# Computer Programming I:
# การเขียนโปรแกรมคอมพิวเตอร์ I

## คำสั่งควบคุม IF, IF-ELSE และ Nested IF



อ.ดร.ปัญญนัท อ้นพงษ์

ภาควิชาคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยศิลปากร

aonpong_p@su.ac.th

# Outline


- **คำสั่งควบคุม**
  - **คำสั่ง If**
  - **คำสั่ง if-else**
  - **คำสั่ง nested if (ซ้อน if)**
- เรื่องลึกๆ ของ if-else กับการพิจารณาเงื่อนไข

# คำสั่งควบคุม


- ที่ผ่านมาเราได้เขียนโฟลวชาร์ตในหลากหลายรูปแบบ
  - แบบตรง ๆ บนลงล่างเส้นเดียว
  - แบบมีทางเลือก ซ้ายขวา
  - แบบมีการวนซ้ำ
  - แบบมีทั้งสองอย่าง
- ตอนนี้ เราได้เรียนการเขียนโปรแกรมมานิดหน่อย คำสั่งที่เราเขียนได้มีดังนี้
  - ส่วนที่ต้องเขียนอยู่แล้ว #include<stdio.h>, void main(){...ชุดคำสั่ง...}
  - ส่วนการรับข้อมูลเข้า-ส่งผลลัพธ์ออก printf, scanf
  - *ด้วยความรู้ตอนนี้เรายังสามารถเขียนโปรแกรมแบบโฟลวชาร์ตแบบแรกเท่านั้น*

# คำสั่งควบคุม


- ในวันนี้เราจะได้เรียนรู้คำสั่งที่ทำให้สามารถเขียนโปรแกรมที่มีทางเลือกได้
  - แบบตรง ๆ บนลงล่างเส้นเดียว
  - แบบมีทางเลือก ซ้ายขวา
  - แบบมีการวนซ้ำ
  - แบบมีทั้งสองอย่าง

# คำสั่งควบคุม

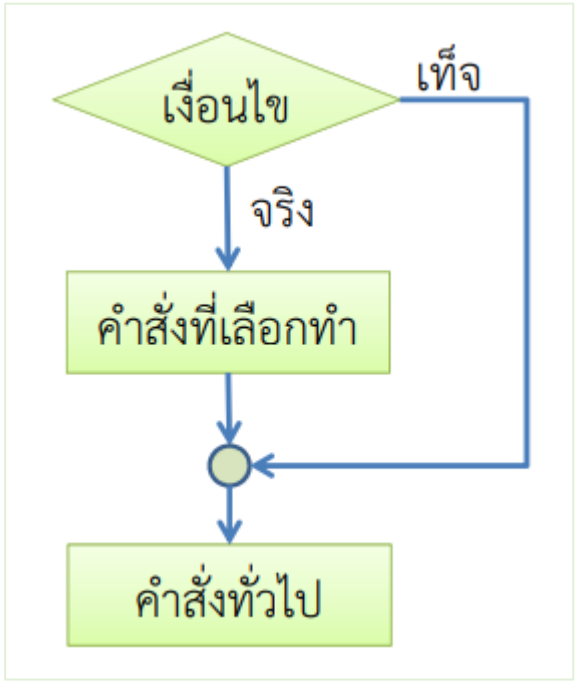

- คำสั่งควบคุม คือคำสั่งที่ทำให้โปรแกรมมีทิศทางไปในทิศทางที่กำหนด
- ด้วยคำสั่งควบคุม ทำให้เราสามารถเขียนโปรแกรมตามโฟลวชาร์ตในลักษณะต่าง ๆ ได้
- คำสั่งควบคุม มี 2 ประเภท
  - คำสั่งเงื่อนไข (Condition Statement)
    - **if-else**
    - switch-case

ใช้แทนกันได้ (แต่ต้องมีการปรับแต่งโค้ด) ถ้าถนัดอันไหนอาจเลือกใช้อันนั้นตลอดก็ได้

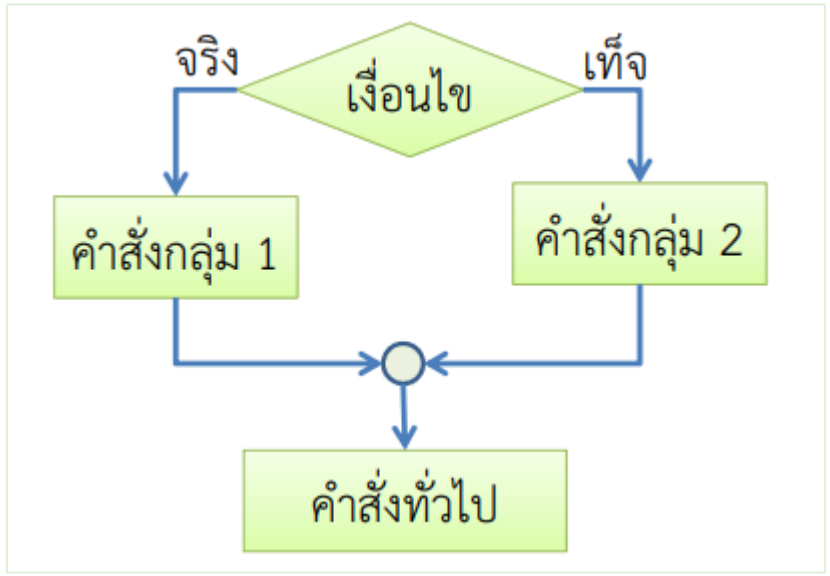
  - คำสั่งทำซ้ำ (Iteration Statement)
    - while
    - do-while
    - for

# คำสั่งเงื่อนไข if

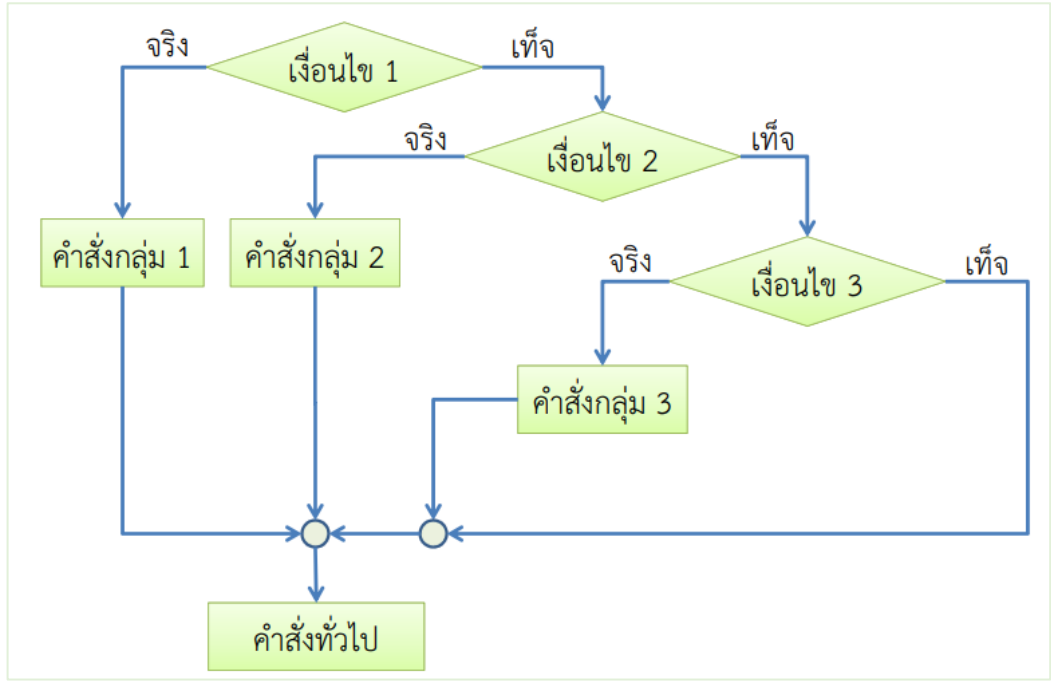
- เพื่อให้ง่ายต่อการอธิบาย จะแบ่งเงื่อนไข if แบ่งออกเป็น 3 ระดับ




if



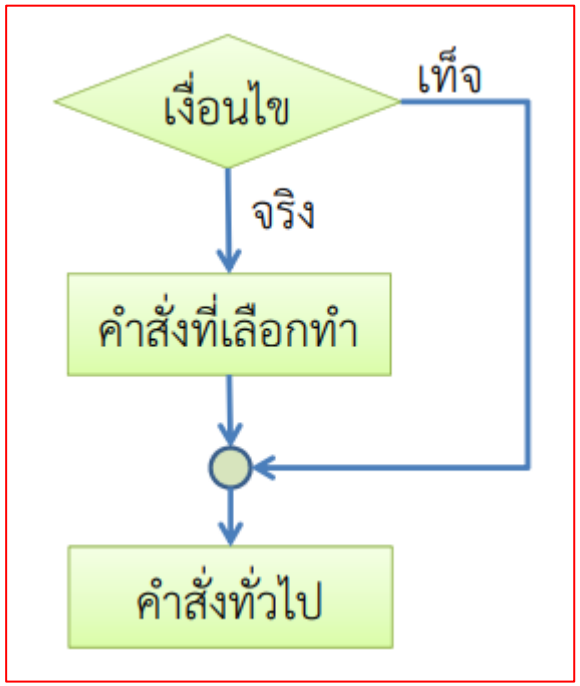

if-else




nested if

# คำสั่งเงื่อนไข if

- โดยจะเริ่มจากง่ายๆ ก่อน




if



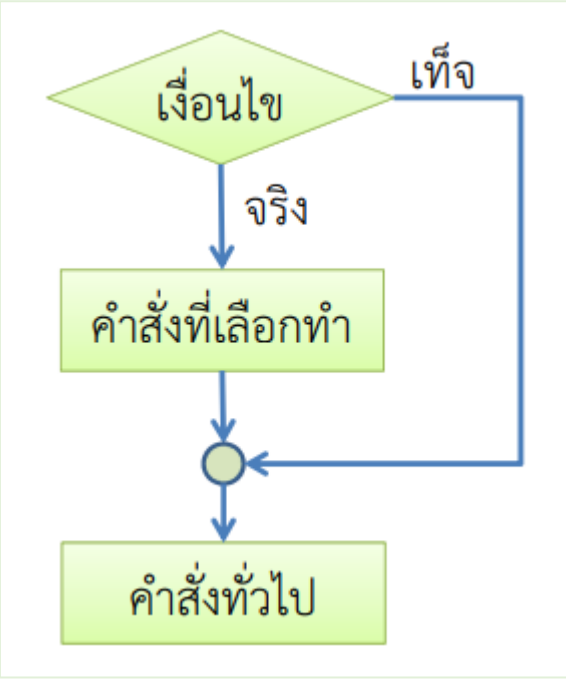

if-else




nested if

# คำสั่งเงื่อนไข if

- เป็นคำสั่งที่จะใช้ควบคุมสิ่งที่อยู่ภายในว่าให้ทำหรือไม่ทำ โดยเช็คจากเงื่อนไขที่กำหนด
  - ถ้าเงื่อนไขเป็นจริง สิ่งที่อยู่ภายในขอบเขตที่กำหนดจะถูกทำ
  - ถ้าเงื่อนไขเป็นเท็จ สิ่งที่อยู่ภายในขอบเขตที่กำหนดจะไม่ถูกทำ




```
if ( เงื่อนไข ){
คำสั่งเลือกทำ
}
คำสั่งทั่วไป
```

# คำสั่งเงื่อนไข if

**ตัวอย่างโจทย์ 1** จงเขียนโปรแกรมภาษาซี ที่พิมพ์คำว่า positive เมื่อผู้ใช้ใส่ค่าตัวเลขจำนวนเต็มที่เป็นบวก (ไม่ต้องพิมพ์อะไรถ้าไม่เป็นบวก)

**จงเขียนโฟลวชาร์ต ซูโดโค้ดและโค้ดภาษาซี**

# คำสั่งเงื่อนไข if


**ซูโดโค้ด**

```
START
READ x
IF x > 0 THEN
    PRINT “positive”
END IF
STOP
```

**ภาษาซี**

# คำสั่งเงื่อนไข if


**ซูโดโค้ด**

```
START
READ x
IF x > 0 THEN
    PRINT “positive”
END IF
STOP
```

**ภาษาซี** *//ในวิชานี้การเริ่มเขียนโปรแกรมมักจะถูกตั้งต้นแบบนี้*

```
#include<stdio.h>
void main(){




}
```

# คำสั่งเงื่อนไข if


**ซูโดโค้ด**

```
START
READ x
IF x > 0 THEN
    PRINT “positive”
END IF
STOP
```

**ภาษาซี**

```
#include<stdio.h>
void main(){
        int x; //ประกาศตัวแปร
        scanf(“%d”, &x); //รับค่า
        if (x > 0)
                printf(“positive”);

}
```

# คำสั่งเงื่อนไข if


**ตัวอย่างโจทย์ 2** จงเขียนโปรแกรมภาษาซี ที่**รับเลขจำนวนเต็มสองค่า**จากผู้ใช้ โปรแกรมนี้จะ**พิมพ์คำว่า positive เมื่อตัวเลขทั้งสองจำนวนเป็นบวก** และจะไม่พิมพ์อะไรเลยหากมีตัวเลขที่ไม่ได้เป็นบวกอยู่ด้วย

