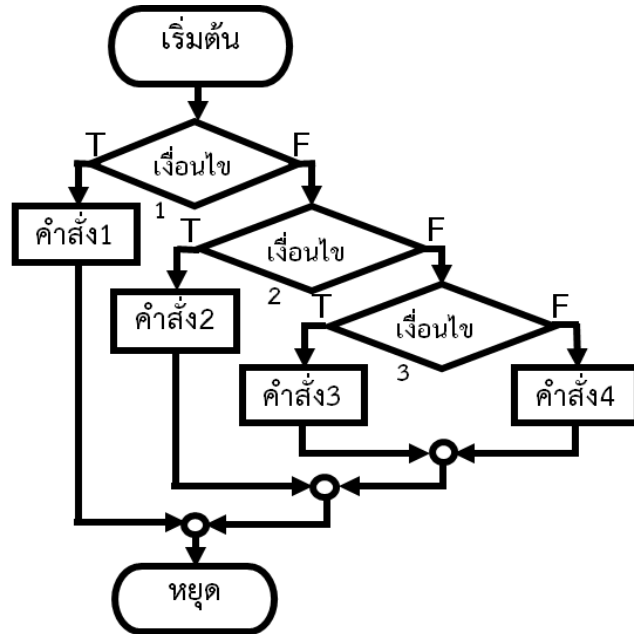
```
#include<stdio.h>
int main()
{
    int num;
    printf("Enter number: ");
    scanf("%d",&num);
    if(num >= 5){
        printf("Pass");
    }
    else{
        printf("Fail");
    }
    return 0;
}
```

# ตัวอย่างโปรแกรม if-else

```
#include <stdio.h>
int main()
{
    int x, y;
    scanf("%d %d ", &x, &y);
    if (x > y)
        printf("x is greater than y ");
    else
        printf("x is less than or equal to y ");
    return 0;
}
```

## 4.1.3 คำสั่งเลือกทำเชิงซ้อน (nested-if statement)

คือ การใช้คำสั่ง **if ซ้อน** อยู่ภายในคำสั่ง if เป็นชั้นๆ เพื่อสร้างทางเลือกของโปรแกรมให้**มีมากกว่า 2 ทางเลือก**



```
if (เงื่อนไขที่1){  
    ชุดคำสั่งที่ 1 ;  
} else if (เงื่อนไขที่ 2) {  
    ชุดคำสั่งที่ 2 ;  
} else if (เงื่อนไขที่ 3) {  
    ชุดคำสั่งที่ 3 ;  
} else {  
    ชุดคำสั่งที่ 4 ;  
}
```

# ตัวอย่างการโปรแกรม nested if

```
int main() {
    float  score;
    char  grade;
    scanf("%f ", &score);

    if (score > 100)
        printf(" Score must be less than or equal to 100");
    else if (score >= 80)
        grade = 'G';
    else if (score >= 50)
        grade = 'P';
    else
        grade = 'F';
    printf("\n Grade = %c", grade);
    return 0;
}
```

## ตัวอย่าง

```
if (เงื่อนไขที่ 1)
    {คำสั่งที่ 1;...}
else if (เงื่อนไขที่ 2)
    {คำสั่งที่ 2;...}
else if (เงื่อนไขที่ 3)
    {คำสั่งที่ 3;...}
else
    {คำสั่งที่ 4;...}
```

## คำถาม

1. คำสั่งที่2 จะกระทำ เมื่อเงื่อนไข1,2,3 เป็นอย่างไร?  
เมื่อเงื่อนไขที่ 1 เป็นเท็จ และเงื่อนไขที่ 2 เป็นจริง
2. คำสั่งที่4 จะกระทำ เมื่อเงื่อนไข1,2,3 เป็นอย่างไร?  
เมื่อเงื่อนไขที่ 1, 2 และ 3 เป็นเท็จ
3. ถ้าเงื่อนไข 1 เท็จ และ เงื่อนไข 2 และ 3 เป็นจริง  
โปรแกรมจะทำคำสั่งใดบ้าง?  
คำสั่งที่ 2 เพียงคำสั่งเดียว
4. หากเงื่อนไขเป็นจริงทั้งหมด?  
จะทำคำสั่งแรก คือ คำสั่งที่ 1 เพียงคำสั่งเดียว

