



การเขียนโปรแกรมคอมพิวเตอร์ 1

Computer Programming I

คำสั่งควบคุม

คำสั่งเงื่อนไข if, if-else และ nested if

ภิญโญ แท้ประสาทสิทธิ์

Emails : pinyotae+111 at gmail dot com, pinyo at su.ac.th

Web : <http://www.cs.su.ac.th/~pinyotae/compro1/>

Facebook Group : [ComputerProgramming@CPSU](https://www.facebook.com/ComputerProgramming@CPSU)

ภาควิชาคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยศิลปากร

สัปดาห์ที่ 5

หัวข้อเนื้อหา



- คำสั่งควบคุม
- คำสั่งเงื่อนไข if
- คำสั่งเงื่อนไข if-else
- คำสั่งเงื่อนไข if ซ้อน if (nested if)

6 กรกฎาคม 2555

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

2

คำสั่งควบคุม



คือ คำสั่งที่ใช้ควบคุมทิศทางการทำงานของโปรแกรมให้เป็นไปตามที่ต้องการ
มี 2 ประเภท :

1. คำสั่งเงื่อนไข (Condition Statement)
 - if, if-else
 - switch-case
2. คำสั่งทำซ้ำ (Iteration Statement)
 - while
 - do-while
 - for

ด้วยการปรับแต่งโค้ดไม่มากนัก เราสามารถใช้ if, if-else, และ switch-case
ทดแทนกันได้ อย่างไรก็ตามของแต่ละอย่างก็มีที่ใช้งานที่เหมาะสมที่สุดแตกต่างกันไป

6 กรกฎาคม 2555

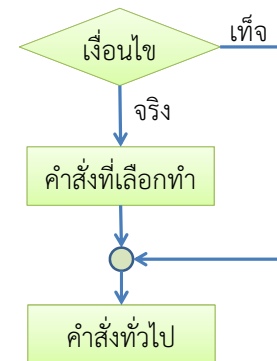
ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

3

คำสั่งเงื่อนไข if



- เป็นคำสั่งที่เลือกทำอย่างใดอย่างหนึ่ง
- โพลวชาร์ตและโค้ดภาษาซีต้นแบบ



```

if ( เงื่อนไข ) {
    คำสั่งที่เลือกทำ
}
คำสั่งทั่วไป
  
```

เวลาที่เป็นโค้ดภาษาซีจะไม่มีคำว่า
'จริง' หรือ 'เท็จ' ปรากฏอยู่
เราต้องรู้ว่า 'คำสั่งที่เลือกทำ' คือ
คำสั่งที่ตาม if มาทันที

6 กรกฎาคม 2555

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

4

ตัวอย่าง

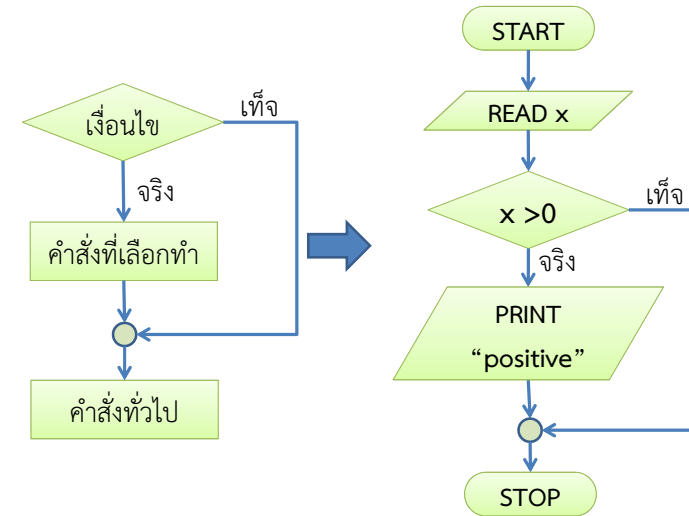


ตัวอย่างโจทย์ จงเขียนโปรแกรมภาษาซี ที่พิมพ์คำว่า positive เมื่อผู้ใช้ใส่ค่าตัวเลขจำนวนเต็มที่เป็นบวก (ไม่ต้องพิมพ์อะไรถ้าไม่เป็นบวก)

วิเคราะห์

1. ข้อมูลเข้าเป็นตัวเลข x
2. ผลลัพธ์เป็นข้อความหรืออาจจะไม่มีผลลัพธ์อะไรก็ได้
3. คำสั่งที่มีการเลือกทำคือการพิมพ์ข้อความ และเงื่อนไขที่ใช้คือ $x > 0$

โฟลวชาร์ต



ซูโดโค้ด และ โค้ดภาษาซี



ซูโดโค้ด

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

ภาษาซี

```
#include <stdio.h>

void main() {
  int x;
  scanf("%d", &x);
  if (x > 0) {
    printf("positive");
  }
}
```

ก้าวแรกกับการใช้ if



```
#include <stdio.h>

void main() {
  int x;
  scanf("%d", &x);
  if (x > 0)
    printf("positive");
}
```

- if ต้องตามด้วยวงเล็บ และภายในวงเล็บต้องมีเงื่อนไขระบุว่าจะทำสิ่งที่ตามมาหรือไม่
- ถ้าเงื่อนไขเป็นจริง โปรแกรมจะทำคำสั่งที่ตามมาทันที
- สิ่งที่แตกต่างกันจากซูโดโค้ดก็คือเราต้องประกาศตัวแปรและให้ความสำคัญกับชนิดข้อมูลของตัวแปรนั้น
- ถ้ามีการอ่านข้อมูลเข้า และ/หรือ การแสดงข้อความจะต้องมีการใช้ `#include <stdio.h>`

ตัวอย่าง : เงื่อนไขที่ซับซ้อนขึ้น



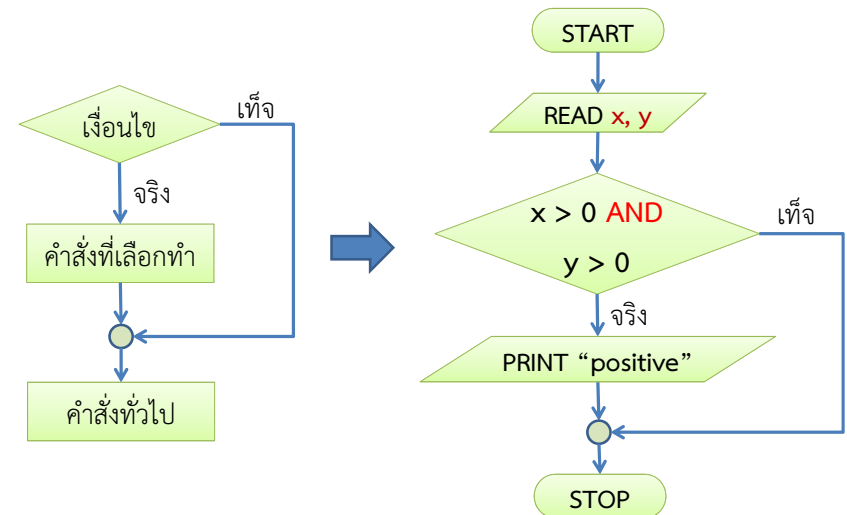
ตัวอย่างโจทย์ จงเขียนโปรแกรมภาษาซี ที่รับเลขจำนวนเต็มสองค่าจากผู้ใช้ โปรแกรมนี้จะพิมพ์คำว่า positive เมื่อตัวเลขทั้งสองจำนวนเป็นบวก และจะไม่พิมพ์อะไรเลยหากมีตัวเลขที่ไม่ได้เป็นบวกอยู่ด้วย