**จงเขียนโฟลวชาร์ต ซูโดโค้ดและโค้ดภาษาซี**

# คำสั่งเงื่อนไข if


**ซูโดโค้ด**

```
START
READ x, y
IF x > 0 AND y > 0 THEN
    PRINT “positive”
END IF
STOP
```

**ภาษาซี**

```
#include<stdio.h>
void main(){
        int x, y; //ประกาศตัวแปร
        scanf(“%d%d”, &x, &y); //รับค่า
        if (x > 0 && y > 0)
                printf(“positive”);

}
```

# คำสั่งเงื่อนไข if


**ตัวอย่างโจทย์ 3** จงเขียนโปรแกรมภาษาซี ที่**รับเลขจำนวนเต็มสองค่า**จากผู้ใช้ โปรแกรมนี้จะ**พิมพ์คำว่า positive เมื่อตัวเลขตัวใดตัวหนึ่งเป็นจำนวนบวก** และจะไม่พิมพ์อะไรเลยหากไม่มีตัวเลขที่เป็นจำนวนบวกอยู่ด้วย

**จงเขียนโฟลวชาร์ต ซูโดโค้ดและโค้ดภาษาซี**

# คำสั่งเงื่อนไข if


**ซูโดโค้ด**

**ภาษาซี**

# คำสั่งเงื่อนไข if


ถ้าโค้ดเป็นแบบนี้ ผลลัพธ์จะเป็นอย่างไร

```
int x, y;
scanf("%d %d", &x, &y);
if(x > 0)
    printf("positive");
if(y > 0)
    printf("positive");
```

# คำสั่งเงื่อนไข if


**ตัวอย่างโจทย์ 4** จงเขียนโปรแกรมภาษาซี ที่รับเลขจำนวนเต็มจากผู้ใช้ โปรแกรมจะพิมพ์คำว่า positive เมื่อผู้ใช้ใส่ค่าตัวเลขที่เป็นบวก และไม่ว่าผู้ใช้จะใส่เลขใดเข้ามาก่อนจบโปรแกรมให้พิมพ์คำว่า good bye

**จงเขียนโฟลวชาร์ต ซูโดโค้ดและโค้ดภาษาซี**

# คำสั่งเงื่อนไข if


**ซูโดโค้ด**

```
START
READ x, y
IF x > 0 THEN
    PRINT “positive”
END IF
PRINT “good bye”
STOP
```

**ภาษาซี**

```
#include<stdio.h>
void main(){
        int x, y; //ประกาศตัวแปร
        scanf(“%d%d”, &x, &y); //รับค่า
        if (x > 0)
                printf(“positive\n”);

        printf(“good bye”);
}
```

# คำสั่งเงื่อนไข if

**ภาษาซี**

```
#include<stdio.h>
void main(){
        int x, y; //ประกาศตัวแปร
        scanf("%d%d", &x, &y); //รับค่า
        if (x > 0)
                printf("positive\n");


        printf("good bye");
}
```

ให้สังเกตลำดับคำสั่งทั้งสองตัวนี้

- สังเกตว่าภายใน if จะมีสองทางเลือก คือทำ/ไม่ทำ
- สิ่งที่อยู่ข้างนอกคือทำเสมอ หรือก็คือไม่ขึ้นอยู่กับ if แล้วนั่นเอง

# คำสั่งเงื่อนไข if


**ถ้าต้องการให้ if มีคำสั่งมากกว่า 1 อย่าง**

การทำงานของ if นั้น จะเลือกทำเฉพาะ**คำสั่งที่อยู่ใต้ประโยค if(เงื่อนไข)** เพียงคำสั่งเดียวเท่านั้น ตัวอย่าง (จากโค้ดที่ได้เห็นมาแล้ว)

```
if(x > 0)
        printf("positive");
printf("Good bye");
```

แบบนี้สิ่งที่มีโอกาสจะถูก “ไม่ทำ” มีเพียงบรรทัดเดียว คือ printf(“positive”);

ส่วนบรรทัด printf(“Good bye”); จะถูกทำเสมอ ไม่ว่า x จะมีค่ามากกว่า 0 หรือไม่

# คำสั่งเงื่อนไข if


**ถ้าต้องการให้ if มีคำสั่งมากกว่า 1 อย่าง**

แต่ถ้าโจทย์บอกว่า

หาก**เลขทั้งสองที่ผู้ใช้ใส่เข้ามาเป็นบวกทั้งคู่** ให้ โปรแกรม

(1) พิมพ์คำว่า positive

(2) หาผลบวก

(3) พิมพ์ผลบวกของเลขที่นำเข้ามา

ถ้าเป็นแบบนี้จะทำยังไง ?

# คำสั่งเงื่อนไข if

**ถ้าต้องการให้ if มีคำสั่งมากกว่า 1 อย่าง**

แต่ถ้าโจทย์บอกว่า

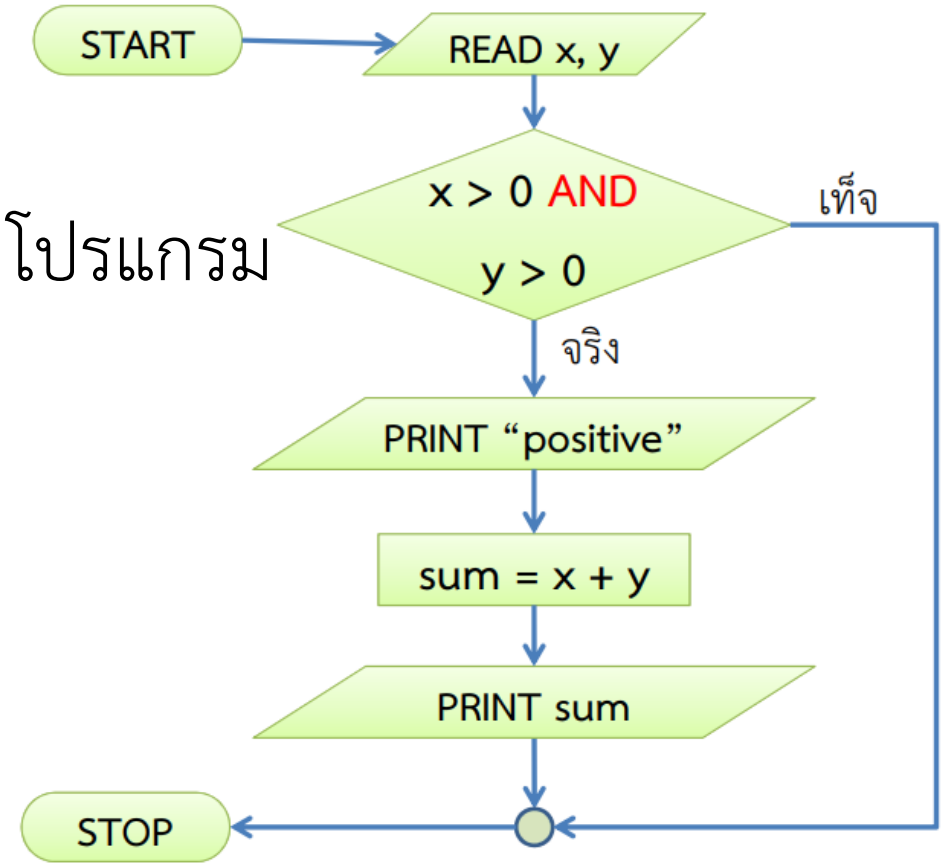
หาก**เลขทั้งสองที่ผู้ใช้ใส่เข้ามาเป็นบวกทั้งคู่** ให้ โปรแกรม

(1) พิมพ์คำว่า positive

(2) หาผลบวก

(3) พิมพ์ผลบวกของเลขที่นำเข้ามา

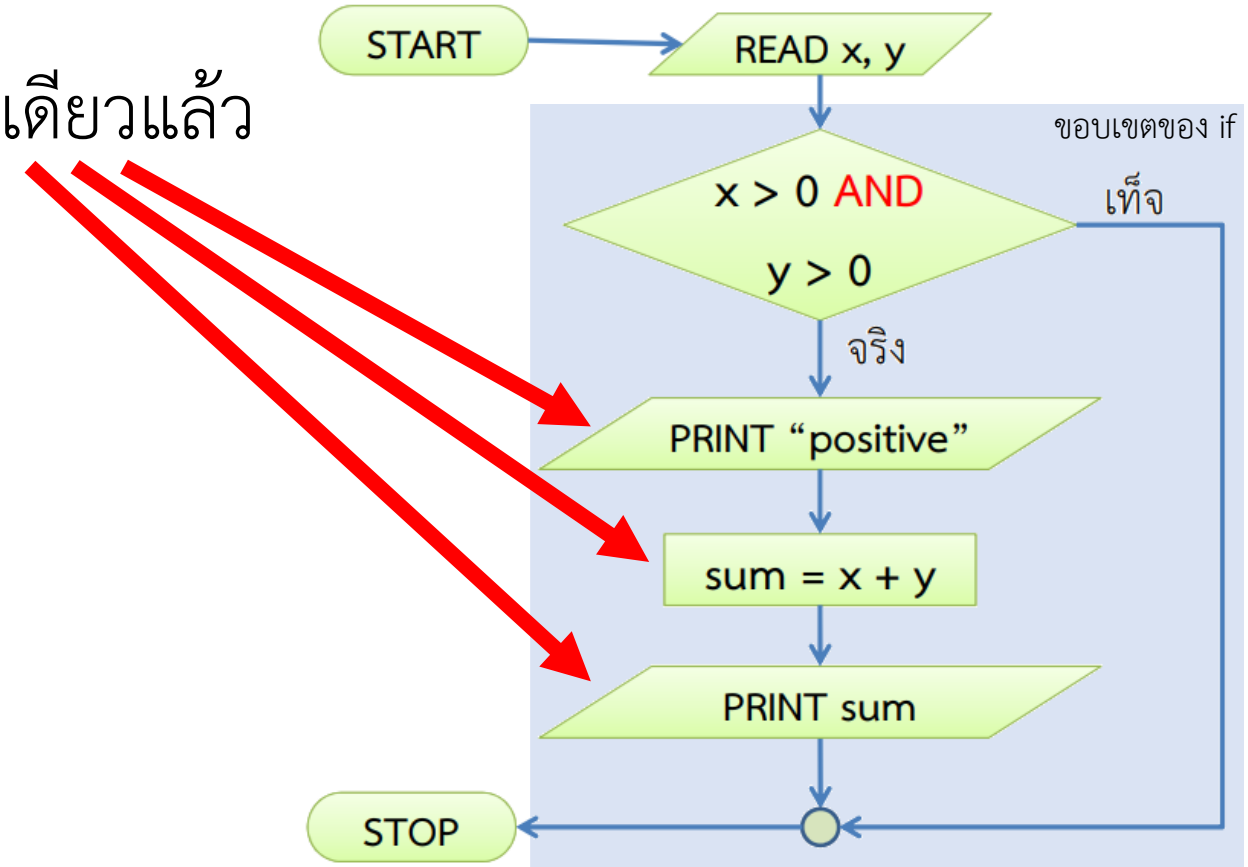
ถ้าเป็นแบบนี้จะทำยังไง ?