# จงเขียนโปรแกรมตัดเกรดจากคะแนนสอบ (X) เงื่อนไข

คะแนน 80-100 เกรด A

คะแนน 70-79 เกรด B

คะแนน 60-69 เกรด C

คะแนน 50-59 เกรด D

คะแนน < 50 เกรด F

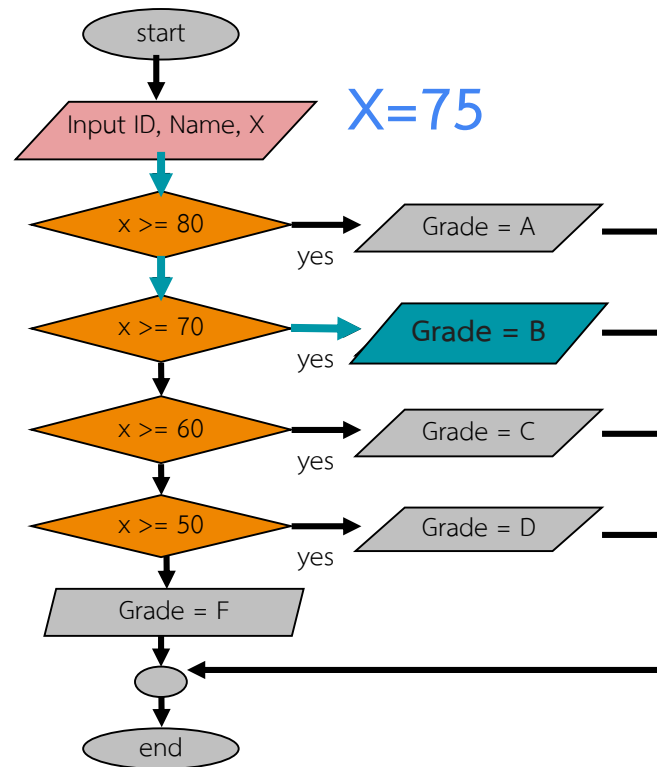
## ผลลัพธ์

Enter ID: 64004100

Enter Name: Wannakarn

Enter X (0-100): 75

Grade = B



# จงเขียนโปรแกรมตัดเกรดจากคะแนนสอบ (X)

เงื่อนไข

คะแนน 80-100 เกรด A

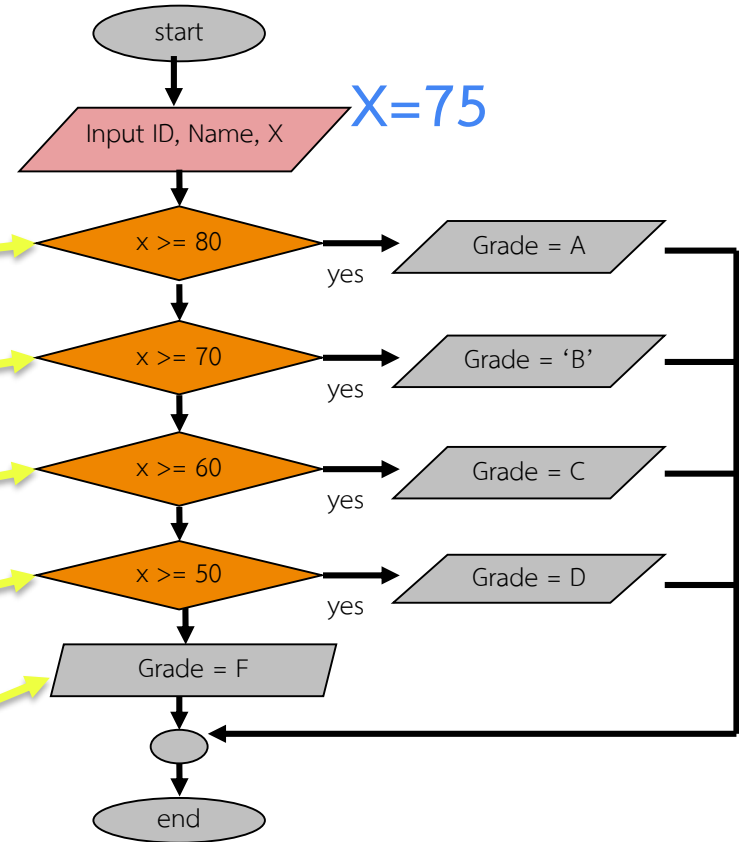
คะแนน 70-79 เกรด B

คะแนน 60-69 เกรด C

คะแนน 50-59 เกรด D

คะแนน < 50 เกรด F

```
if (X >= 80)
    printf ("Grade = A");
else if (X >= 70)
    printf ("Grade = B");
else if (X >= 60)
    printf ("Grade = C");
else if (X >= 50)
    printf ("Grade = D");
else
    printf ("Grade = F");
```



# จงเขียนโปรแกรมตัดเกรดจากคะแนนสอบ (X)

เงื่อนไข