วิเคราะห์ ในกรณีนี้เราต้องพิจารณาเงื่อนไขจากค่าสองค่าพร้อมกัน วิธีที่นิยมใช้กันมากก็คือ 'ให้แยกออกเป็นสองเงื่อนไขย่อย แล้วนำมารวมกันด้วยวิธีทางตรรกศาสตร์'

สำหรับข้อนี้หากเราตั้งชื่อทั้งสองค่าว่า x และ y เราจะได้เงื่อนไขย่อยเป็น

(1) $x > 0$ กับ (2) $y > 0$ เนื่องจากเงื่อนไขทั้งสองต้องเป็นจริงพร้อมกัน การนำเงื่อนไขย่อยนี้มารวมกันจึงใช้ 'และ' ซึ่งก็คือเครื่องหมาย \wedge ในตรรกศาสตร์ และ เครื่องหมาย $\&\&$ ในภาษาซี

โฟลวชาร์ต



การใช้เครื่องหมาย && เพื่อรวมเงื่อนไขใน if



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

```
void main() {
```

```
    int x, y;
```

```
    scanf("%d %d", &x, &y);
```

```
    if(x > 0 && y > 0) {
```

```
        printf("positive");
```

```
    }
```

```
}
```

ใช้เครื่องหมาย && ภายในวงเล็บเงื่อนไขของ if

ตอนนี้มีสองค่าที่ต้องรับมา เราจึงใช้ตัวแปรสองตัว

แบบทดสอบความเข้าใจตัวเอง

โปรแกรมจะทำอะไร ถ้าข้อมูลเข้าคือ

- 5 3
- 5 0
- -1 2
- 0 0

ตัวอย่าง : เงื่อนไขที่ซับซ้อนขึ้น (2)



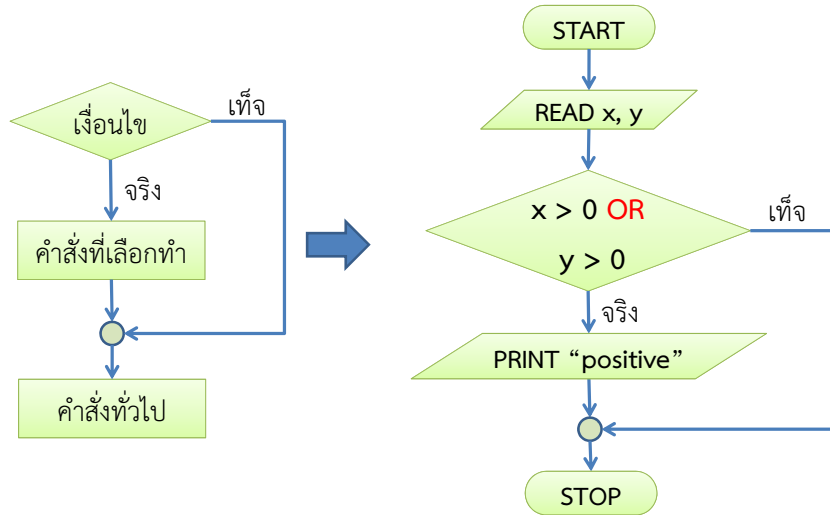
ตัวอย่างโจทย์ จงเขียนโปรแกรมภาษาซี ที่รับเลขจำนวนเต็มสองค่าจากผู้ใช้ โปรแกรมนี้จะพิมพ์คำว่า positive หนึ่งครั้ง เมื่อมีตัวเลขอย่างน้อยหนึ่งตัวเป็นบวก และจะไม่พิมพ์อะไรเลยหากไม่มีตัวเลขที่เป็นบวกอยู่ด้วย

วิเคราะห์ ตอนนี้เงื่อนไขเปลี่ยนเป็น 'มีอย่างน้อยหนึ่งตัว' แสดงว่ามีตัวหนึ่งที่เป็นบวกก็เพียงพอแล้ว และจะเป็น x หรือ y ก็ได้ หรือตัวเลขทั้งสองจะเป็นบวกด้วยกันทั้งคู่ก็ได้

ปัญหา นี้ ต้องการหาว่าเงื่อนไขอย่างน้อยหนึ่งตัว เป็นจริงหรือไม่ จะตรงกับกรรวมเงื่อนไขด้วยคำว่า 'หรือ' ซึ่งใช้เครื่องหมาย \vee ในตรรกศาสตร์ และใช้เครื่องหมาย $\|\|$ ในภาษาซี

คนจำนวนมากเริ่มงตรงนี้ นักศึกษาควรไปทบทวนเรื่องตรรกศาสตร์มาด้วย

โฟลวชาร์ต



การใช้เครื่องหมาย || เพื่อรวมเงื่อนไขใน if



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

ใช้เครื่องหมาย || ภายในวงเล็บเงื่อนไขของ if

แบบทดสอบความเข้าใจตัวเอง
โปรแกรมจะทำอะไร ถ้าข้อมูลเข้าคือ

- 5 3
- 5 0
- -1 2
- 0 0

สิ่งที่มีมือใหม่มักทำผิด



เวลาที่บอกว่า ‘ถ้าเงื่อนไขเป็นจริงอย่างน้อยหนึ่งอย่าง’

- มือใหม่จะคิดผิดแล้วแยกเงื่อนไขออกจากกัน แล้วใช้ if สองครั้ง
- ถ้าจับเงื่อนไขแยกกัน การทำงานจะผิดไปจากเดิม

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

จะเกิดอะไรขึ้น ถ้าหากว่าผู้ใช้ใส่เลข 5 3 เข้ามา ?