# คำสั่งเงื่อนไข if

**ถ้าต้องการให้ if มีคำสั่งมากกว่า 1 อย่าง**

สังเกตว่าตอนนี้ใน if ไม่ได้มีเพียงคำสั่งเดียวแล้ว



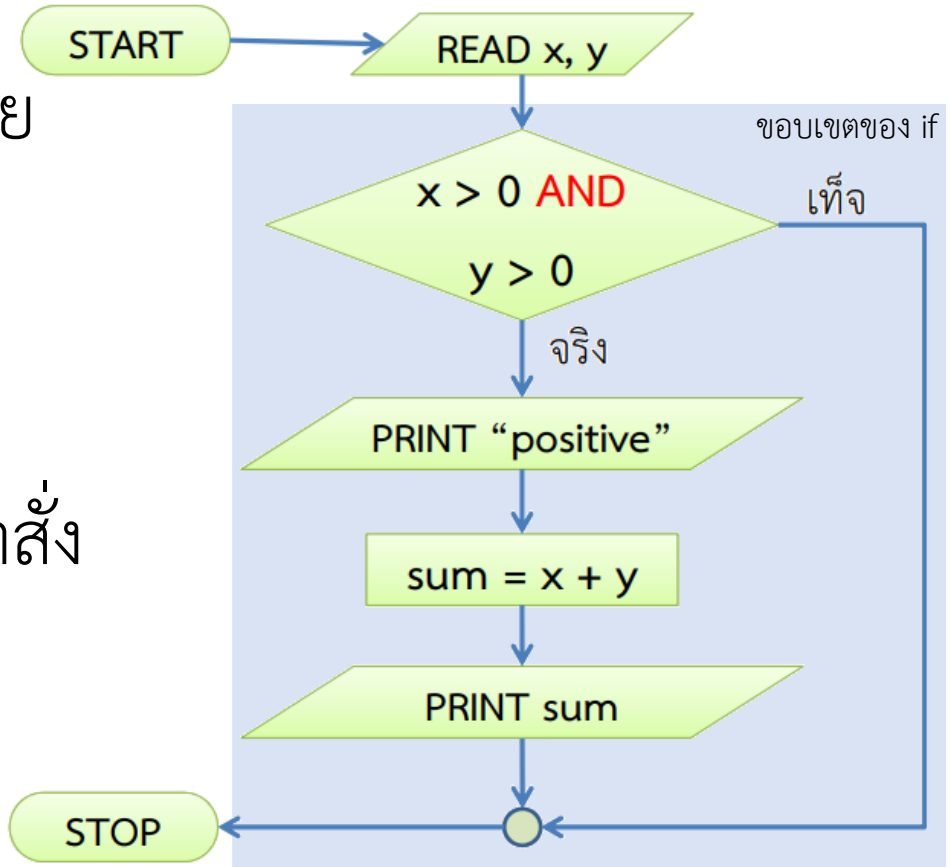
# คำสั่งเงื่อนไข if

**ถ้าต้องการให้ if มีคำสั่งมากกว่า 1 อย่าง**

ถ้าเป็นแบบนี้ คำสั่งที่ตามหลัง if จะใช้เครื่องหมาย

**{ ... }** ในการรวบคำสั่งเข้าด้วยกัน

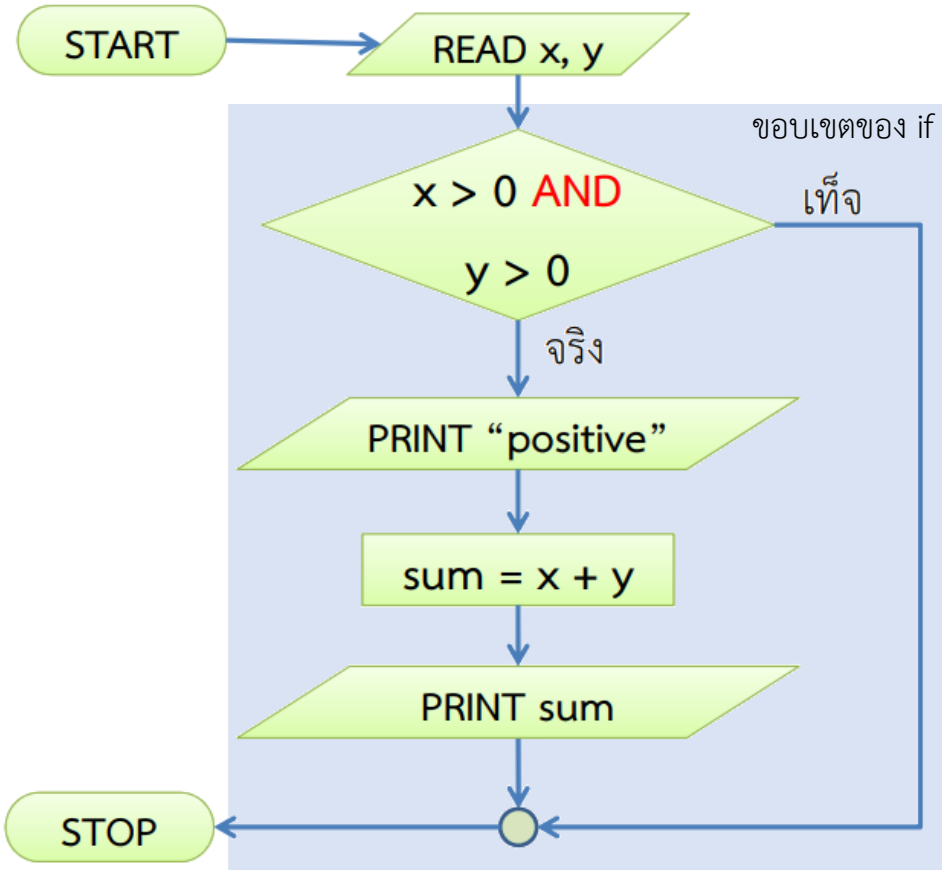
if จะมองว่าสิ่งที่ตามมาคือคำสั่งทั้งก้อน (จะมีกี่คำสั่งก็ได้แล้ว)



# คำสั่งเงื่อนไข if

ถ้าต้องการให้ if มีคำสั่งมากกว่า 1 อย่าง

```
#include<stdio.h>
void main(){
        int x, y, sum;
        scanf("%d%d", &x, &y);
        if(x > 0 && y > 0) {
                printf("positive");
                sum = x + y;
                printf("%d", sum);
        }
}
```



# คำสั่งเงื่อนไข if


- ในขณะเดียวกัน แม้ว่าเราจะมีคำสั่งใน if เพียงคำสั่งเดียว ก็สามารถใช้เครื่องหมาย { ... } ครอบได้เช่นกัน
- จริงๆแล้วการใส่เครื่องหมาย { ... } ทุกครั้งก็เป็นสิ่งที่ควรทำ (แม้จะไม่ใช่เรื่องบังคับ) เพราะหลายครั้งที่เรามาเพิ่มคำสั่งใน if ทีหลัง จะสะดวกกว่าและไม่ทำให้สับสนเท่า
- และถ้าเราใส่เครื่องหมาย { ... } เสมอ เราก็ไม่ต้องมาคอยกังวลว่าจะต้องใส่เครื่องหมายดี หรือไม่ใส่เครื่องหมายดี (ใส่ลูกเดียวเลย)

# คำสั่งเงื่อนไข if


```
#include<stdio.h>
void main(){
        int x, y; //ประกาศตัวแปร
        scanf("%d%d", &x, &y); //รับค่า
        if (x > 0)
        {
                printf("positive\n");
        }
        printf("good bye");
}
```

# คำสั่งเงื่อนไข if


**จุดที่ควรระมัดระวัง: ลองพิจารณาโค้ดต่อไปนี้**

```
#include<stdio.h>
void main(){
        int x, y, sum;
        scanf("%d%d", &x, &y);
        if(x > 0 && y > 0) {
                printf("positive");
                sum = x + y;
                printf("%d", sum);
        }
}
```

```
#include<stdio.h>
void main(){
        int x, y, sum;
        scanf("%d%d", &x, &y);
        if(x > 0 && y > 0)
                printf("positive");
                sum = x + y;
                printf("%d", sum);

}
```

# คำสั่งเงื่อนไข if


- โค้ดทั้งสองกรอบในหน้าที่แล้วทำงานไม่เหมือนกัน
- กรอบสีเขียว (ซ้าย) จะมีการเลือกทำ/ไม่ทำทุกคำสั่งในเครื่องหมาย { ... } (คือมองเป็นคำสั่งทั้งก้อน)
- กรอบสีส้ม (ขวา) จะมีการเลือกทำ/ไม่ทำเฉพาะคำสั่ง printf(“positive”); เท่านั้น เพราะเป็นคำสั่งที่ติดกับ if

- การแท็บไม่ได้มีผลต่อโค้ดในภาษาซี ดังนั้นถ้ามาดูโค้ดในตัวอย่างกันดีๆอีกครั้ง จะได้ดังนี้

# คำสั่งเงื่อนไข if


**จุดที่ควรระมัดระวัง: ทั้งสองโค้ดนี้มีการทำงานเหมือนกัน** (Tab ไม่ส่งผลต่อการจัดกลุ่มโค้ด แค่ทำให้ดูง่ายขึ้น)

```
#include<stdio.h>
void main(){
        int x, y, sum;
        scanf("%d%d", &x, &y);
        if(x > 0 && y > 0) {
                printf("positive");
                sum = x + y;
                printf("%d", sum);
        }
}
```

```
#include<stdio.h>
void main(){
int x, y, sum;
scanf("%d%d", &x, &y);
if(x > 0 && y > 0) {
printf("positive");
sum = x + y;
printf("%d", sum);
}
}
```

# คำสั่งเงื่อนไข if


**จุดที่ควรระมัดระวัง: ทั้งสองโค้ดนี้มีการทำงานเหมือนกัน** (Tab ไม่ส่งผลต่อการจัดกลุ่มโค้ด แค่ทำให้ดูง่ายขึ้น)

```
#include<stdio.h>
void main(){
        int x, y, sum;
        scanf("%d%d", &x, &y);
        if(x > 0 && y > 0)
                printf("positive");
                sum = x + y;
                printf("%d", sum);

}
```

```
#include<stdio.h>
void main(){
int x, y, sum;
scanf("%d%d", &x, &y);
if(x > 0 && y > 0)
printf("positive");
sum = x + y;
printf("%d", sum);

}
```

# คำสั่งเงื่อนไข if

**จุดที่ควรระมัดระวัง: ทั้งสองโค้ดนี้มีการทำงานเหมือนกัน** (Tab ไม่ส่งผลต่อการจัดกลุ่มโค้ด แค่ทำให้ดูง่ายขึ้น)

```
#include<stdio.h>
void main(){
        int x, y, sum;
        scanf("%d%d", &x, &y);
        if(x > 0 && y > 0)
                printf("positive");
                sum = x + y;
                printf("%d", sum);

}
```

```
#include<stdio.h>
void main(){
        int x, y, sum;
        scanf("%d%d", &x, &y);
        if(x > 0 && y > 0)
                printf("positive");
        sum = x + y;
        printf("%d", sum);

}
```

# คำสั่งเงื่อนไข if


**ตัวอย่างโจทย์ 5** จงเขียนโปรแกรมภาษาซีที่รับค่าจำนวนเต็มจากผู้ใช้มาค่าหนึ่ง หากตัวเลขนั้นหารด้วย 9 ไม่ลงตัว ให้พิมพ์คำว่า not divisible ไม่เช่นนั้นก็ให้จบการทำงานของโปรแกรมโดยไม่ต้องพิมพ์ข้อความใด ๆ ออกมา

**วิเคราะห์โจทย์**

**ข้อมูลนำเข้า** เลขจำนวนเต็ม 1 ตัว

**ข้อมูลส่งออก** แบ่งเป็น 2 กรณี ถ้าหาร 9 ไม่ลงตัว จะพิมพ์คำว่า “not divisible”

ถ้าหาร 9 ลงตัว จะไม่มีข้อมูลส่งออก

# คำสั่งเงื่อนไข if


**ตัวอย่างโจทย์ 5** จงเขียนโปรแกรมภาษาซีที่รับค่าจำนวนเต็มจากผู้ใช้มาค่าหนึ่ง หากตัวเลขนั้นหารด้วย 9 ไม่ลงตัว ให้พิมพ์คำว่า not divisible ไม่เช่นนั้นก็ให้จบการทำงานของโปรแกรมโดยไม่ต้องพิมพ์ข้อความใด ๆ ออกมา

**วิเคราะห์โจทย์**

**จะเอาอะไรมาใช้เป็นเงื่อนไข**

หาร 9 ไม่ลงตัวคืออะไร? นักศึกษาคิดว่าเงื่อนไขใดต่อไปนี้ใช้ได้บ้าง

(1) x%9 != 0 (2) x%9 == 1 (3) !(x%9 == 0) (4) x%9

# คำสั่งเงื่อนไข if


```c
#include <stdio.h>
void main() {
    int x;
    scanf("%d", &x);
    if (x % 9 != 0) {
        printf("not divisible");
    }
}
```

```c
#include <stdio.h>
void main() {
    int x;
    scanf("%d", &x);
    if( !(x % 9 == 0) ) {
        printf("not divisible");
    }
}
```

# คำสั่งเงื่อนไข if


**ตัวอย่างโจทย์ 6** จงเขียนโปรแกรมภาษาซีที่รับค่าจำนวนเต็มจากผู้ใช้มาสองจำนวน หากตัวเลขที่รับมาตัวใดตัวหนึ่งแต่ไม่ใช่ทั้งสองเป็นบวก โปรแกรมจะพิมพ์คำว่า one-positive ไม่เช่นนั้นโปรแกรมจะจบการทำงานโดยไม่พิมพ์อะไรออกมา

**วิเคราะห์โจทย์**

**ข้อนี้มีวิธีการทำที่หลากหลายมากสุดแต่จะสรรค์สร้าง**

# คำสั่งเงื่อนไข if


**ตัวอย่างโจทย์ 6** จงเขียนโปรแกรมภาษาซีที่รับค่าจำนวนเต็มจากผู้ใช้มาสองจำนวน หากตัวเลขที่รับมาตัวใดตัวหนึ่งแต่ไม่ใช่ทั้งสองเป็นบวก โปรแกรมจะพิมพ์คำว่า one-positive ไม่เช่นนั้นโปรแกรมจะจบการทำงานโดยไม่พิมพ์อะไรออกมา

**วิเคราะห์โจทย์**

แนวคิดที่ 1 เมื่อข้อมูลนำเข้ามีเพียง 2 ตัว เราอาจจะเขียนโปรแกรมโดยใช้แนวคิดตรรกศาสตร์ตรงๆเลยก็ได้ โดยถ้าเขียนประโยคทางตรรกศาสตร์จะได้ว่า $(p \wedge \sim q) \vee (\sim p \wedge q)$

```
if ((x > 0 && y < 0)||(x < 0 && y > 0))
    printf("one-positive");
```

ใช้ประโยคนี้ได้หรือไม่? *(ไม่ได้!)*

```
if (x > 0 || y > 0)
    printf("one-positive");
```

# คำสั่งเงื่อนไข if


```
#include<stdio.h>
void main(){
        int x, y;
        scanf("%d%d", &x, &y);
        if ((x > 0 && y < 0)||(x < 0 && y > 0)) {
                printf("one-positive");
        }
}
```