คะแนน 80-100 เกรด A

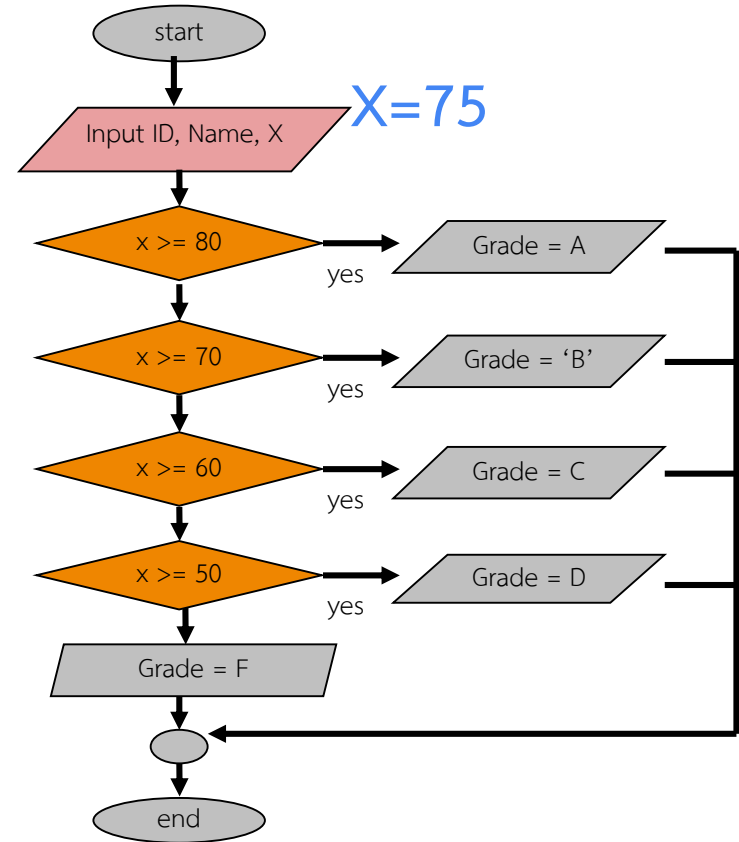
คะแนน 70-79 เกรด B

คะแนน 60-69 เกรด C

คะแนน 50-59 เกรด D

คะแนน < 50 เกรด F

```
if (X >= 80 && X <= 100)
    printf ("Grade = A");
else if (X >= 70 && X <= 79)
    printf ("Grade = B");
else if (X >= 60 && X <= 79)
    printf ("Grade = C");
else if (X >= 50 && X <= 59)
    printf ("Grade = D");
else
    printf ("Grade = F");
```





## 4.2 คำสั่ง switch-case

เป็นคำสั่งตรวจสอบค่าทางคณิตศาสตร์ที่เป็นเลขจำนวนเต็ม(ข้อมูลชนิด int) หรืออักขระ(char) กับ ค่าคงที่หลัง case

ถ้าตรงกับค่าใน case ใด ก็จะทำคำสั่งที่อยู่หลัง case นั้น และคำสั่งหลังทุก case ถัดมาโดยไม่ตรวจสอบอีก ถ้าต้องการทำคำสั่งหลัง case เดียว ให้เพิ่มคำสั่ง break เป็นคำสั่งสุดท้ายใน case นั้น