ย้อนดูปัญหาง่าย ๆ แต่ต่างกันเล็กน้อย



ตัวอย่างโจทย์ จงเขียนโปรแกรมภาษาซี ที่รับเลขจำนวนเต็มจากผู้ใช้งาน โปรแกรมจะพิมพ์คำว่า positive เมื่อผู้ใช้ใส่ค่าตัวเลขที่เป็นบวก และไม่ว่าผู้ใช้จะใส่เลขใดเข้ามา ก่อนจบโปรแกรมให้พิมพ์คำว่า good bye

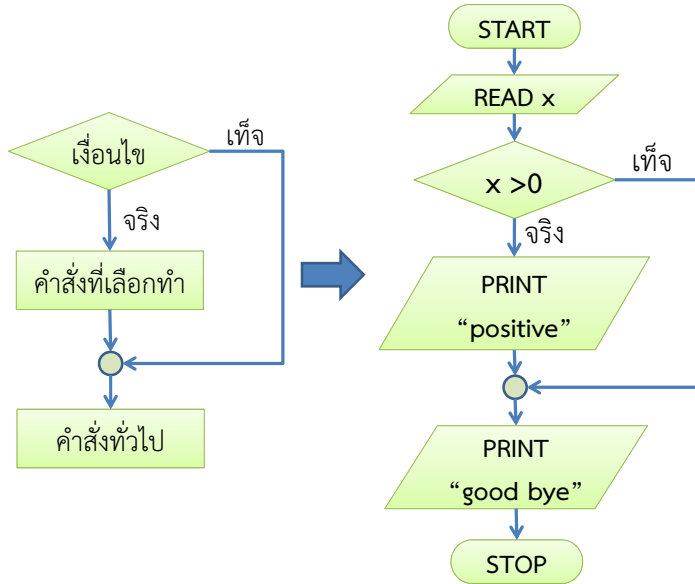
วิเคราะห์ เห็นได้ว่างานที่ต้องทำมีสองแบบ แบบแรกคือแบบเลือกทำ และแบบที่สองก็คือยังงี้ก็ต้องทำแน่ ๆ

มีเทคนิคในการคิดง่าย ๆ แต่ได้ผลก็คือ

- งานที่เลือกทำจะเกี่ยวข้องกับ if คือ ทำเมื่อเงื่อนไขที่กำหนดให้เป็นจริง
- งานที่ต้องทำแน่ ๆ จะอยู่นอก if คือ ต้องทำอย่างไม่มีเงื่อนไข

จากโจทย์ เห็นได้ว่า การพิมพ์คำว่า positive เกิดขึ้นเมื่อเงื่อนไขเป็นจริง ส่วนการพิมพ์คำว่า good bye เป็นสิ่งที่ต้องทำโดยไม่มีเงื่อนไข

โฟลวชาร์ต



โค้ดภาษาซี



```

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

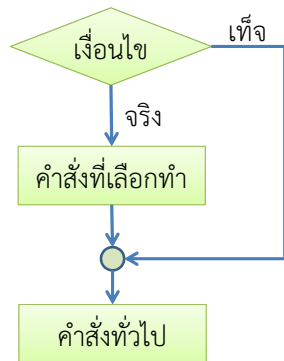
คำสั่งนี้ตามหลัง if มาทันทีจึงอยู่ใต้เงื่อนไขของ if

คำสั่งนี้อยู่ภายนอก if โปรแกรมจะทำคำสั่งนี้อย่างไม่มีเงื่อนไข ถ้าเทียบกับโฟลวชาร์ต มันก็คือคำสั่งที่มาหลังการบรรจบกันของทางแยกทั้งสองนั่นเอง

สรุปเรื่องเกี่ยวกับ if แบบพื้นฐาน



- คำสั่งที่ตาม if มาทันทีจะอยู่ภายใต้เงื่อนไขที่กำหนดไว้ในวงเล็บ ()
 - ถ้าเงื่อนไขในวงเล็บเป็นจริง โปรแกรมจะทำคำสั่งที่ตาม if มา
 - ถ้าเงื่อนไขไม่เป็นจริง คำสั่งที่ตามมานั้นจะถูกข้ามไป



สรุปเรื่องเกี่ยวกับ if แบบพื้นฐาน



- คำสั่งที่ตาม if มาทันทีจะอยู่ภายใต้เงื่อนไขที่กำหนดไว้ในวงเล็บ ()
 - ถ้าเงื่อนไขในวงเล็บเป็นจริง โปรแกรมจะทำคำสั่งที่ตาม if มา
 - ถ้าเงื่อนไขไม่เป็นจริง คำสั่งที่ตามมานั้นจะถูกข้ามไป
- ถ้ามีคำสั่งที่สองตามมา คำสั่งนั้นถือว่าเป็นคำสั่งทั่วไป
 - โปรแกรมจะทำคำสั่งที่สองนี้อย่างไม่มีเงื่อนไข
 - ในภาษาซี ย่อหน้าไม่ได้บอกว่ามันอยู่ภายใต้ if หรือเปล่า
 - สิ่งที่มาตามทันทีถือว่าอยู่ภายใต้ if

```

if(x > 0)
    printf("positive\n");
printf("good bye");
  
```

ถ้าเราอยากให้ if ทำคำสั่งมากกว่าหนึ่งคำสั่งละ



ตัวอย่างที่ผ่านมาก่อนหน้าทั้งหมด เราบอกว่า ‘ถ้าเงื่อนไขเป็นจริง ให้พิมพ์คำว่า positive’ แล้วก็เลิกการันไป

- แต่ถ้าเราบอกว่า “หากเลขทั้งสองที่ผู้ใช้ใส่เข้ามาเป็นบวกทั้งคู่ ให้โปรแกรม (1) พิมพ์คำว่า positive, (2) หาผลบวก และ (3) พิมพ์ผลบวกของเลข” ถ้าเป็นแบบนี้จะอย่างไร ?
- กรณีนี้มีสิ่งที่ต้องการให้ทำภายใต้เงื่อนไขที่เป็นจริงมากกว่าหนึ่งอย่าง
➔ ควรรวมคำสั่งพวกนั้นไว้เป็นโค้ดบล็อกเดียวกัน
- เรารวมโค้ดหลาย ๆ คำสั่งเป็นบล็อกเดียวได้ด้วยการเอาโค้ดไปใส่ไว้ในวงเล็บปีกกา
- จากนั้นเอาวงเล็บปีกกาที่บรรจุโค้ดไว้แล้ว วางต่อจาก if ทันที แบบนี้จะถือว่า ทุกอย่างในวงเล็บปีกกานั้นอยู่ภายใต้ if

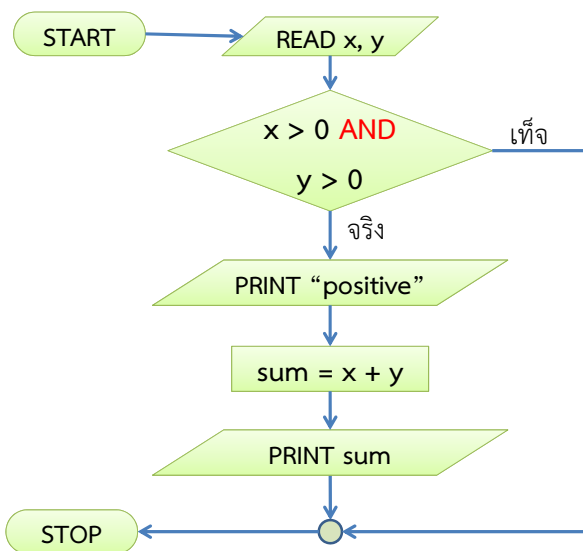
ตัวอย่างการให้ if ทำคำสั่งมากกว่าหนึ่งคำสั่ง



ตัวอย่างโจทย์ จงเขียนโปรแกรมภาษาซีที่รับค่าจำนวนเต็มจากผู้เข้ามาสองจำนวน หากจำนวนทั้งสองเป็นบวกทั้งคู่ โปรแกรมจะพิมพ์คำว่า positive จากนั้นจะทำการบวกเลขทั้งสองนั้น แล้วพิมพ์ผลบวกออกมา แต่หากตัวเลขที่ใส่เข้ามาไม่ได้อาจเป็นบวกอยู่ด้วย โปรแกรมจะจบการทำงานโดยไม่พิมพ์อะไรออกมา

วิเคราะห์ สิ่งที่ต้องทำเมื่อเงื่อนไข if เป็นจริงมีอยู่มากกว่าหนึ่งอย่าง เมื่อคำสั่งที่อยู่ภายใต้เงื่อนไข if มีมากกว่าหนึ่ง เราต้องรวมคำสั่งพวกนี้เป็นก้อนเดียวด้วยการนำไปใส่ไว้ในวงเล็บปีกกา

โฟลวชาร์ต



ตัวอย่างการใช้ { } เพื่อรวมคำสั่งให้เป็นบล็อกเดียว