# คำสั่งเงื่อนไข if


**แนวคิดที่ 1+ ใช้ตรรกศาสตร์เหมือนเดิม แต่เราสามารถเก็บค่าทางตรรกศาสตร์แบบนี้ก็ได้**

จาก $(p \wedge \sim q) \vee (\sim p \wedge q)$

**โค้ดภาษาซี**

```
p = x > 0;
q = y > 0;
if ((p && !q)||(!p && q))
        printf("one-positive");
```

# คำสั่งเงื่อนไข if

```
#include<stdio.h>
void main(){
        int x, y;
        scanf("%d%d", &x, &y);
        int p = x > 0;
        int q = y > 0;
        if ((p && !q)||(!p && q)) {
                printf("one-positive");
        }
}
```

(x>0 && !(y>0)) || (!(x>0) && y>0)

# คำสั่งเงื่อนไข if


**แนวคิดที่ 2  ใช้การนับ ว่ามีเลขที่มากกว่า 0 อยู่กี่ตัว**

วิธีนี้เป็นอีกทางเลือกหนึ่งที่สามารถทำได้

**ข้อเสีย**

ถ้าเป็นโจทย์ข้อนี้ที่ต้องการตรวจสอบข้อมูลเพียงสองตัว การทำด้วยแนวคิดนี้อาจไม่เหมาะสมเท่าวิธีการดำเนินการด้วยตรรกศาสตร์ เพราะอาจต้องคิดซับซ้อนกว่า

**ข้อดี**

ถ้าเปลี่ยนโจทย์ให้ซับซ้อนขึ้น เช่นต้องการตรวจสอบข้อมูลมากกว่าสองตัว การแก้ไขโค้ดในลักษณะนี้จะง่ายกว่ามาก

# คำสั่งเงื่อนไข if

**แนวคิดที่ 2 ใช้การนับ ว่ามีเลขที่มากกว่า 0 อยู่กี่ตัว (ถ้าตัวเดียวค่อยพิมพ์ค่า)**

```
#include<stdio.h>
void main(){
        int x, y, count=0;
        scanf("%d%d", &x, &y);
        if (x > 0){
                count++;
        }
        if (y > 0){
                count++;
        }
//ต่อคอลลัมน์ขวา
        if(count == 1){
                printf("one-positive");
        }
}
```

# คำสั่งเงื่อนไข if

สมมติว่าถ้ามีโจทย์อีกข้อย่อยหนึ่ง บอกว่าทั้งสองค่าต้องเป็นจำนวนบวก เราก็แก้เลขแค่ตัวเดียว

```
#include<stdio.h>
void main(){
        int x, y, count=0;
        scanf("%d%d", &x, &y);
        if (x > 0){
                count++;
        }
        if (y > 0){
                count++;
        }
//ต่อคอลลัมน์ขวา
        if(count == 2){
                printf("one-positive");
        }
}
```

# คำสั่งเงื่อนไข if

สมมติว่าถ้ามีโจทย์อีกข้อย่อยหนึ่ง บอกให้รับค่าเข้ามา 4 ค่า และต้องมี 2 ค่าพอดีที่เป็นจำนวนบวก จึงจะพิมพ์ “two-positive” ออกมา

```
#include<stdio.h>
void main(){
        int w, x, y, z, count=0;
        scanf(“%d%d”, &x, &y);
        if (w > 0){
                count++;
        }
        if (x > 0){
                count++;
        }
//ต่อคอลลัมน์ขวา
        if (y > 0){
                count++;
        }
        if (z > 0){
                count++;
        }
        if(count == 2){
                printf(“two-positive”);
        }
}
```

# คำสั่งเงื่อนไข if


**สมมติว่าถ้ามีโจทย์อีกข้อย่อยหนึ่ง บอกให้รับค่าเข้ามา 4 ค่า และต้องมี 2 ค่าพอดีที่เป็นจำนวนบวก จึงจะพิมพ์ “two-positive” ออกมา**

**คำถาม** ถ้าเป็นวิธีที่ใช้เทคนิคทางตรรกศาสตร์ (แนวคิดที่ 1 และ 1+) จะสามารถทำโจทย์ข้อนี้ได้หรือไม่?

# สรุป: คำสั่งเงื่อนไข if

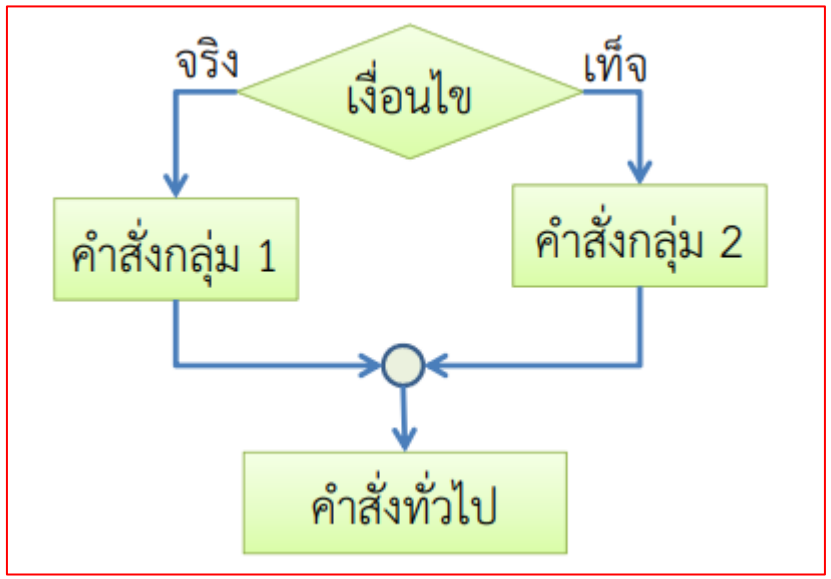

- คำสั่ง if เป็นคำสั่งที่ควบคุมการทำงาน/ไม่ทำงานของคำสั่งที่อยู่ภายในขอบเขต
- สามารถเรียกใช้ได้โดย if(เงื่อนไข){คำสั่ง} โดยคำสั่งในเครื่องหมายปีกกาจะทำงานเมื่อเงื่อนไขในวงเล็บเป็นจริงเท่านั้น
- คำสั่งอื่นๆที่อยู่นอกเครื่องหมายปีกกาจะทำงานตามปกติ (ถือว่าไม่ได้อยู่ในขอบเขตควบคุมของ if)
- ถ้าไม่มีเครื่องหมายปีกกา ถือว่าคำสั่งที่อยู่ติดกับคำสั่ง if เป็นคำสั่งที่อยู่ในขอบเขตควบคุมของ if เพียงคำสั่งเดียว โดยไม่สนใจการย่อหน้าหรือการขึ้นบรรทัดใหม่
- สามารถใช้ operator ทางตรรกศาสตร์ในเงื่อนไขได้ (&&, || หรือ !) เพื่อทำให้โค้ดสั้นลง
- บางครั้งการยอมให้โค้ดยาวโดยแยกเงื่อนไขออกมาอาจทำให้ผู้เขียนโปรแกรมมองได้ง่ายกว่า แต่ก็เกี่ยวกับความชำนาญของคนๆนั้นด้วย ดังนั้นให้เลือกวิธีที่ทำให้เรามั่นใจมากกว่า

# คำสั่งเงื่อนไข if-else

- ตอนนี้เรากำลังจะขยับไปทำในกรณีที่มีคำสั่งที่ต้องทำงานเมื่อเงื่อนไขเป็นเท็จด้วย




if



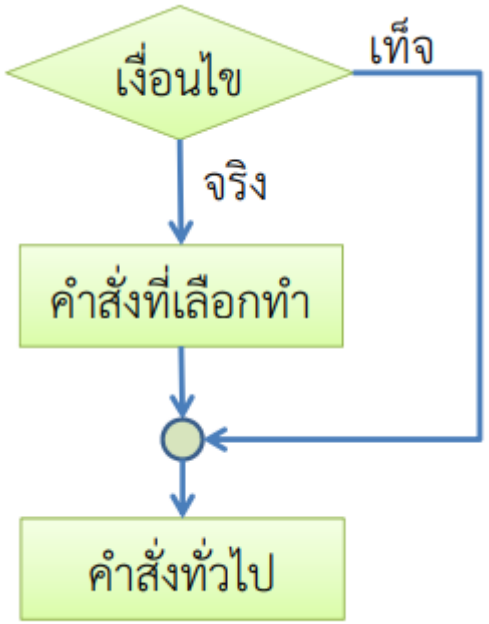

if-else

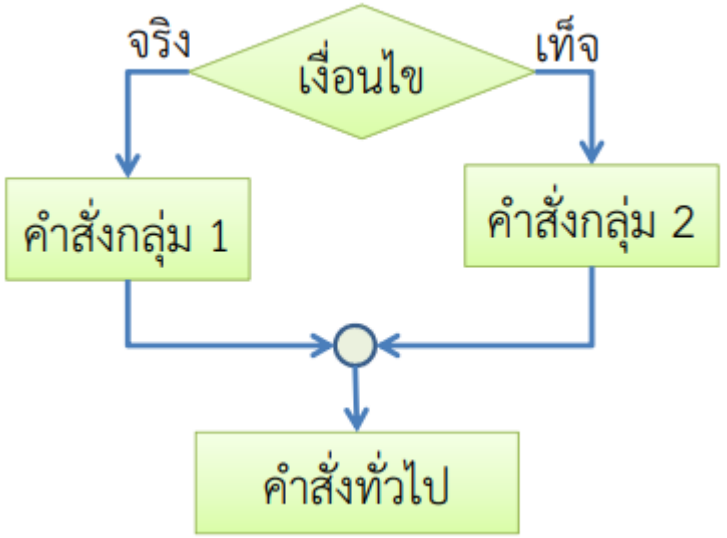



nested if

# คำสั่งเงื่อนไข if-else

- เปรียบเทียบ if แบบโดดๆ กับ if-else

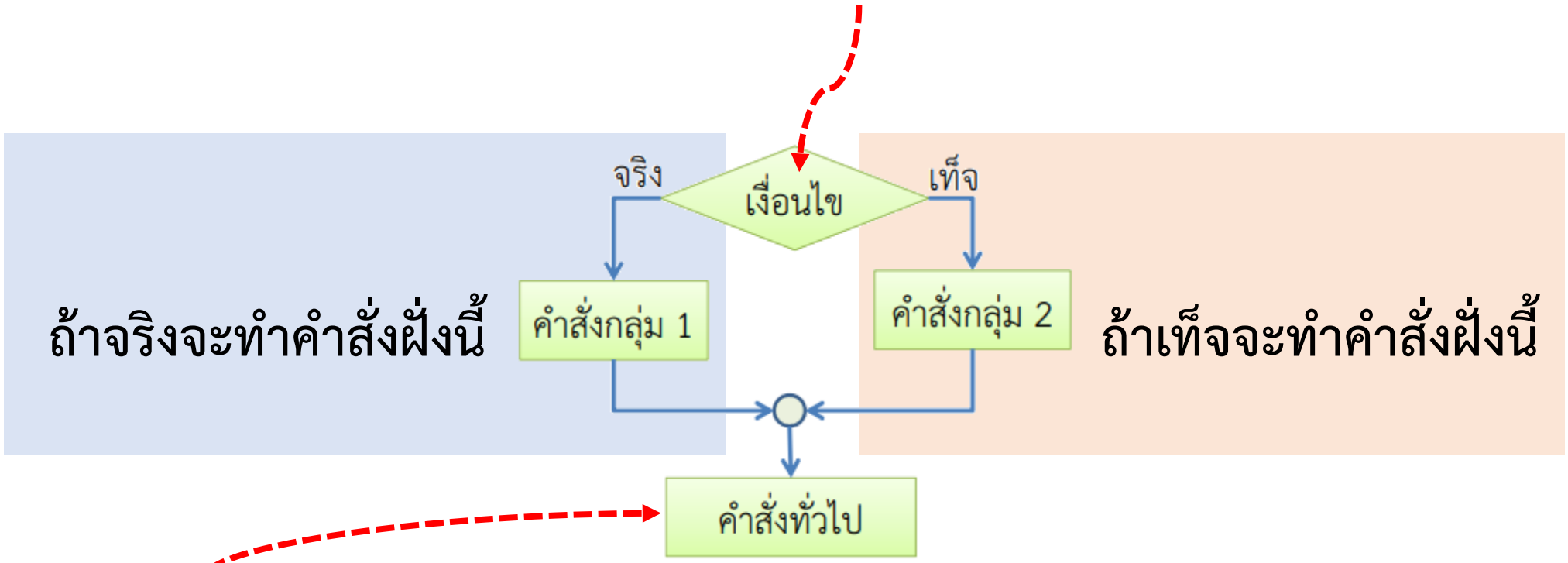



if สามารถเลือก ทำ/ไม่ทำ คำสั่งที่อยู่ในขอบเขตได้ **if-else สามารถเลือกเส้นทางที่จะทำคำสั่งได้**