ถ้าไม่ตรงกับค่าใน case ใด จะทำคำสั่งหลังคำสั่ง default

## 4.2

## คำสั่ง switch-case

switch (ตัวแปร)

{

case ค่าคงที่ 1:

คำสั่งที่ 1;

คำสั่งที่ n;

case ค่าคงที่ 2 :

คำสั่งที่ 2 ;

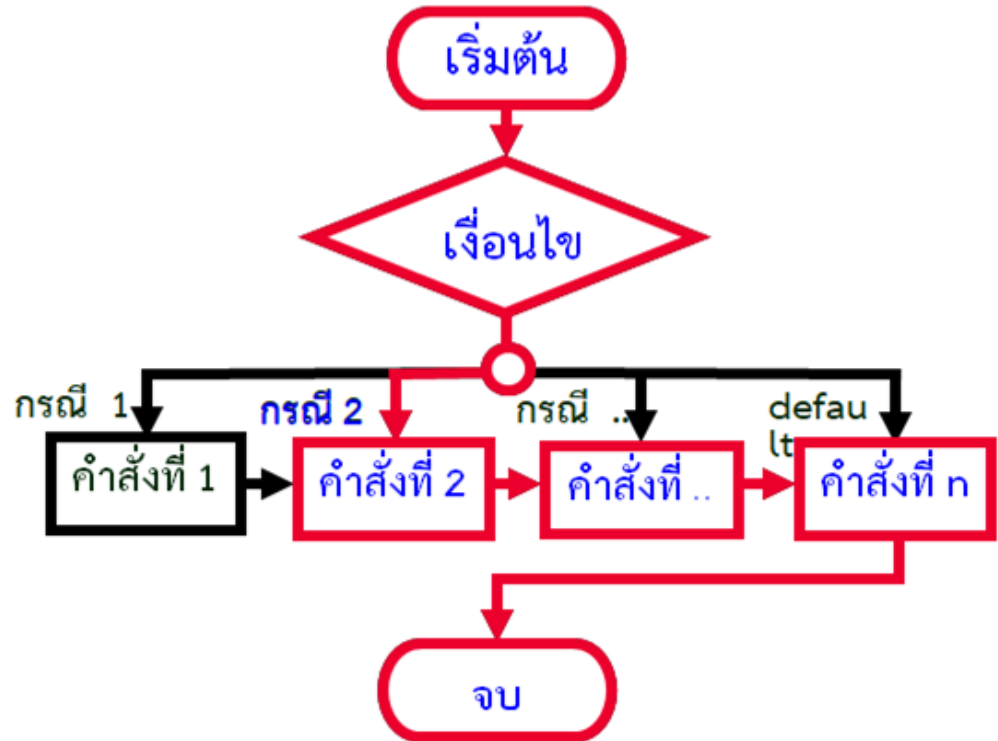
คำสั่งที่ n ;

default :

คำสั่งที่ 3 ;

คำสั่งที่ n ;

}



## 4.2

# คำสั่ง switch-case

```
#include <stdio.h>
int main()
{
    int num=2;
    switch(num+1)
    {
        case 1:
            printf("Case1: Value is: %d", num);
        case 2:
            printf("Case2: Value is: %d", num);
        case 3:
            printf("Case3: Value is: %d", num);
    }
    return 0;
}
```

ผลลัพธ์

Case3: Value is: 3

## 4.2

# คำสั่ง switch-case

```
#include <stdio.h>
int main()
{
    int num=2;
    switch(num+1)
    {
        case 1:
            printf("Case1: Value is: %d", num);
            case 1:
                printf("Case1: Value is: %d", num);
            case 2:
                printf("Case2: Value is: %d", num);
            case 3:
                printf("Case3: Value is: %d", num);
        }
    return 0;
}
```