```

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

บล็อกวงเล็บปีกกานี้ตามหลัง if มาทันที จึงถือว่าอยู่ภายใต้ if
ถ้าเงื่อนไขใน if เป็นจริง คำสั่งที่อยู่ในบล็อกก็จะถูกทำงานไปด้วยกันทั้งหมด

แล้วถ้ามีคำสั่งเดียวจะเอาไปใส่ใน { } ได้หรือเปล่า



ตอบ ได้ และเป็นสิ่งที่ควรทำเสมอ แม้แต่ผมก็ทำอย่างนั้นเวลาทำงานจริง ภาษาซีไม่บังคับว่าถ้ามีคำสั่งเดียวจะต้องใส่ไว้ในวงเล็บปีกกา แต่เพื่อป้องกันความผิดพลาด อันเกิดจากการใส่คำสั่งที่สองตามมาแล้วลืมวงเล็บปีกกา ผู้มีประสบการณ์จำนวนมากจึงเลือกที่จะใส่วงเล็บปีกกาไว้ตั้งแต่แรก

```
int x;
scanf("%d", &x);
if(x > 0) {
    printf("positive\n");
}
printf("good bye");
```

คำสั่งเดียวก็ใส่ไว้ในวงเล็บปีกกาได้ และเราควรทำแบบนี้ตลอด

สิ่งที่ตามหลังวงเล็บปีกกามาถือว่าอยู่นอก if ตามปรกติ

ข้อมรกับตัวอย่างที่ซับซ้อนขึ้น



ตัวอย่างโจทย์ จงเขียนโปรแกรมภาษาซีที่รับค่าจำนวนเต็มจากผู้เข้ามาค่าหนึ่ง หากตัวเลขนั้นหารด้วย 9 ไม่ลงตัว ให้พิมพ์คำว่า not divisible ไม่เช่นนั้นก็ให้จบการทำงานของโปรแกรมโดยไม่ต้องพิมพ์ข้อความใด ๆ ออกมา

วิเคราะห์ ให้ข้อมูลเข้าดังกล่าวคือ x หาก x หารด้วย 9 ลงตัว แสดงว่าเงื่อนไขคือ $x \% 9 == 0$ และ ถ้าหารไม่ลงตัวแสดงว่าเงื่อนไขคือ $x \% 9 != 0$ (อย่าใช้ว่า $x \% 9 == 1$ เพราะไม่ครอบคลุมบางกรณี)

หรือเราอาจใช้เครื่องหมายนิเสธจากตรรกศาสตร์เพื่อกลับค่าความจริงของการหารลงตัวเป็นการหารไม่ลงตัวแทน ซึ่งจะได้เงื่อนไขที่ตรงข้ามกับการหารลงตัวเป็น $!(x \% 9 == 0)$

(ขอให้นักศึกษาซ่อมทำตัวอย่างพวกนี้ โดยไม่ต้องดูการวิเคราะห์และเฉลย)

โค้ดทดสอบการหารไม่ลงตัว



แบบใช้เงื่อนไขทางตัวเลข

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

แบบใช้เทคนิคทางตรรกะเข้าช่วย

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

ข้อมรกับตัวอย่างที่ซับซ้อนขึ้น (2)



(บางทีโจทย์เปลี่ยนไปนิดเดียวก็ทำให้นักศึกษาที่หลงคิดว่าตัวเองเข้าใจสะดุ้งได้)

ตัวอย่างโจทย์ จงเขียนโปรแกรมภาษาซีที่รับค่าจำนวนเต็มจากผู้เข้ามาสองจำนวน หากตัวเลขที่รับมาตัวใดตัวหนึ่งแต่ไม่ใช่ทั้งสองเป็นบวก โปรแกรมจะพิมพ์คำว่า one-positive ไม่เช่นนั้นโปรแกรมจะจบการทำงานโดยไม่ต้องพิมพ์อะไรออกมา

วิเคราะห์ วิธีหนึ่งที่เป็นไปได้คือนับว่าตัวเลขที่ได้มา มีกี่ตัวที่เป็นบวก ถ้ามีตัวเดียวก็ให้พิมพ์ ไม่เช่นนั้นก็ไม่ต้องทำอะไร การนับแบบนี้มีข้อดีตรงที่ว่ามันเป็นสิ่งที่อิสระจากกัน คือเหมือนเอา if มาต่อกัน แทนที่จะเป็นเอา if มาซ้อนกัน (เพราะการนับว่าตัวเลขแรกเป็นบวกสิ้นสุดลงก่อนที่จะนับตัวที่สอง การใช้ if ทดสอบการเป็นบวกของตัวเลขจึงแยกออกจากกัน)

โค้ดภาษาซีสำหรับโจทย์ ‘เลขบวกค่าเดียว’



```
void main() {
    int x, y;
    scanf("%d %d", &x, &y);
    int count = 0;
    if(x > 0) {
        count = count + 1;
    }
    if(y > 0) {
        count = count + 1;
    }
    ....
}
```

```
if (count == 1) {
    printf("one-positive");
}
}
```

6 กรกฎาคม 2555

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

29

ข้อมรกับตัวอย่างที่ซับซ้อนขึ้น (3)



(โจทย์เดิม แต่เปลี่ยนวิธีคิด)