# คำสั่งเงื่อนไข if-else

- ขั้นแรกคอมไพเลอร์จะตรวจสอบเงื่อนไขตรงนี้



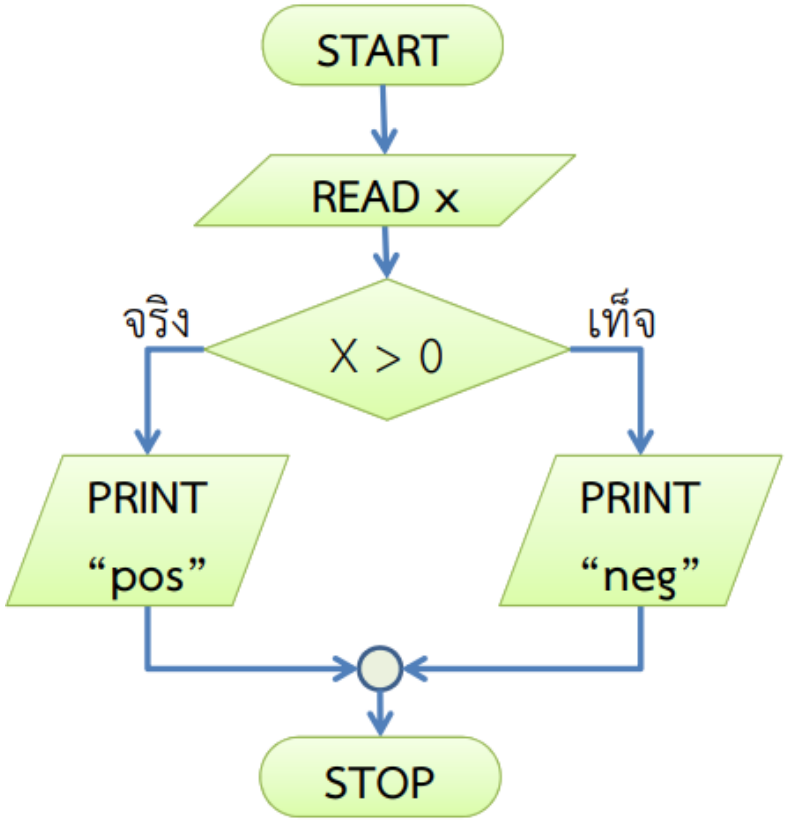

ส่วน**คำสั่งทั่วไป**ที่อยู่ล่างสุดนั้นอยู่นอกเหนือการควบคุมของ if-else (ไม่เกี่ยวกับเงื่อนไขแล้ว ยังไงก็ต้องทำ)

# คำสั่งเงื่อนไข if-else


- **ตัวอย่างโจทย์** จงเขียนโปรแกรมที่รับค่าตัวเลขจากผู้ใช้เข้ามาหนึ่งตัว หากตัวเลขเป็นบวกให้พิมพ์คำว่า pos ไม่เช่นนั้นให้พิมพ์ว่า neg

# คำสั่งเงื่อนไข if-else

- **ตัวอย่างโจทย์** จงเขียนโปรแกรมที่รับค่าตัวเลขจากผู้ใช้เข้ามาหนึ่งตัว หากตัวเลขเป็นบวกให้พิมพ์คำว่า pos ไม่เช่นนั้นให้พิมพ์ว่า neg




```
#include<stdio.h>
void main() {
        int x;
        scanf("%d", &x);
        if (x > 0)
                printf("pos");
        else
                printf("neg");
}
```

# คำสั่งเงื่อนไข if-else


- **คล้ายกับคำสั่ง if โดดๆ หนึ่งอย่างคือ ถ้าไม่มีเครื่องหมายปีกกาด้านหลัง if หรือ else คำสั่งที่อยู่ในขอบเขตของ if หรือ else จะเป็นคำสั่งที่ติดกับคำสั่งเงื่อนไขนั้นเพียงคำสั่งเดียว**
- เช่นจากตัวอย่างโค้ด แบบนี้คำสั่งที่อยู่ในขอบเขตของ if (x > 0) คือ printf(“pos”); เพียงคำสั่งเดียว และคำสั่งที่อยู่ในขอบเขตของ else คือ printf(“neg”); เพียงคำสั่งเดียว
- ถ้าต้องการให้ใน if หรือ else มีหลายคำสั่งก็สามารถใส่เครื่องหมายปีกกาลงไปเพื่อกำกับขอบเขตของชุดคำสั่งได้ (แม้จะมีคำสั่งเดียวก็ทำได้)

```
#include<stdio.h>
void main() {
        int x;
        scanf("%d", &x);
        if (x > 0)
                printf("pos");
        else
                printf("neg");
}
```

# คำสั่งเงื่อนไข if-else : โค้ดที่ทำงานเหมือนกัน


```
#include<stdio.h>
void main() {
        int x;
        scanf("%d", &x);
        if (x > 0)
                printf("pos");
        else
                printf("neg");
}
```

```
#include<stdio.h>
void main() {
        int x;
        scanf("%d", &x);
        if (x > 0) {
                printf("pos");
        }
        else {
                printf("neg");
        }
}
```

```
#include<stdio.h>
void main() {
        int x;
        scanf("%d", &x);
        if (x > 0)
        printf("pos");
        else
        printf("neg");
}
```

# คำสั่งเงื่อนไข if-else : โค้ดที่ทำงานเหมือนกัน


```
#include<stdio.h>
void main() {
        int x;
        scanf("%d", &x);
        if (x > 0)    printf("pos");
        else    printf("neg");
}
```

```
#include<stdio.h>
void main() {
int x;
scanf("%d", &x);
if (x > 0)
printf("pos");
else
printf("neg");
}
```

# คำสั่งเงื่อนไข if-else: สิ่งที่มักเกิดความผิดพลาด

- **ต้องการให้ if หรือ else ทำงานมากกว่า 1 อย่าง แต่ลืมใส่เครื่องหมายปีกกา**

```c
#include <stdio.h>

void main() {
    float x, y;
    scanf("%f", &x);
    if (x > 0)
        printf("pos");
        printf("I'm bad");
    else
        printf("neg");

}
```

ระหว่าง if-else จะมีอะไรมาขั้นไม่ได้

```
main.c: In function 'main':
main.c:17:5: error: 'else' without a previous 'if'
   17 |     else
      |     ^~~~
```

แบบนี้จะทำให้คอมไพเลอร์แจ้ง error เพราะมีคำสั่งไปขั้นระหว่าง if และ else

# คำสั่งเงื่อนไข if-else: สิ่งที่มักเกิดความผิดพลาด

- **ต้องการให้ if หรือ else ทำงานมากกว่า 1 อย่าง แต่ลืมใส่เครื่องหมายปีกกา**

```c
#include <stdio.h>

void main() {
    float x, y;
    scanf("%f", &x);
    if (x > 0)
        printf("pos");
    else
        printf("neg");
        printf("I'm bad");

}
```

Input ค่า x เป็น 5

แม้ว่าจะเข้าเงื่อนไขใน if แต่ก็ยังพิมพ์ I'm bad ออกมา

```
5
posI'm bad

...Program finished with exit code 0
Press ENTER to exit console.
```

ผู้เขียนโปรแกรมนี้น่าจะตั้งใจให้คำสั่งนี้อยู่ใน else (ตั้งใจให้พิมพ์เมื่อ x <= 0 เท่านั้น)

แบบนี้ไม่ error แต่บรรทัด printf("I'm bad"); จะถือว่าอยู่นอกการควบคุมของ if-else

# คำสั่งเงื่อนไข if-else: สิ่งที่มักเกิดความผิดพลาด

- **แนวทางแก้ไข : ใส่ปีกกาทุกครั้งไปเลย แม้ว่าจะมีเพียงคำสั่งเดียวก็ตาม**

```c
#include <stdio.h>

void main() {
    float x, y;
    scanf("%f", &x);
    if (x > 0){
        printf("pos");
    }
    else{
        printf("neg");
        printf("I'm bad");
    }
}
```

ใส่ปีกกา { ... }

```
5
pos

...Program finished with exit code 0
Press ENTER to exit console.
```

```
-3
negI'm bad

...Program finished with exit code 0
Press ENTER to exit console.
```

# คำสั่งเงื่อนไข if-else


- **ตัวอย่างโจทย์ 7** จงเขียนโปรแกรมรับเลขทศนิยมแบบ single precision สองจำนวน หากตัวเลขที่สองเป็นบวกให้หาผลบวกของเลขทั้งสอง แล้วพิมพ์ผลลัพธ์ออกมา หากตัวเลขตัวที่สองเป็นศูนย์หรือเป็นลบ ให้หาผลคูณของเลขทั้งสองแล้วพิมพ์ผลลัพธ์ออกมา

วิเคราะห์โจทย์ อะไรคือข้อมูลขาเข้า ขาออก โฟลวชาร์ตเป็นอย่างไร

** เราต้องแยกให้ออกว่า else ที่เราคิดใช้นั้นมันเป็นตัวแทนของอีกกรณีหนึ่งได้จริงหรือไม่ ถ้าไม่ใช่หรือไม่แน่ใจให้ใช้ nested if ที่จะสอนต่อไป

# คำสั่งเงื่อนไข if-else


```c
#include <stdio.h>

void main() {
    float x, y, result;
    scanf("%f %f", &x, &y);
    if(y > 0) {
        result = x + y;
        printf("%f", result);
    } else {
        result = x * y;
        printf("%f", result);
    }

}
```

```c
#include <stdio.h>

void main() {
    float x, y;
    scanf("%f %f", &x, &y);
    if(y > 0) {
        printf("%f", x + y);
    } else {
        printf("%f", x * y);
    }
}
```

# คำสั่งเงื่อนไข if-else

```c
#include <stdio.h>

void main() {
    float x, y, result;
    scanf("%f %f", &x, &y);
    if(y > 0) {
        result = x + y;
    } else {
        result = x * y;
    }
    printf("%f", result);
}
```

```
6 4
10.000000

...Program finished with exit code 0
Press ENTER to exit console.
```

```
9 -5
-45.000000

...Program finished with exit code 0
Press ENTER to exit console.
```

# คำสั่งเงื่อนไข nested-if

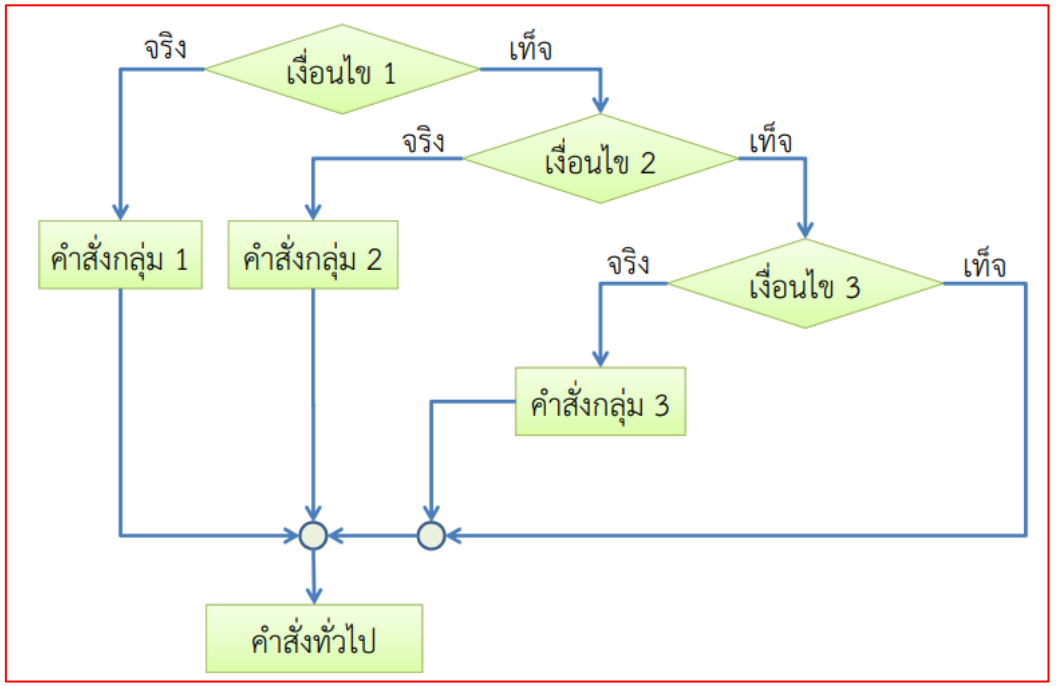
- Nested-if คือคำสั่งเงื่อนไขที่เหมือนจะดูซับซ้อนกว่า if และ if-else ที่อธิบายไปก่อนหน้า