ERROR

# ตัวอย่าง

จงเขียนโปรแกรมแสดงเลข 1 2 และ 3 เป็นภาษาอังกฤษ

```
#include<stdio.h>
int main()
{
    int c;
    printf("Enter integer 1 or 2 or 3 : ");
    scanf("%d",&c);
    switch(c)
    {
        case 1: printf("ONE\n");
        case 2: printf("TWO\n");
        case 3: printf("THREE\n");
    }
    return 0;
}
```

ต้องใช้คำสั่ง break; มาช่วย

ผลลัพธ์

Enter integer 1 or 2 or 3 : 3

THREE

ผลลัพธ์

Enter integer 1 or 2 or 3 : 2

TWO

THREE

← ไม่ต้องการ

ผลลัพธ์

Enter integer 1 or 2 or 3 : 1

ONE

TWO

THREE

} ไม่ต้องการ

# คำสั่ง break

- คำสั่ง break จะใช้สำหรับการควบคุมการกระทำ โดยบังคับการกระทำ
- บ่อยครั้งที่จะใช้คำสั่ง break เป็นคำสั่งสุดท้ายในแต่ละ case
- หากไม่มีคำสั่ง break ในชุดคำสั่งของ case ไต โปรแกรมจะทำงานต่อไป ในคำสั่งของ case ถัดๆไป ด้วยจนจบ

switch (variable or an integer expression)

```
{    case ค่าที่ 1 : คำสั่งที่ 1; break;
      case ค่าที่ 2 : คำสั่งที่ 2; break;
      case ค่าที่ 3 : คำสั่งที่ 3; break;
      case ... .. break;
}
```

switch (ตัวแปร)

{

case ค่าคงที่ 1 :  
คำสั่งที่ 1 ;  
คำสั่งที่ n ;

break;

case ค่าคงที่ 2 :  
คำสั่งที่ 2 ;  
คำสั่งที่ n ;  
break;

default :  
คำสั่งที่ 3 ;  
คำสั่งที่ n ;

}

กรณีใช้  
break

ผังงาน

เริ่มต้น

เงื่อนไข

กรณี 1

คำสั่งที่ 1

break

กรณี 2

คำสั่งที่ 2

break

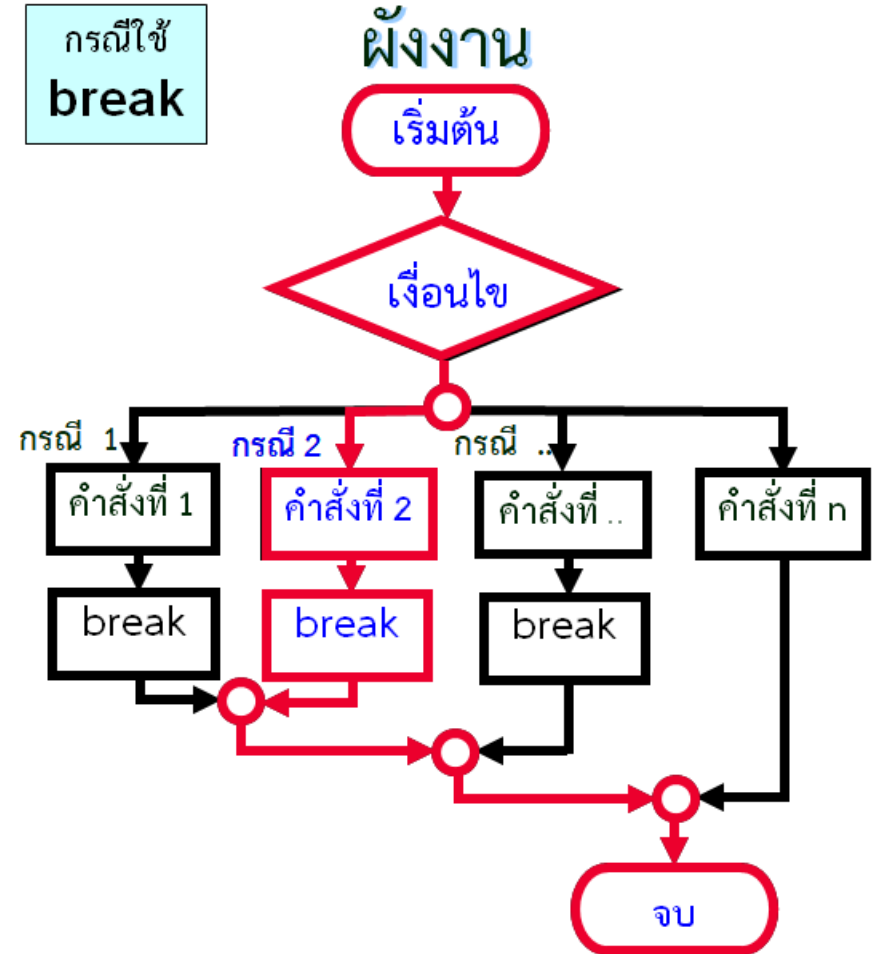
กรณี ..