ตัวอย่างโจทย์ จงเขียนโปรแกรมภาษาซีที่รับค่าจำนวนเต็มจากผู้ใช้งานสองจำนวน หากตัวเลขที่รับมาตัวใดตัวหนึ่งแต่ไม่ใช่ทั้งสองเป็นบวก โปรแกรมจะพิมพ์คำว่า one-positive ไม่เช่นนั้นโปรแกรมจะจบการทำงานโดยไม่พิมพ์อะไรออกมา

วิเคราะห์ เราสามารถใช้ความสามารถทางตรรกศาสตร์แก้ปัญหานี้ได้ (ถึงแม้มันจะยากกว่าวิธีที่ผ่านมาก็ตาม แต่การศึกษาวิธีนี้ถือว่าการเรียนรู้ที่ดี)

กำหนดให้ p แทนข้อความว่า x เป็นบวก และ q แทนข้อความ y เป็นบวก

จำนวนเต็มจะเป็นบวกแค่ตัวเดียวเมื่อ p หรือ q เป็นจริง แต่ไม่ใช่ทั้งสอง

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

6 กรกฎาคม 2555

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

30

โค้ดสำหรับโจทย์ ‘เลขบวกค่าเดียว’ แบบตรรกศาสตร์



```
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");
    }
}
```

คำนวณค่าความจริงเก็บไว้ก่อน แบบนิพจน์ตรรกศาสตร์ได้

อยู่ในรูปเดียวกับที่แสดงไว้ก่อนหน้า

6 กรกฎาคม 2555

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

31

โค้ดสำหรับโจทย์ ‘เลขบวกค่าเดียว’ แบบตรรกศาสตร์ (2)



เนื่องจากโปรแกรมเมอร์มักไม่คิดจะคำนวณค่าความจริงเก็บไว้ก่อน โค้ด

โดยมากจะทำการเปรียบเทียบลงไปเงื่อนไขของ if เลย

```
void main() {
    int x, y;
    scanf("%d %d", &x, &y);
    if( ( (x > 0) && !(y > 0) ) ||
        (!(x > 0) && (y > 0) ) )
    {
        printf("one-positive");
    }
}
```

แต่ถึงโปรแกรมเมอร์จะนิยมทำแบบนี้มากกว่า ก็ใช่ว่ามันจะดีสักเท่าไร

6 กรกฎาคม 2555

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

32

สรุปเรื่องเกี่ยวกับ if แบบยากขึ้นมาน้อย



- คำสั่งที่ตาม if มาทันทีจะอยู่ภายใต้เงื่อนไขที่กำหนดไว้ในวงเล็บ ()
 - ถ้าเงื่อนไขในวงเล็บเป็นจริง โปรแกรมจะทำคำสั่งที่ตาม if มา
 - ถ้าเงื่อนไขไม่เป็นจริง คำสั่งที่ตามมานั้นจะถูกข้ามไป
- สิ่งที่มาตามทันทีนั้น จะเป็นคำสั่งโดดหรือเป็นบล็อกวงเล็บปีกกาก็ได้
- เงื่อนไขของ if อาจจะเป็นเงื่อนไขย่อยสองอันที่เชื่อมกันด้วยตัวดำเนินการทางตรรกะอย่าง && หรือ || ก็ได้
- เงื่อนไขของ if จะมีการใช้เครื่องหมายนิเสธก็ได้ จะผสมการทำงานทางตรรกะเข้าไปหลาย ๆ อย่างก็ได้
- ถ้าเงื่อนไขเป็นอิสระจากกัน การแยกออกมาคิดจะทำให้อ่านเข้าใจง่ายกว่า แต่ธรรมชาติของการคิดแต่ละคนไม่เหมือนกัน → เลือกวิธีที่เรามั่นใจ

6 กรกฎาคม 2555

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

33

หัวข้อเนื้อหา



- คำสั่งควบคุม
- คำสั่งเงื่อนไข if
- คำสั่งเงื่อนไข if-else
- คำสั่งเงื่อนไข if ซ้อน if (nested if)

6 กรกฎาคม 2555

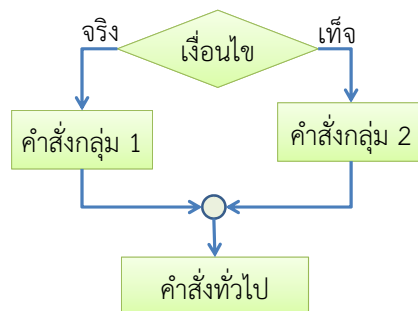
ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

34

คำสั่งเงื่อนไข if-else



- คำสั่ง if แบบโดดเป็นการเลือกทำหรือไม่ทำ
- ส่วน if-else เป็นการเลือกทำอย่างใดอย่างหนึ่งจากทางเลือกสองทาง
- คำสั่งที่จะเลือกทำจึงแบ่งออกเป็นสองกลุ่ม โปรแกรมเลือกทำกลุ่มแรกเมื่อเงื่อนไขของ if เป็นจริง และทำกลุ่มที่สองเมื่อเงื่อนไขของ if เป็นเท็จ



6 กรกฎาคม 2555

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

35

ตัวอย่างโจทย์ if-else



ตัวอย่างโจทย์ จงเขียนโปรแกรมที่รับค่าตัวเลขจากผู้ใช้เข้ามาหนึ่งตัว หากตัวเลขเป็นบวกให้พิมพ์คำว่า pos ไม่เช่นนั้นให้พิมพ์ว่า neg

วิเคราะห์

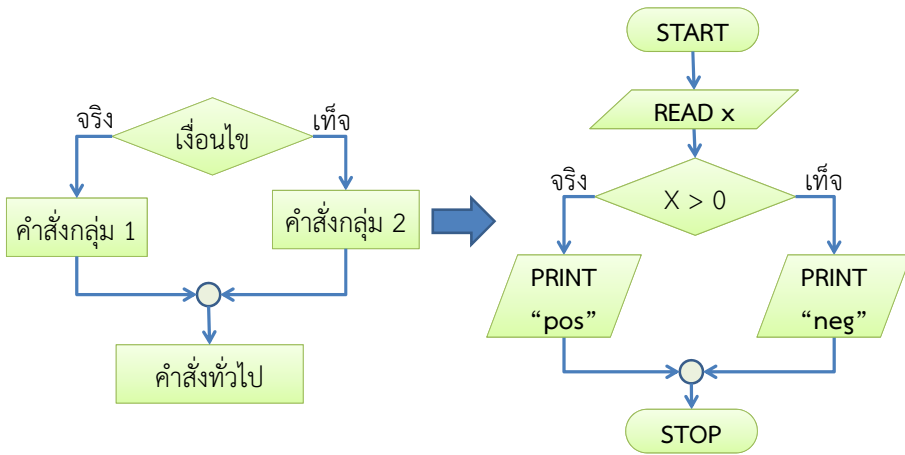
- ข้อมูลเข้าเป็นตัวเลข ผลลัพธ์เป็นข้อความบนหน้าจอ
- เงื่อนไขมีอย่างเดียว เป็นการเลือกทำอย่างใดอย่างหนึ่ง
- การเลือกทำอย่างใดอย่างหนึ่งแบบนี้สามารถใช้ if-else มาแก้ปัญหาได้