if



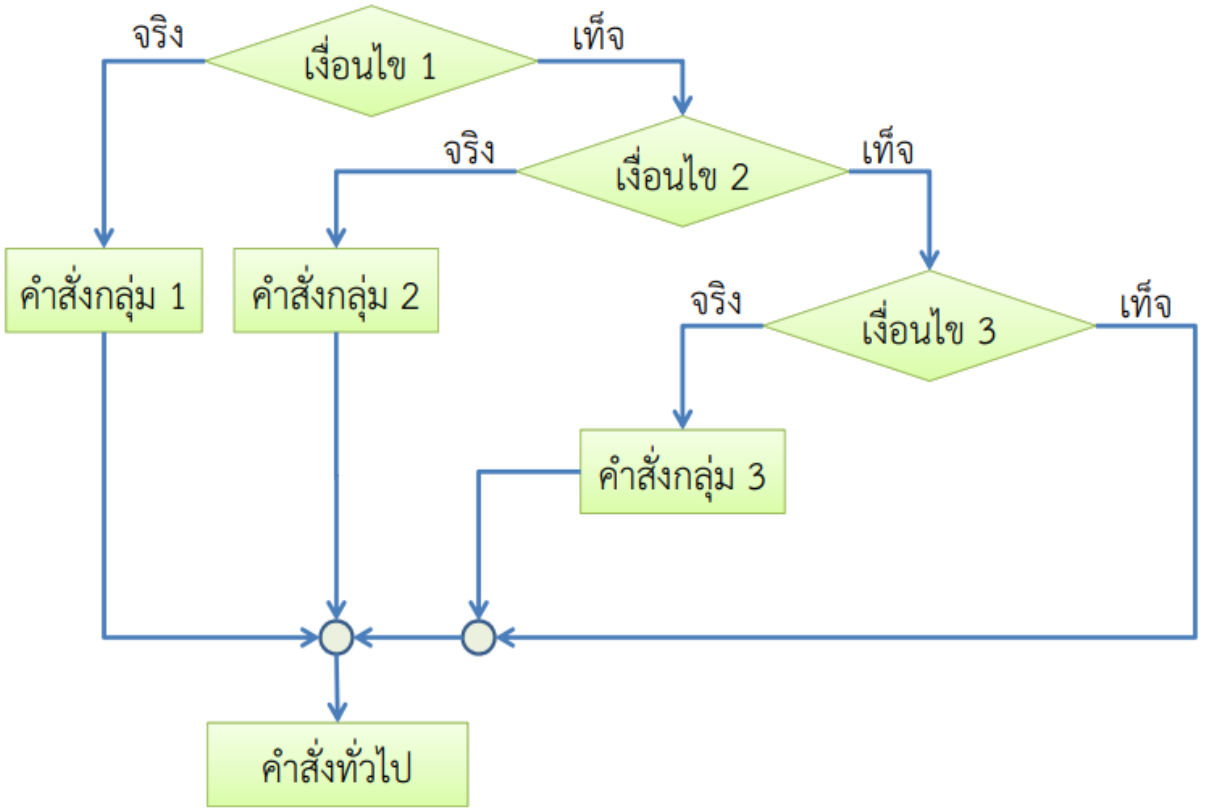

if-else




nested if

# คำสั่งเงื่อนไข nested-if

- แต่จริงๆ แล้ว การทำงานของมันก็อยู่บนพื้นฐาน if-else ทั่วๆ ไป เพียงแต่ประยุกต์ให้สามารถทำงานได้ซับซ้อนมากยิ่งขึ้น
- เมื่อสามารถจัดเรียงลำดับได้หลายรูปแบบ จึงทำให้ความเป็นไปได้ในการเขียนโปรแกรมเปิดกว้างมากขึ้น
- และความที่มันเปิดกว้าง จึงทำให้วิธีการดูหลากหลาย และทำให้ดูเหมือน if-else จะยากขึ้นกว่าเดิมหลายเท่า
- แก้ไขได้โดยการทำโจทย์และจับทางที่ถนัดเพียงทางเดียว (อย่าเน้นแค่การอ่านชีต)



# คำสั่งเงื่อนไข nested-if


- การซ้อนเงื่อนไขมีได้หลายวิธี สุดแต่คนที่เขียนจะสรรค์สร้าง
- แต่เพื่อการอธิบายจะสามารถแบ่งออกเป็นกลุ่มใหญ่ๆ ที่มักทำให้ผู้กำลังฝึกฝนสับสนได้สองวิธี
  - **การใช้ if-else if ต่อเนื่องกันไป**
  - การใช้ if ซ้อนไว้ใน if หลายๆ ชั้น

- *พอมีจุดนี้ขึ้นมา แม้ว่ามันจะทำงานบนพื้นฐาน if-else ธรรมดา แต่การที่มันสามารถประยุกต์ได้หลายแบบทำให้มันซับซ้อนกว่าเดิมมาก จนต้องเพิ่มบทเรียนสำหรับการณ์นี้โดยเฉพาะ ใครที่เข้าใจ if-else เป็นอย่างดีแล้วจะสามารถเข้าใจเรื่องนี้ได้ง่ายขึ้น*

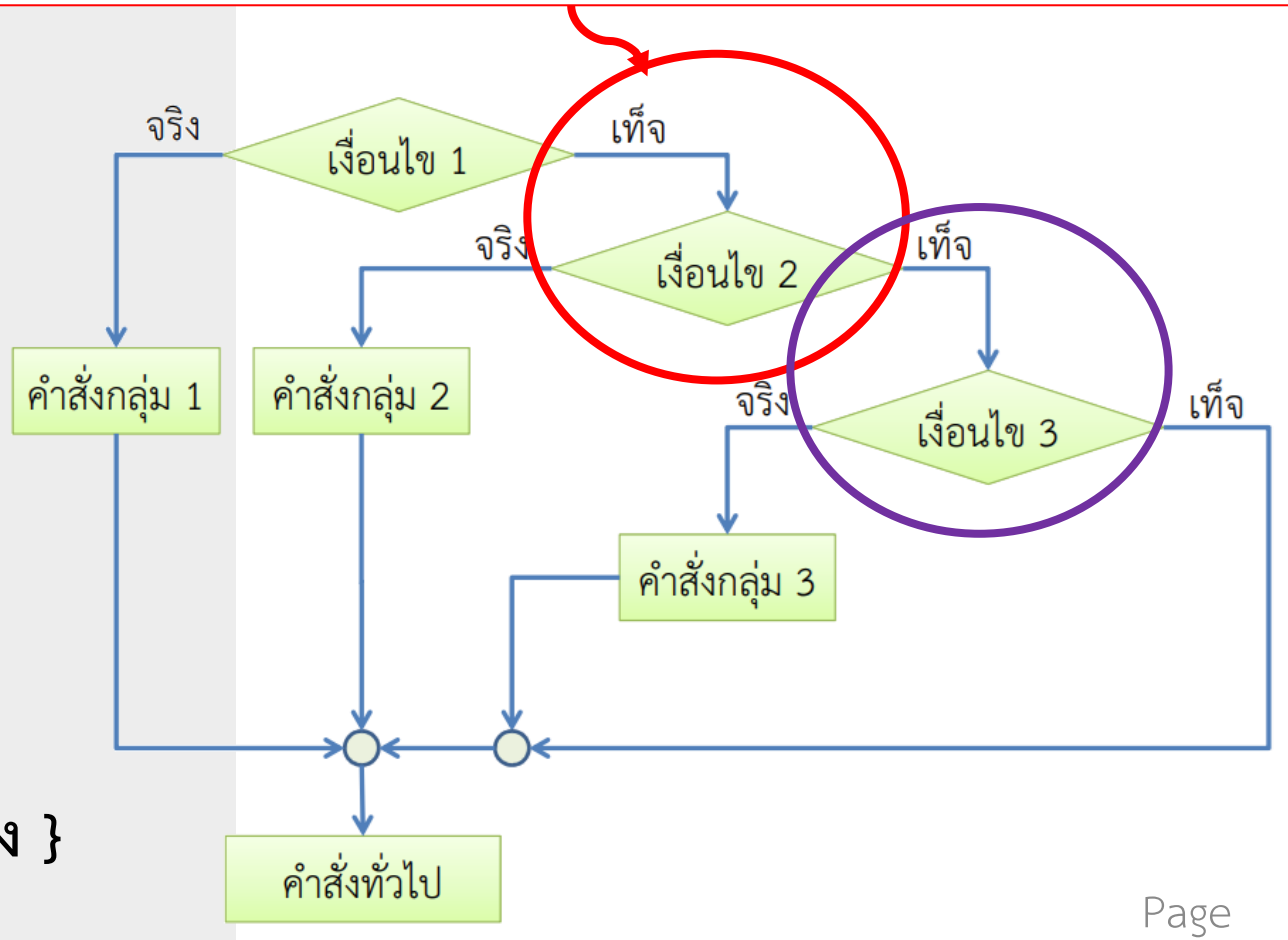
# คำสั่งเงื่อนไข nested-if


```
if (เงื่อนไขที่ 1) {
        คำสั่งเมื่อเงื่อนไขที่ 1 จริง
} else if (เงื่อนไขที่ 2) {
        คำสั่งเมื่อเงื่อนไขที่ 2 จริง
} else if (เงื่อนไขที่ 3) {
        คำสั่งเมื่อเงื่อนไขที่ 3 จริง                    // จะซ้อนกันกี่เงื่อนไขก็ได้
} ...
else { คำสั่งเมื่อไม่มีเงื่อนไขใดๆเป็นจริง }
```

# คำสั่งเงื่อนไข nested-if

```
if (เงื่อนไขที่ 1) {
        คำสั่งเมื่อเงื่อนไขที่ 1 จริง
} else if (เงื่อนไขที่ 2) {
        คำสั่งเมื่อเงื่อนไขที่ 2 จริง
} else if (เงื่อนไขที่ 3) {
        คำสั่งเมื่อเงื่อนไขที่ 3 จริง
} ...
else { คำสั่งเมื่อไม่มีเงื่อนไขใดๆเป็นจริง }
```

สังเกตว่าเงื่อนไข else if ที่อยู่ระดับต่ำกว่า จะถูกพิจารณาเมื่อลำดับที่สูงกว่าเป็นเท็จเท่านั้น



# คำสั่งเงื่อนไข nested-if


```
if (เงื่อนไขที่ 1) {
        คำสั่งเมื่อเงื่อนไขที่ 1 จริง
} else if (เงื่อนไขที่ 2) {
        คำสั่งเมื่อเงื่อนไขที่ 2 จริง
} else if (เงื่อนไขที่ 3) {
        คำสั่งเมื่อเงื่อนไขที่ 3 จริง
} ...
else { คำสั่งเมื่อไม่มีเงื่อนไขใดๆเป็นจริง }
```

คอมไพเลอร์จะ**ยอมตรวจสอบเงื่อนไขที่ 2 เมื่อเงื่อนไขที่ 1 เป็นเท็จไปแล้วเท่านั้น**
ทำนองเดียวกันกับเงื่อนไขที่อยู่ในระดับต่ำกว่า
ถ้าเงื่อนไขแรกเป็นจริงไปแล้ว คอมไพเลอร์จะไม่พิจารณาต่อ แม้เงื่อนไขที่ต่ำกว่าจะเป็นจริง

**ถ้าไม่มีเงื่อนไขใดเป็นจริงเลย จะทำคำสั่งใน else**
*จะมี else หรือไม่มีก็ได้
**ถ้าไม่มีและไม่มีเงื่อนไขอื่นใดเป็นจริง ก็จะไม่ทำคำสั่งใดเลย

# คำสั่งเงื่อนไข nested-if


**ตัวอย่างโจทย์ 8** จงเขียนโปรแกรมที่รับค่าตัวเลขจากผู้ใช้เข้ามา โดยที่

(1) หากตัวเลขเป็นคู่และเป็นลบให้พิมพ์คำว่า both even and negative

(2) หากเป็นคู่แต่ไม่เป็นลบให้พิมพ์ว่า even และ

(3) หากเป็นลบแต่ไม่เป็นคู่ให้พิมพ์ว่า negative

# คำสั่งเงื่อนไข nested-if


```c
#include <stdio.h>

void main() {
    int x;
    scanf("%d", &x);
    if(x % 2 == 0 && x < 0) {
        printf("both even and negative");
    } else if (x % 2 == 0 && x >= 0) {
        printf("even");
    } else if (x < 0 && x % 2 != 0) {
        printf("negative");
    }
}
```

# คำสั่งเงื่อนไข nested-if


ถ้าเงื่อนไขนี้เป็นจริงไปแล้ว โปรแกรมจะพิมพ์ “both even and negative” และไม่พิจารณาเงื่อนไขอื่นต่อ แม้ว่าจะเป็นจริงก็ตาม

```c
#include <stdio.h>

void main() {
    int x;
    scanf("%d", &x);
    if(x % 2 == 0 && x < 0) {
        printf("both even and negative");
    } else if (x % 2 == 0 && x >= 0) {
        printf("even");
    } else if (x < 0 && x % 2 != 0) {
        printf("negative");
    }
}
```

# คำสั่งเงื่อนไข nested-if

```c
#include <stdio.h>

void main() {
    int x;
    scanf("%d", &x);
    if(x % 2 == 0 && x < 0) {
        printf("both even and negative");
    } else if (x % 2 == 0 && x >= 0) {
        printf("even");
    } else if (x < 0 && x % 2 != 0) {
        printf("negative");
    }
}
```

เราสามารถย่อเงื่อนไขบางอย่างได้ เช่นบรรทัดนี้

การที่เงื่อนไขนี้จะถูกพิจารณาได้ แสดงว่าเงื่อนไขแรกเป็นเท็จ หมายความว่า x จะต้องมีสมบัติอย่างน้อยอย่างใดอย่างหนึ่งต่อไปนี้

1. หารสองไม่ลงตัว
2. ไม่น้อยกว่า 0

เราจึงสามารถย่อเงื่อนไขตรงนี้ได้ เพราะจะไม่ส่งผลต่อค่าความจริงแล้ว

# คำสั่งเงื่อนไข nested-if

```c
#include <stdio.h>

void main() {
    int x;
    scanf("%d", &x);
    if(x % 2 == 0 && x < 0) {
        printf("both even and negative");
    } else if (x % 2 == 0 && x >= 0) {
        printf("even");
    } else if (x < 0 && x % 2 != 0) {
        printf("negative");
    }
}
```

```c
#include <stdio.h>

void main() {
    int x;
    scanf("%d", &x);
    if(x % 2 == 0 && x < 0) {
        printf("both even and negative");
    } else if(x % 2 == 0) {
        printf("even");
    } else if(x < 0) {
        printf("negative");
    }
}
```