คำสั่งที่ ..

break

คำสั่งที่ n

จบ



# ตัวอย่าง (ต่อ)

```
#include<stdio.h>
int main()
{
    int c;
    printf("Enter integer 1 or 2 or 3 : ");
    scanf("%d",&c);
    switch(c)
    {
        case 1: printf("ONE\n");
                break;
        case 2: printf("TWO\n");
                break;
        case 3: printf("THREE\n");
                break;
    }
    return 0;
}
```

ผลลัพธ์

Enter integer 1 or 2 or 3 : 3

THREE

ผลลัพธ์

Enter integer 1 or 2 or 3 : 2

TWO

ผลลัพธ์

Enter integer 1 or 2 or 3 : 1

ONE

ถ้าป้อนเลข 4 เข้าไปในโปรแกรมจะเกิดอะไรขึ้น ?



# คำสั่ง default

- **default** เป็นคำสั่งวน
- **default** เป็นอีกกรณีหนึ่งในคำสั่ง switch-case มักวางไว้เป็นกรณีสุดท้าย
- ในกรณีที่ตรวจสอบแล้วพบว่า นิพจน์ มีค่าไม่ตรงกับ **case** ใดๆเลยข้างต้น โปรแกรมจะเข้าไปทำงานในส่วนของ **default**
- ไม่จำเป็นต้องใส่ **break** หลังชุดคำสั่งของ **default**

# ตัวอย่าง (ต่อ)

```
#include<stdio.h>
int main()
{
    int c;
    printf("Enter integer 1 or 2 or 3 : ");
    scanf("%d",&c);
    switch(c)
    {
        case 1: printf("ONE\n");
                break;
        case 2: printf("TWO\n");
                break;
        case 3: printf("THREE\n");
                break;
        default: printf("Out of range");
    }
    return 0;
}
```

ผลลัพธ์

Enter integer 1 or 2 or 3 : 4  
Out of range

# คำสั่ง return

- คือคำสั่งที่ใช้ในการออกจากฟังก์ชันหรือโปรแกรม

ผลลัพธ์

```
Enter integer 1 or 2 or 3 : 1
ONE
```

ผลลัพธ์

```
Enter integer 1 or 2 or 3 : 2
TWO
Goodbye
```

```
#include<stdio.h>
int main()
{
    int c;
    printf("Enter integer 1 or 2 or 3 : ");
    scanf("%d",&c);
    switch(c)
    {
        case 1: printf("ONE\n");
                return 0;
        case 2: printf("TWO\n");
                break;
        case 3: printf("THREE\n");
                break;
        default: printf("Out of range");
    }
    printf("Goodbye");
    return 0;
}
```



สรุปคำสั่งควบคุมแบบมีทางเลือกหรือเงื่อนไข

# ค่าของนิพจน์ตรรกศาสตร์

มีค่าเป็น **true** หรือ **false** ซึ่งอาจเกิดได้จาก

1. ข้อมูลชนิด boolean
2. การดำเนินการเปรียบเทียบ (Comparison Operators)  
โดยตัวดำเนินการเชิงสัมพันธ์ (Relational Operators)
3. การดำเนินการทางตรรกะ โดยตัวดำเนินการตรรกะ (Logical operators)

## 3.1 ข้อมูลชนิด boolean

Boolean เป็นประเภทข้อมูลที่มีได้เพียงสองค่าคือจริง true และเท็จ false เก็บค่าที่เป็นไปได้เพียงสองค่า