6 กรกฎาคม 2555

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

36

โฟลวชาร์ต if-else โจทย์ pos-neg



โค้ด if-else สำหรับโจทย์ pos-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");
    }
}
```

หลักการทั่วไปของ if-else



- สิ่งก็ตาม if มาทันที จะถูกทำงานถ้าเงื่อนไขใน if เป็นจริง
- สิ่งก็ตาม else มาทันที จะถูกทำงานถ้าเงื่อนไขใน if เป็นเท็จ
- เช่นเดียวกับ if สิ่งก็ตามมาทันทีต้องเป็นคำสั่งโดด หรือไม่ก็วงเล็บปีกกา
- เมื่อจบสิ่งที่ตาม if มาทันที ก็ถือว่าของอื่น ๆ อยู่นอก if
ถึงจุดนี้ถ้าจะมี else ก็ให้ใส่ตามไปเลย อย่าเอาคำสั่งอะไรไปคั่นเป็นอันขาด

```
if (x > 0)
    printf("pos");
else ✓
    printf("neg");
```

```
if (x > 0)
    printf("pos");
else ✗
    printf("neg");
```

สิ่งที่คนมักจะทำผิด



- เมื่อต้องการให้ if ทำงานหลายอย่าง กลับไม่ยอมเอา 'งานหลายอย่าง' นั้นใส่ไว้ในวงเล็บปีกกาเพื่อรวมให้เป็นบล็อกเดียวกัน ทำให้คำสั่งที่สอง สาม สี่ ที่ตามมาถูกนับว่าอยู่นอก if
- ปัญหาเดียวกันก็เกิดกับ else คือไม่ยอมเอาคำสั่งที่จะให้ทำด้วยกัน ไปอยู่ในวงเล็บปีกกา

เพื่อป้องกันความผิดพลาดในระยะยาว แนะนำว่าต่อให้มีคำสั่งเดียวก็ให้ใส่วงเล็บปีกกาทุกครั้ง ถึงโค้ดจะยาวขึ้น แต่ก็ช่วยลดความผิดพลาดได้

- ไม่ยอมคิดให้ดีกว่าสิ่งไหนที่จะทำเมื่อเงื่อนไขเป็นจริง (IF) สิ่งไหนที่จะทำเมื่อเงื่อนไขเป็นเท็จ (ELSE) และสิ่งไหนที่จะทำโดยไม่มีเงื่อนไข (นอก IF-ELSE) ทั้งที่จริงแล้ว หากเราคิดของพวกนี้ออก เราจะเอาโค้ดไปใส่ได้ถูกที่ทุกครั้ง

ตัวอย่างโจทย์ if-else



โจทย์ จงเขียนโปรแกรมรับเลขทศนิยมแบบ single precision สองจำนวน หากตัวเลขที่สองเป็นบวกให้หาผลบวกของเลขทั้งสอง แล้วพิมพ์ผลลัพธ์ออกมา หากตัวเลขที่สองเป็นศูนย์หรือเป็นลบให้หาผลคูณของเลขทั้งสองแล้วพิมพ์ผลลัพธ์ออกมา

วิเคราะห์ เนื่องจากตัวเลขถ้าไม่เป็นบวก ก็ต้องเป็นลบหรือศูนย์ เราจึงสามารถใช้ if สำหรับจำนวนบวก และตัวเลขอื่น ๆ ใช้ else ได้เลย

ข้อควรระวัง เราต้องแยกให้ออกว่า else ที่เราคิดใช้นั้นมันเป็นตัวแทนของอีกกรณีหนึ่งได้จริงหรือไม่ ถ้าไม่ใช่หรือไม่แน่ใจให้ใช้ nested if ที่จะสอนต่อไป

โค้ด if-else โจทย์บวกหรือคูณ



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

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

โค้ด if-else โจทย์บวกหรือคูณ



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

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

หัวข้อเนื้อหา



- คำสั่งควบคุม
- คำสั่งเงื่อนไข if
- คำสั่งเงื่อนไข if-else
- คำสั่งเงื่อนไข if ซ้อน if (nested if)
 - แบบต่อเนื่องกันไป if – else if – else ...
 - แบบซ้อนไว้ด้านใน

```
if ( ) {
    if ( ) {
    }
}
```

การซ้อนเงื่อนไขมีสองแบบ



1. การใช้ if – else if ต่อเนื่องกันไป
2. การใช้ if ภายใต้อีกตัวหนึ่ง

เรื่องการซ้อนเงื่อนไขเป็นแนวคิดที่มีใหม่มักหลงคิดว่าตัวเองเข้าใจ แต่พอให้ทำแบบฝึกหัดหรือข้อสอบจะทำได้ยาก เพราะไม่สามารถจัดลำดับความคิดที่จัดการกับเงื่อนไขได้อย่างถูกต้อง ทำให้มีที่ผิดมากมายตามมา

เรื่องนี้เป็นหนึ่งในเหตุผลสำคัญที่คนไม่ผ่านวิชานี้ (แต่เหตุผลที่สำคัญที่สุดก็คือ นักศึกษาขี้เกียจ เข้าใจผิดว่าตัวเองรู้เรื่อง คิดว่าตัวเองเข้าใจ ผันว่าจะทำข้อสอบหรือแก้ปัญหาได้ แต่ความผิมนั้นไม่ได้เป็นความจริงเลย)

การใช้ if – else if ต่อเนื่องกันไป



ลักษณะทั่วไป

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

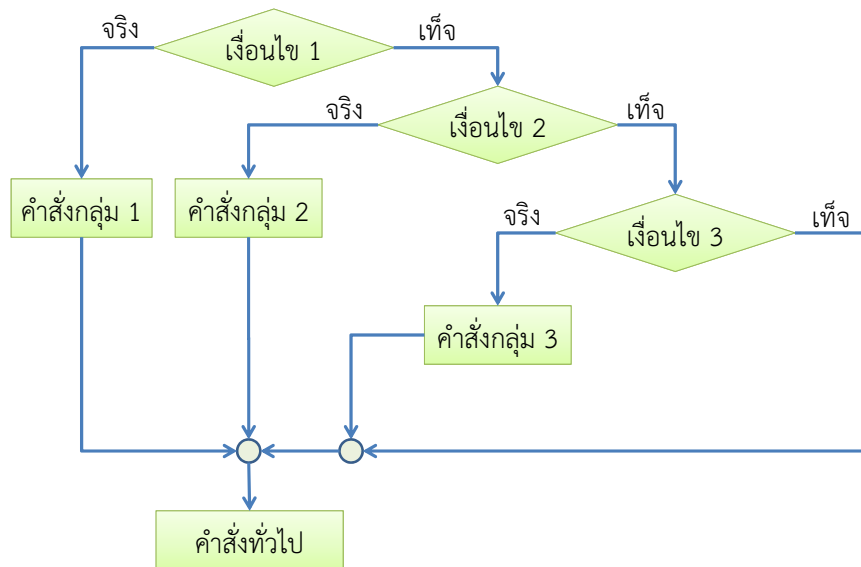
เงื่อนไขที่ซ้อนต่อ ๆ กันมาเช่นนี้จะมีที่เงื่อนไขก็ได้ จะซ้อนกันแค่ครั้งเดียวก็ได้ จะซ้อนกันสองครั้ง สามครั้ง หรือมากกว่าสามครั้งก็ได้

โปรแกรมจะพิจารณาทีละเงื่อนไขตามลำดับการปรากฏ

วิธีนี้โปรแกรมจะเลือกทำงานที่อยู่ภายใต้เงื่อนไขที่เป็นจริงอันแรก

และไม่สนใจงานที่เหลือที่ตามมาแม้ว่าเงื่อนไขที่ตามมาจะเป็นจริงก็ตาม

โฟลวชาร์ต if – else if ...



ตัวอย่าง การใช้ if - else if



ตัวอย่างโจทย์ จงเขียนโปรแกรมที่รับค่าตัวเลขจากผู้ใช้เข้ามา โดยที่

- (1) หากตัวเลขเป็นคู่และเป็นลบให้พิมพ์คำว่า both even and negative
- (2) หากเป็นคู่แต่ไม่เป็นลบให้พิมพ์ว่า even และ
- (3) หากเป็นลบแต่ไม่เป็นคู่ให้พิมพ์ว่า negative

วิเคราะห์ แม้เงื่อนไขจะดูคล้ายเดิม แต่เราสังเกตได้ว่าการจะพิมพ์คำว่า both even and negative ออกมาได้ โปรแกรมต้องพิจารณาทั้งความเป็นคู่และเป็นลบไปด้วยกัน จะแยกออกมาจากกันไม่ได้ และนี่เป็นเหตุผลที่ทำให้เงื่อนไขทั้งสองไม่เป็นอิสระจากกัน

โค้ดตัวอย่าง if-else if สำหรับโจทย์ คู่และค่าลบ