เทียบเท่ากัน แต่การย่อโค้ดไม่ใช่สิ่งที่จำเป็นต้องทำเสมอไป ให้ทำเมื่อมั่นใจเท่านั้น

# คำสั่งเงื่อนไข nested-if


**ความผิดพลาดที่มักพบบ่อย : การเรียงลำดับความสำคัญของเงื่อนไขส่งผลต่อการพิจารณาด้วย**

```c
#include <stdio.h>

void main() {
    int x;
    scanf("%d", &x);
    if(x % 2 == 0) {
        printf("even");
    } else if(x % 2 == 0 && x < 0) {
        printf("both even and negative");
    } else if(x < 0) {
        printf("negatve");
    }
}
```

# คำสั่งเงื่อนไข nested-if

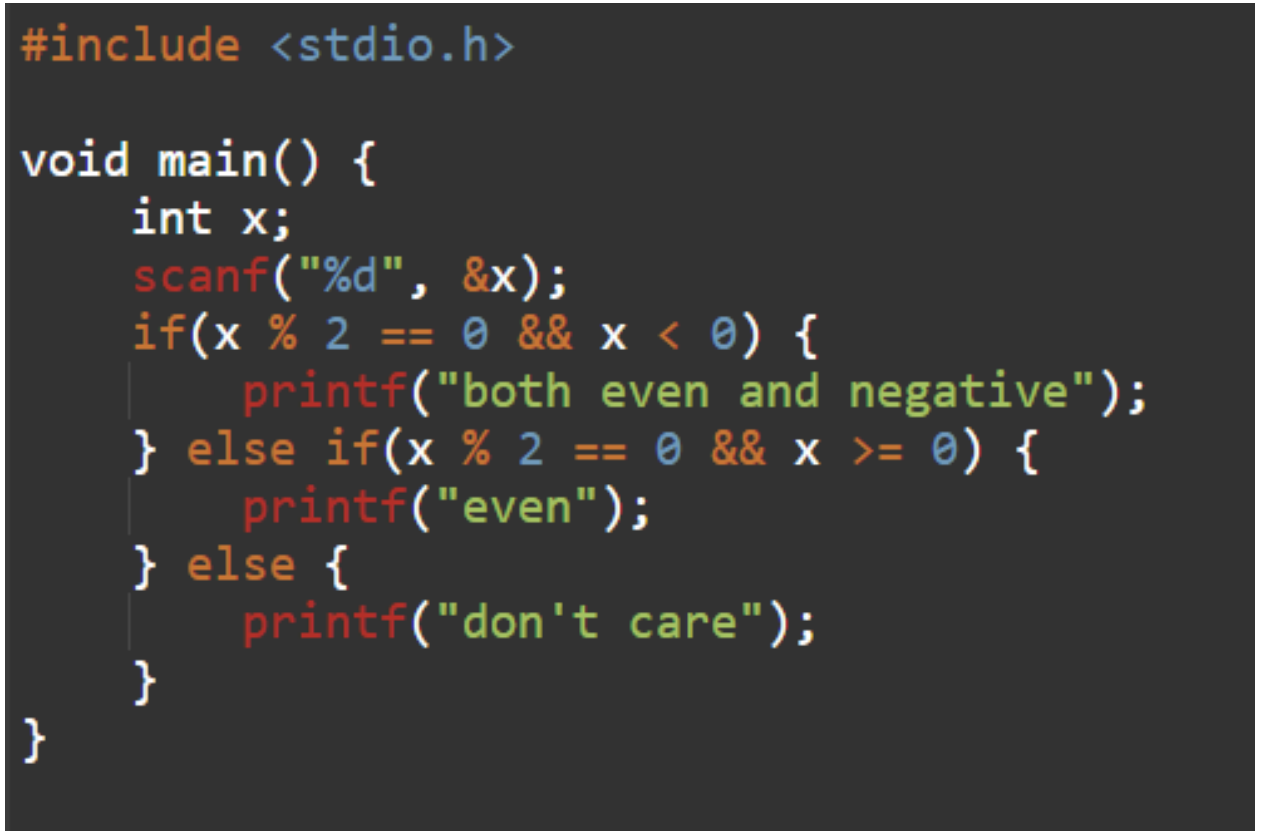

**ตัวอย่างโจทย์ 9** จงเขียนโปรแกรมที่รับค่าตัวเลขจากผู้ใช้เข้ามา โดยที่

(1) หากตัวเลขเป็นคู่และเป็นลบให้พิมพ์คำว่า both even and negative

(2) หากเป็นคู่แต่ไม่เป็นลบให้พิมพ์ว่า even และ

(3) หากไม่เข้าเงื่อนไขใด ๆ ก่อนหน้าเลยให้พิมพ์ว่า don’t care

# คำสั่งเงื่อนไข nested-if



```c
#include <stdio.h>

void main() {
    int x;
    scanf("%d", &x);
    if(x % 2 == 0 && x < 0) {
        printf("both even and negative");
    } else if(x % 2 == 0 && x >= 0) {
        printf("even");
    } else {
        printf("don't care");
    }
}
```

# คำสั่งเงื่อนไข nested-if

**ตัวอย่างโจทย์ 10** การตัดเกรดในบางมหาวิทยาลัยจะแบ่งออกเป็นสามระดับคือ ตก, ผ่าน, และยอดเยี่ยม โดยมีเกณฑ์การตัดเกรดดังนี้ น้อยกว่า 40 คะแนนคือตก (F) ได้ถึง 40 คะแนนแต่น้อยกว่า 80 คะแนนคือผ่าน (P) และได้ 80 คะแนนขึ้นไปคือยอดเยี่ยม (A) จงเขียนโปรแกรมภาษาซีที่รับคะแนนนักศึกษามาเป็นเลขทศนิยมและตัดเกรดคะแนนนั้น

| ข้อมูลเข้า (คะแนน) | ผลลัพธ์ (เกรด) |
|---|---|
| -80 | F |
| 25 | F |
| 40 | P |
| 87 | A |

# คำสั่งเงื่อนไข nested-if

```c
#include <stdio.h>
void main() {
    float point;
    scanf("%f", &point);
    if(point < 40) {
        printf("F");
    } else if(point >= 40 && point < 80) {
        printf("P");
    } else if(point >= 80) {
        printf("A");
    }
}
```

โค้ดที่ใช้ในการตัดเกรด

# คำสั่งเงื่อนไข nested-if

```c
#include <stdio.h>
void main() {
    float point;
    scanf("%f", &point);
    if(point < 40) {
        printf("F");
    } else if(point >= 40 && point < 80) {
        printf("P");
    } else if(point >= 80) {
        printf("A");
    }
}
```

```c
#include <stdio.h>
void main() {
    float point;
    scanf("%f", &point);
    if(point < 40) {
        printf("F");
    } else if(point < 80) {
        printf("P");
    } else if(point >= 80) {
        printf("A");
    }
}
```

โค้ดทั้งสองให้ผลเทียบเท่ากันหรือไม่ (1)

# คำสั่งเงื่อนไข nested-if

```c
#include <stdio.h>
void main() {
    float point;
    scanf("%f", &point);
    if(point < 40) {
        printf("F");
    } else if(point >= 40 && point < 80) {
        printf("P");
    } else if(point >= 80) {
        printf("A");
    }
}
```

```c
#include <stdio.h>
void main() {
    float point;
    scanf("%f", &point);
    if(point >= 80) {
        printf("A");
    } else if(point >= 40) {
        printf("P");
    } else if(point < 40) {
        printf("F");
    }
}
```

โค้ดทั้งสองให้ผลเทียบเท่ากันหรือไม่ (3)

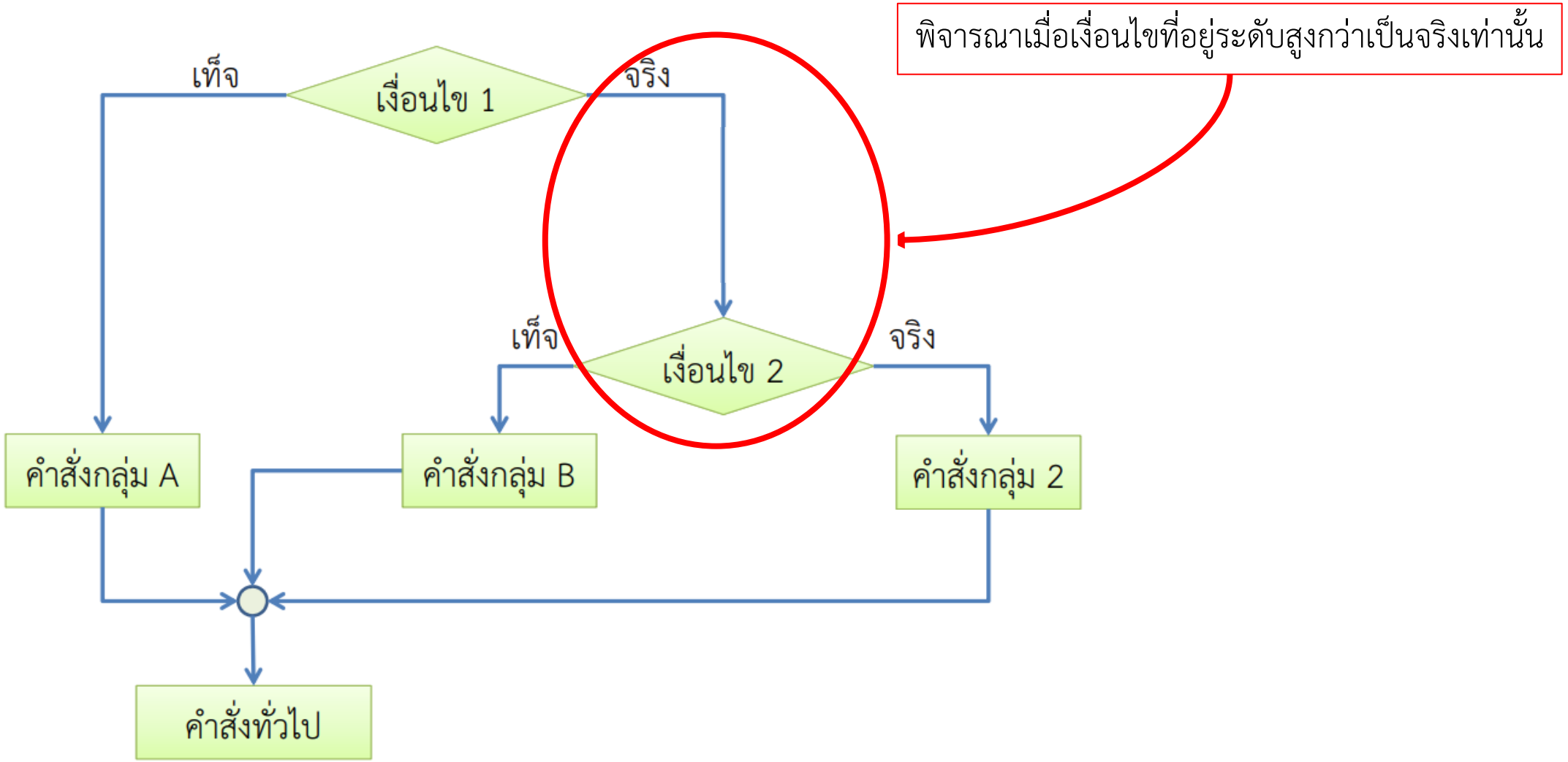
# คำสั่งเงื่อนไข nested-if


- การซ้อนเงื่อนไขมีได้หลายวิธี สุดแต่คนที่เขียนจะสรรค์สร้าง
- แต่เพื่อการอธิบายจะสามารถแบ่งออกเป็นกลุ่มใหญ่ๆ ที่มักทำให้ผู้กำลังฝึกฝนสับสนได้สองวิธี
  - การใช้ if-else if ต่อเนื่องกันไป
  - **การใช้ if ซ้อนไว้ใน if หลายๆ ชั้น**

# คำสั่งเงื่อนไข nested-if


- if – else if แบบที่ผ่านมาจะมีการพิจารณาเงื่อนไขต่อ ๆ กันไป
- เงื่อนไขที่ตามมาจะถูกพิจารณาเมื่อเงื่อนไขก่อนหน้าเป็นเท็จเท่านั้น
- ถ้าเงื่อนไขก่อนหน้าเป็นจริงไปแล้ว เงื่อนไขที่ปรากฏหลังจากนั้นจะไม่ถูกพิจารณา แม้ว่าจะเป็นจริงก็ตาม
- ในทางกลับกัน ถ้าเราต้องการให้เงื่อนไขที่ตามมาทำงานเมื่อเงื่อนไขก่อนหน้าเป็นจริง เราจะใช้การใช้ if ซ้อนไว้ใน if หลายๆ ชั้น

# คำสั่งเงื่อนไข nested-if



# คำสั่งเงื่อนไข nested-if


```
if (เงื่อนไขที่ 1) {
        if (เงื่อนไขที่ 2) {
                if (เงื่อนไขที่ 3) {
                        ... //กี่ชั้นก็ได้
                }
        }
}
```