- 1 หรือ true สำหรับค่าที่เป็นจริง
- 0 หรือ false สำหรับค่าที่เป็นเท็จ

```
1 #include<stdio.h>
2 #include <stdbool.h>
3 int main(){
4     bool x = true;
5     printf("%d\n", x);
6     printf(x ? "true" : "false");
7     return 0;
8 }
```

ผลการรัน :

```
1
true
```

## 3.2

# ตัวดำเนินการเชิงสัมพันธ์ (Relational Operators)

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

### 3.3 ตัวดำเนินการตรรกะ (Logical operators)

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



# ลำดับในการดำเนินการ

( )	ตัวดำเนินการทางคณิตศาสตร์
++ - + - !	
* / %	
+ -	
< <= > >=	Relation Operators
== !=	
&&	Logical Operators
= += -= *= /= %=	

## 4.1

# คำสั่ง if-else

## if

```
if (เงื่อนไข)  
    คำสั่ง ;
```

```
if (เงื่อนไข) {  
    ชุดคำสั่งที่ 1;  
    ชุดคำสั่งที่ n;  
}
```

## if-else

```
if (เงื่อนไข) {  
    ชุดคำสั่งที่ 1 ;  
} else {  
    ชุดคำสั่งที่ 2 ;  
}
```

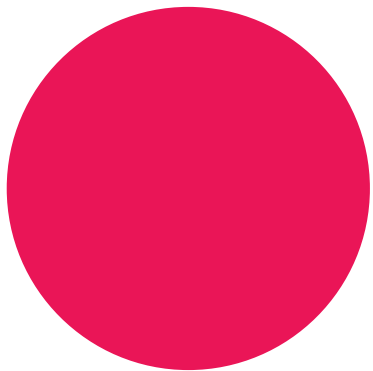
## nested-if

```
if (เงื่อนไขที่1){  
    ชุดคำสั่งที่ 1 ;  
} else if (เงื่อนไขที่ 2) {  
    ชุดคำสั่งที่ 2 ;  
} else {  
    ชุดคำสั่งที่ 3 ;  
}
```

## 4.2

## คำสั่ง switch-case

```
switch (นิพจน์ที่ต้องการตรวจสอบ)
{
    case ค่าคงที่ 1 :
        คำสั่งที่ 1 ;
        คำสั่งที่ n ;
    case ค่าคงที่ 2 :
        คำสั่งที่ 2 ;
        คำสั่งที่ n ;
    default :
        คำสั่งที่ 3 ;
        คำสั่งที่ n ;
}
```



**แบบฝึกหัด ครั้งที่ 2**

## แบบฝึกหัด ครั้งที่ 2

2.1) จงเขียนโปรแกรม รับคะแนนสอบ จำนวน 4 วิชา วิชาละ 100 คะแนน แล้วหาคะแนนเฉลี่ย โดยถ้าคะแนนที่กรอกมากกว่า 100 หรือ น้อยกว่า 0 ให้โปรแกรมแสดงข้อความว่า Invalid input!!

2.2) จงเขียนโปรแกรมรับตัวเลขจำนวน 5 ตัว แล้วแสดงผลว่าตัวเลขที่รับเข้ามาเป็นจำนวนคู่ และคี่ อย่างละกี่ตัว

2.3) จงเขียนโปรแกรมคำนวณโบนัสของพนักงาน ให้โปรแกรมรับหมายเลขประจำตัวพนักงาน จำนวน 5 หลัก ฐานเงินเดือน และโอทีที่ได้ กำหนดให้ ถ้าเงินเดือนรวมตั้งแต่ 100000 บาท ได้รับ 10% ของเงินเดือนรวม เงินเดือนตั้งแต่ 70000 บาท ได้รับ 7% เงินเดือน 50000 บาท ได้รับ 5% เงินเดือน 30000 บาท ได้รับ 3% นอกนั้นได้รับ 1%

หมายเหตุ: เป็นเงินเดือนรวม = ฐานเงินเดือน +ot

## แบบฝึกหัด ครั้งที่ 2

2.4) จงเขียนโปรแกรมรับค่าของคะแนนสอบกลางภาค 100 คะแนน และคะแนนสอบปลายภาค 100 คะแนน จากคีย์บอร์ด แล้วหาคะแนนเฉลี่ยเพื่อตัดเกรดตามเงื่อนไขต่อไปนี้