```
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");
    }
}
```

เช่นเดียวกับการใช้ if ทั่ว ๆ ไป จะมี else เปล่า ๆ เป็นอันสุดท้ายหรือไม่ก็ได้ ขึ้นอยู่กับว่าเงื่อนไขสอดคล้องกับจุดประสงค์หรือเปล่านั้น

การทำงานของโปรแกรมเมื่อพบเงื่อนไขที่เป็นจริง



โปรแกรมจะเลือกทำชุดคำสั่งที่อยู่ภายใต้เงื่อนไขที่เป็นจริงอันแรก และไม่สนใจงานภายใต้ if อื่น ๆ ที่ตามมาแม้ว่าเงื่อนไขนั้นจะเป็นจริงก็ตาม

```
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");
}
```

ตัวอย่าง ถ้าเราใส่เลข -4 เข้าไป เงื่อนไขแรกจะเป็นจริง เพราะเป็นคู่และเป็นลบ โปรแกรมจะทำคำสั่งภายใต้เงื่อนไขแรกทันที และไม่สนใจเงื่อนไขอื่น ๆ เลย ไม่ว่ามันจะเป็นจริงหรือไม่ก็ตาม

ความเทียบเท่ากันของโปรแกรม



การทำงานของ if - else if ทำให้เราสามารถย่อเงื่อนไขได้ หากเรามั่นใจว่า ผลการทำงานสอดคล้องกับวัตถุประสงค์ของโปรแกรม

```
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");
}
```

จะเกิดอะไรขึ้นถ้าหากเราเอาเงื่อนไข $x >= 0$ ออกไป

ลองใส่เลข -8, 8, 0, 5, -5 เข้าไปแล้วได้ผลลัพธ์ในใจดู

โปรแกรมที่เทียบเท่ากัน



```
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");
}
```

```
if(x % 2 == 0 && x < 0) {
    printf("both even and negative");
} else if(x % 2 == 0) {
    printf("even");
} else if(x < 0) {
    printf("negative");
}
```

แบบที่สองเป็นโค้ดที่กะทัดรัดขึ้น แต่ถ้าไม่เข้าใจจริงอย่าใช้

อันตรายจากการใช้โค้ดกระทัดรัดอย่างคนรู้ไม่เท่าทัน



```
if(x % 2 == 0 && x < 0) {  
    printf("both even and negative");  
} else if(x % 2 == 0) {  
    printf("even");  
} else if(x < 0) {  
    printf("negative");  
}
```

```
if(x % 2 == 0) {  
    printf("even");  
} else if(x % 2 == 0 && x < 0) {  
    printf("both even and negative");  
} else if(x < 0) {  
    printf("negative");  
}
```

ลองใส่เลข -8, 8, 0, 5, -5 เข้าไปแล้วเทียบผลลัพธ์จากโปรแกรมทั้งสอง

else ที่ห้อยท้าย if - else if ไว้เป็นอย่างไร



- การใช้ else เปล่า ๆ โดยไม่มีเงื่อนไขกำกับ หมายความว่า ถ้าเงื่อนไขที่มาก่อนหน้าใน if หรือ else if ไม่มีตัวไหนที่เป็นจริงเลย → ให้ทำชุดคำสั่งที่ตาม else มาทันที
- เปรียบเทียบ if - else กับ if - else if - ... - else
 - คำสั่งภายใต้ else ใน if - else นั้นจะทำงานเมื่อเงื่อนไขใน if ไม่เป็นจริง
 - คำสั่งภายใต้ else ใน if - else if - ... - else จะทำงานถ้าหากไม่มีเงื่อนไขใด ๆ เลยทั้งใน if และ else if ที่เป็นจริง

ตัวอย่าง การใช้ if - else if



ตัวอย่างโจทย์ จงเขียนโปรแกรมที่รับค่าตัวเลขจากผู้ใช้เข้ามา โดยที่

- หากตัวเลขเป็นคู่และเป็นลบให้พิมพ์คำว่า both even and negative
- หากเป็นคู่แต่ไม่เป็นลบให้พิมพ์ว่า even และ
- หากไม่เข้าเงื่อนไขใด ๆ ก่อนหน้าเลยให้พิมพ์ว่า don't care

วิเคราะห์ ตรงกรณีสุดท้ายมันชี้ให้เห็นว่า ให้พิมพ์ข้อความหากเงื่อนไขก่อนหน้าเป็นเท็จทั้งหมด นั่นก็คือเราสามารถใส่ else ปิดท้ายไว้ได้เลย

โค้ด even, negative or don't care