# คำสั่งเงื่อนไข nested-if

```
if (เงื่อนไขที่ 1) {
        คำสั่ง
        if (เงื่อนไขที่ 2) {
                if (เงื่อนไขที่ 3) {
                        ... //กี่ชั้นก็ได้
                }
        }
        else {
                คำสั่ง
        }
}
```

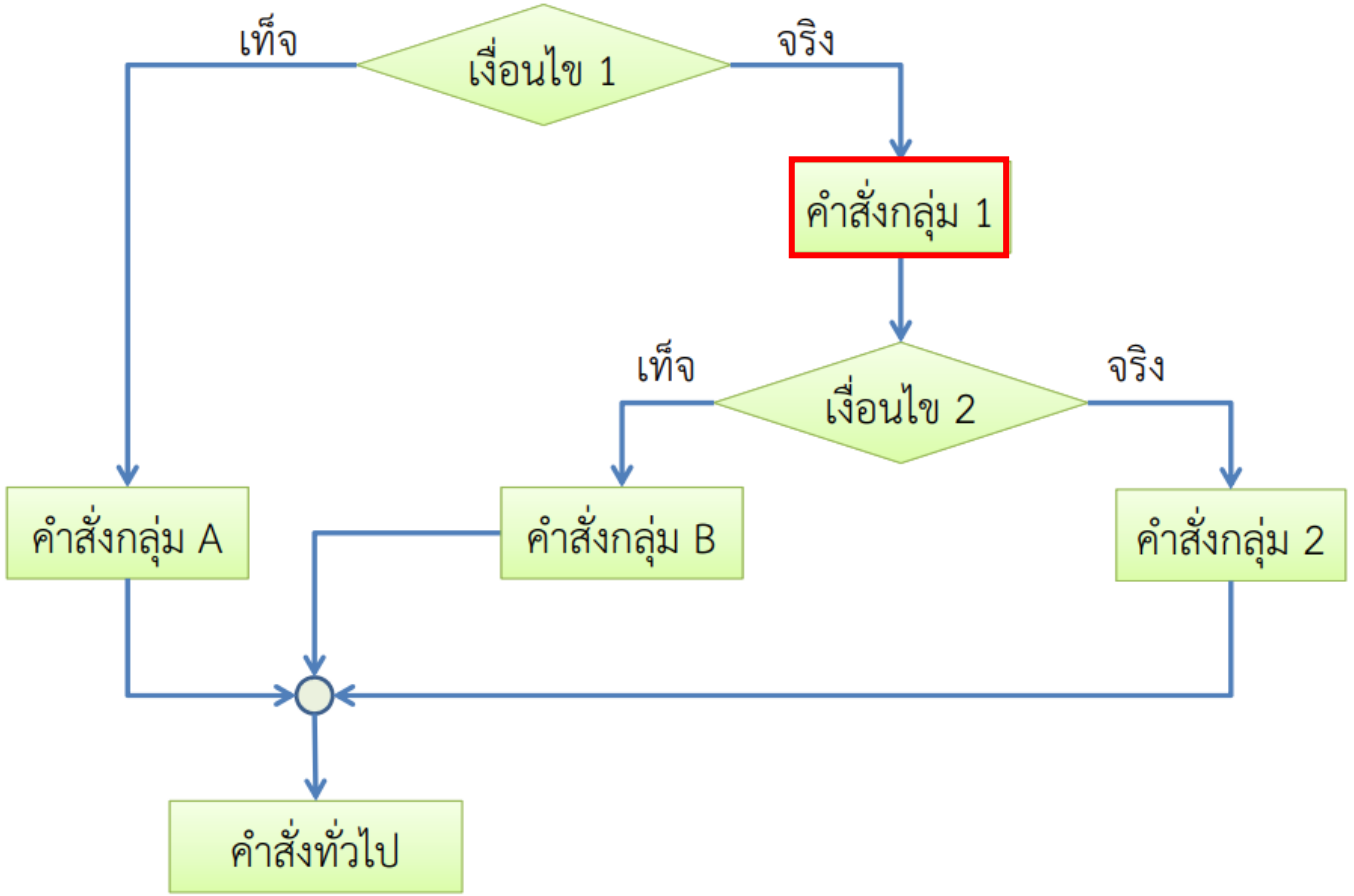
**ด้วยความที่จริงๆแล้วมันก็คือ if-else ธรรมดาที่ถูกนำมาซ้อนกัน เราจึงแทรกคำสั่งระหว่างกลางได้**

กล่าวคือ if ข้างในไม่ต้องเป็นจริง คำสั่งนี้ก็จะยังถูกทำเมื่อเงื่อนไขที่ 1 เป็นจริง

**จะมี else หรือ if-else ก็ได้ (จริงๆแล้ว if แต่ละตัวเป็นคำสั่งที่อิสระจากกัน)**

# คำสั่งเงื่อนไข nested-if: ตัวอย่าง

```
if (เงื่อนไขที่ 1) {
   คำสั่งกลุ่ม 1
   if (เงื่อนไขที่ 2) {
      คำสั่งกลุ่ม 2
   } else {
      คำสั่งกลุ่ม B
   }
} else {
   คำสั่งกลุ่ม A
}
```



# คำสั่งเงื่อนไข nested-if


**ตัวอย่างโจทย์ 10 (เดิม)** การตัดเกรดในบางมหาวิทยาลัยจะแบ่งออกเป็นสามระดับคือ ตก, ผ่าน, และ ยอดเยี่ยม โดยมีเกณฑ์การตัดเกรดดังนี้ น้อยกว่า 40 คะแนนคือตก (F) ได้ถึง 40 คะแนนแต่น้อยกว่า 80 คะแนนคือผ่าน (P) และได้ 80 คะแนนขึ้นไปคือยอดเยี่ยม (A) จงเขียนโปรแกรมภาษาซีที่รับคะแนนนักศึกษามาเป็นเลขทศนิยมและตัดเกรดคะแนนนั้น

| ข้อมูลเข้า (คะแนน) | ผลลัพธ์ (เกรด) |
|---|---|
| -80 | F |
| 25 | F |
| 40 | P |
| 87 | A |

# คำสั่งเงื่อนไข nested-if

```c
#include <stdio.h>
void main() {
    float point;
    scanf("%f", &point);
    if(point < 80) {
        if(point >= 40) {
            printf("P");
        } else {
            printf("F");
        }
    } else {
        printf("A");
    }
}
```

**ตรรกศาสตร์ของ if ซ้อนใน if**

- เงื่อนไขจะเหมือนกับการเชื่อมด้วย AND
- printf(“P”); นี้จะถูกกระทำก็ต่อเมื่อ point < 80 && point >= 40 เป็นจริง
- else จะมีเงื่อนไข (ที่มองไม่เห็น) ว่า point < 80 && !(point >= 40)

# คำสั่งเงื่อนไข nested-if


**ตัวอย่างโจทย์ 11** จงเขียนโปรแกรมที่รับค่าจำนวนเต็มจากผู้ใช้ ถ้าเลขนั้นหารด้วยสามลงตัว โปรแกรมจะพิมพ์เลขตัวนั้นออกมาและรับเลขจำนวนถัดมา ถ้าเลขตัวที่สองนั้นหารด้วยสามไม่ลงตัวก็จะพิมพ์ตัวเลขตัวที่สองออกมา แต่ถ้าตัวเลขที่สองหารด้วยสามลงตัว โปรแกรมจะพิมพ์เลขตัวแรกออกมา ในกรณีที่เลขตัวแรกหารด้วยสามไม่ลงตัว โปรแกรมจะพิมพ์เลข -1 และจบการทำงานทันที

(ถ้าไม่เขียน Flowchart สับสนแย่แน่)

# คำสั่งเงื่อนไข nested-if


```c
#include <stdio.h>
void main() {
    int x, y;
    scanf("%d", &x);
    if(x % 3 == 0) {
        printf("%d\n",x);
        scanf("%d", &y);
        if(y % 3 != 0) {
            printf("%d", y);
        } else {
            printf("%d", x);
        }
    } else {
        printf("-1");
    }
}
```

# Outline


- คำสั่งควบคุม
  - คำสั่ง If
  - คำสั่ง if-else
  - คำสั่ง nested if (ซ้อน if)
- **เรื่องลึกๆ ของ if-else กับการพิจารณาเงื่อนไข**

# เรื่องลึก ๆ ของ if-else กับการพิจารณาเงื่อนไข


ในกรณีที่เงื่อนไขประกอบด้วยเงื่อนไขย่อยที่เชื่อมด้วย && (และ) หรือ || (หรือ) โปรแกรมภาษาซีจะทำการตรวจเงื่อนไขจากซ้ายไปขวาแค่เพียงพอที่จะสรุปค่าความจริงของเงื่อนไขรวมได้ เช่น

- ถ้ามี p && q โปรแกรมจะตรวจ p ก่อน ซึ่งหาก p เป็นเท็จ เรารู้ได้แน่เลย ว่า p && q ต้องเป็นเท็จแน่ ๆ ดังนั้น โปรแกรมจะ**ไม่พิจารณา q** แต่จะสรุปค่าความจริงและไม่ทำคำสั่งที่อยู่ใน if
- ถ้า p เป็นจริง การจะสรุปค่าความจริงจะต้องตรวจ q ด้วย ดังนั้นโปรแกรมก็จะต้องทำการตรวจค่าความจริงของ q ด้วย
- ถ้าเงื่อนไขคือ p || q โปรแกรมจะตรวจ p ก่อน ถ้าหาก p เป็นจริงแล้ว โปรแกรม จะสรุปได้เลยว่า p || q เป็นจริงแน่นอน และจะ**ไม่ตรวจค่า q** แต่จะทำ คำสั่งใน if เลย

# เรื่องลึก ๆ ของ if-else กับการพิจารณาเงื่อนไข


- การคอมไพเลอร์ไม่พิจารณาข้อมูลที่เหลือ เพราะไม่ส่งผลต่อคำตอบของเงื่อนไขแบบนี้เรียกว่า Short Circuit
- Short Circuit ทำให้การประมวลผลของคอมพิวเตอร์ลดลง
- แต่บางครั้งก็จะให้ผลลัพธ์ที่ไม่พึงประสงค์ออกมาด้วย

# เรื่องลึก ๆ ของ if-else กับการพิจารณาเงื่อนไข

```c
#include <stdio.h>
void main() {
    if(5 / 0 == 7 && 3 / 2 == 0) {
        printf("Check Point 1\n");
    } else {
        printf("Check Point 2\n");
    }
}
```

```
main.c: In function 'main':
main.c:11:10: warning: division by zero [-Wdiv-by-zero]
   11 |     if(5 / 0 == 7 && 3 / 2 == 0) {
      |          ^
```

แครช เพราะมีการหารด้วย 0 ก่อน

```c
#include <stdio.h>
void main() {
    if(3 / 2 == 0 && 5 / 0 == 7) {
        printf("Check Point 1\n");
    } else {
        printf("Check Point 2\n");
    }
}
```

```
Check Point 2


...Program finished with exit code 14
Press ENTER to exit console.
```

สามารถทำงานได้ เพราะพบว่า 3/2 ทำให้ทั้งประโยคเป็นเท็จอย่างแน่นอน จึงไม่มีการคิด 5/0 ที่ควรจะแครช

# เรื่องลึก ๆ ของ if-else กับการพิจารณาเงื่อนไข

```c
#include <stdio.h>
void main() {
    if(3 / 2 != 0 || 5 / 0 == 7) {
        printf("Check Point 1\n");
    } else {
        printf("Check Point 2\n");
    }
}
```

```
Check Point 1


...Program finished with exit code 14
Press ENTER to exit console.
```

สามารถทำงานได้ เพราะพบว่า 3/2 ทำให้ทั้งประโยคเป็นจริงอย่างแน่นอน จึงไม่มีการคิด 5/0 ที่ควรจะแครช

```c
#include <stdio.h>
void main() {
    if(5 / 0 != 0 || 3 / 2 == 7) {
        printf("Check Point 1\n");
    } else {
        printf("Check Point 2\n");
    }
}
```

```
main.c: In function ‘main’:
main.c:11:10: warning: division by zero [-Wdiv-by-zero]
   11 |     if(5 / 0 != 0 || 3 / 2 == 7) {
      |          ^
```

แครช เพราะมีการหารด้วย 0 ก่อน

# สรุป


- คำสั่งควบคุม
  - คำสั่ง If --> *เลือกทำ ไม่ทำคำสั่ง ขึ้นอยู่กับเงื่อนไข*
  - คำสั่ง if-else --> *เลือกทำคำสั่งที่ 1 หรือ 2 (เลือกทางใดทางหนึ่ง) ขึ้นกับเงื่อนไข*
  - คำสั่ง nested if (ซ้อน if) --> *เป็นโครงสร้างที่ซับซ้อนขึ้น เอางานอื่นมาแทรกได้ และจะยอมพิจารณาเงื่อนไขข้างในเมื่อได้พิจารณาเงื่อนไขนอกว่าเป็นจริงแล้วเท่านั้น*
- เรื่องลึกๆ ของ if-else กับการพิจารณาเงื่อนไข --> *เมื่อประโยคมี and หรือ or ถ้าพบนิพจน์ใดทำให้ทั้งประโยคเป็นจริงหรือเท็จอย่างแน่นอน จะไม่มีการพิจารณานิพจน์อื่นต่อ ทำให้แม้คำสั่งนั้นจะทำให้เกิดแครช ก็จะไม่แครช*

# Note