คะแนนเฉลี่ย	$80 \leq x \leq 100$	จะได้ grade = 'G'	หมายถึง	Good
	$50 \leq x < 80$	จะได้ grade = 'P'	หมายถึง	Pass
	$0 \leq x < 50$	จะได้ grade = 'F'	หมายถึง	Fail

(กำหนดให้โปรแกรมนี้จะทำงานก็ต่อเมื่อคะแนนที่กรอก ไม่น้อยกว่า 0 หรือ มากกว่า 100)

### ตัวอย่างการแสดงผล

Please enter midterm score : 80

Please enter final score : 90

Your Score = 85 %

Grade = G , Good



## แบบฝึกหัด ครั้งที่ 2

2.5) จงเขียนโปรแกรมรับค่าเลขจำนวนเต็ม (N) จากคีย์บอร์ด และพิมพ์ตัวเลขตัวสุดท้ายของ N เป็นข้อความ กำหนดให้ 0 = Zero, 1 = One, 2= Two, 3=Three , 4 =Four, 5= Five, 6 =Six, 7= Seven, 8 = Eight, 9 = Nine

ตัวอย่างการแสดงผล

Please enter number : 1062

Two

## แบบฝึกหัด ครั้งที่ 2

2.6) ให้รับอักขระ 1 ตัว โดย

- ถ้าตัวอักขระที่ป้อนเข้าไบนั้นเป็นตัวพิมพ์ใหญ่ให้แสดงผลเป็นตัวพิมพ์เล็ก
- ถ้าตัวอักขระที่ป้อนเข้าไบนั้นเป็นตัวพิมพ์เล็กให้แสดงผลเป็นตัวพิมพ์ใหญ่
- แล้วพัฒนาโปรแกรมต่อ โดยถ้าป้อนตัวอักขระเข้าไแต่ไม่ใช่ตัวอักษรให้แสดงคำว่า error



# แบบฝึกหัด ครั้งที่ 2

## ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

## แบบฝึกหัด ครั้งที่ 2

2.7) จงเขียนโปรแกรมหาพื้นที่และเส้นรอบวงของวงกลม โดยที่ผู้ใช้ป้อนรัศมี และให้เลือกว่าต้องการหาอะไร กำหนดให้ เมื่อกด 1 แสดงว่าต้องการหาพื้นที่ แต่ถากด 2 แสดงว่าต้องการหาเส้นรอบวง

### ตัวอย่างการแสดงผล

Please input radius : 3

Calculator Menu :

1. Find area
2. Find circumference

Choose menu : 1

Area = 28.26

### ตัวอย่างการแสดงผล

Please input radius : 3

Calculator Menu :

1. Find area
2. Find circumference

Choose menu : 2

Circumference = 18.84

## แบบฝึกหัด ครั้งที่ 2

2.8) จงเขียนโปรแกรมคำนวณค่าบริการตัดหญ้าของบริษัทรับตัดหญ้าแห่งหนึ่ง โดยรับพื้นที่ตัดหญ้าจากแป้นพิมพ์ (ตารางวา) แล้วแสดงอัตราค่าบริการ ซึ่งมีอัตราค่าบริการดังนี้

- พื้นที่ต่ำกว่า 80 ตารางวา คิดตารางวาละ 12.50 บาท
- พื้นที่ 80 ถึง 200 ตารางวา คิดตารางวาละ 10 บาท
- พื้นที่มากกว่า 200 ถึง 400 ตารางวา คิดตารางวาละ 7.50 บาท
- พื้นที่มากกว่า 400 ตารางวา คิดตารางวาละ 5 บาท

### ตัวอย่างการแสดงผล

Please input area : 100

Total service costs : 1000 baht

# การตั้งชื่อไฟล์

64xxx-ข้อที่

เช่น

เลขประจำตัว 64999 ข้อที่ 2-1

จะได้ 64999-2-1.c หรือ 64999-2-1.cpp