```
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");  
    }  
}
```

ตัวอย่างปัญหา : ตัดเกรด



โจทย์ การตัดเกรดในบางมหาวิทยาลัยจะแบ่งออกเป็นสามระดับคือ ตก, ผ่าน, และ ยอดเยี่ยม โดยมีเกณฑ์การตัดเกรดดังนี้ น้อยกว่า 40 คะแนนคือตก (F) ได้ถึง 40 คะแนนแต่น้อยกว่า 80 คะแนนคือผ่าน (P) และได้ 80 คะแนนขึ้นไปคือยอดเยี่ยม (A) จงเขียนโปรแกรมภาษาซีที่รับคะแนนนักศึกษาเข้ามาเป็นเลขทศนิยมและตัดเกรด คะแนนนั้น

ตัวอย่าง

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

โค้ดตัดเกรด



```
#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");
    }
}
```

โค้ดตัดเกรดอันนี้เทียบเท่ากับอันที่แล้วหรือไม่



```
#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");
    }
}
```

แล้วอันนี้ล่ะ



```
#include <stdio.h>

void main() {
    float point;
    scanf("%f", &point);

    if(point >= 80) {
        printf("A");
    } else if(point >= 40 && point < 80) {
        printf("P");
    } else if(point < 40) {
        printf("F");
    }
}
```

ถ้าเป็นอันนี้ว่าไง



```
#include <stdio.h>

void main() {
    float point;
    scanf("%f", &point);

    if(point >= 80) {
        printf("A");
    } else if(point < 80) {
        printf("P");
    } else if(point < 40) {
        printf("F");
    }
}
```

6 กรกฎาคม 2555

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

61

สุดท้ายแล้ว ถ้าตอบพวกนี้ได้หมดแสดงว่าเข้าใจจริง



```
#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");
    }
}
```

6 กรกฎาคม 2555

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

62

หัวข้อเนื้อหา



- คำสั่งควบคุม
- คำสั่งเงื่อนไข if
- คำสั่งเงื่อนไข if-else
- คำสั่งเงื่อนไข if ซ้อน if (nested if)
 - แบบต่อเนื่องกันไป if - else if - else ...
 - แบบซ้อนไว้ด้านใน
if () {
 if () {

 }
}

6 กรกฎาคม 2555

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

63

การซ้อน if ไว้ภายใน if



- if - else if แบบที่ผ่านมาจะมีการพิจารณาเงื่อนไขต่อ ๆ กันไป
 - เงื่อนไขที่ตามมาจะถูกพิจารณาเมื่อเงื่อนไขก่อนหน้าเป็นเท็จ
- แล้วถ้าเราอยากให้พิจารณาเงื่อนไขที่ตามมา เมื่อเงื่อนไขก่อนหน้าเป็นจริงล่ะ
→ ต้องซ้อน if อีกตัวพร้อมเงื่อนไขไว้ข้างใน if เหมือนเป็นคำสั่งที่ให้ทำเมื่อเงื่อนไขของ if ตัวแรกเป็นจริง

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

เงื่อนไขที่ซ้อนไว้ด้านในแบบนี้จะมีกี่ชั้นก็ได้ จะซ้อนกันแค่ครั้งเดียวก็ได้ จะซ้อนกันสองครั้ง สามครั้ง หรือมากกว่าสามครั้งก็ได้

ภายใน if แต่ละอันจะทำงานอย่างอื่นด้วยก็ได้ ไม่จำเป็นต้องทำที่ if ตัวในสุดแต่เพียงอย่างเดียว

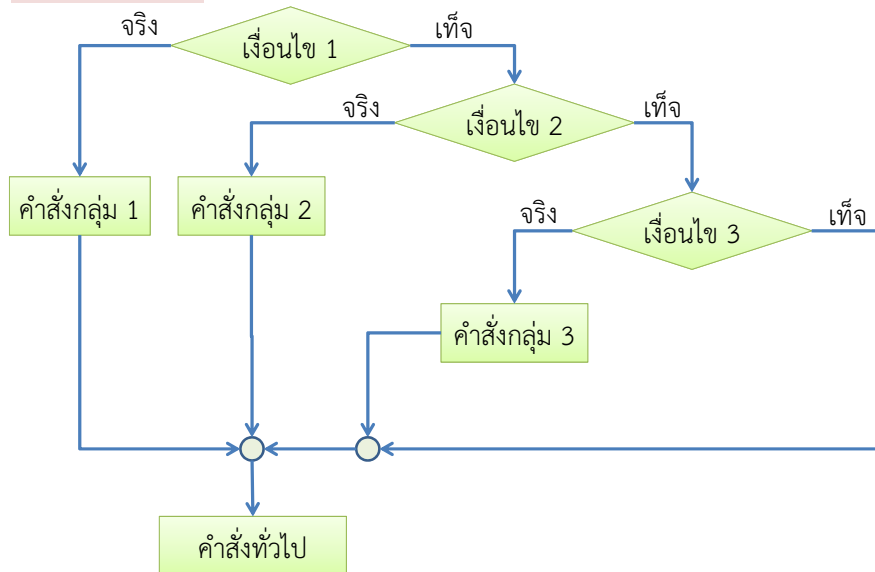
6 กรกฎาคม 2555

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

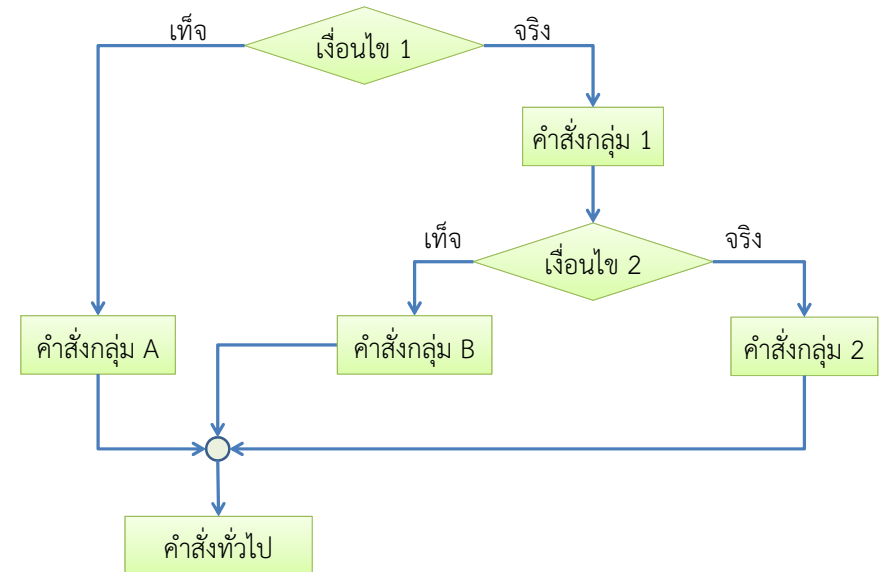
64

แบบก่อนหน้า

โฟลวชาร์ต if - else if ...



โฟลวชาร์ตของการซ้อน if ไว้ข้างใน (แบบเบา ๆ)



โค้ดที่แสดงโครงสร้างของ nested if หน้าที่แล้ว

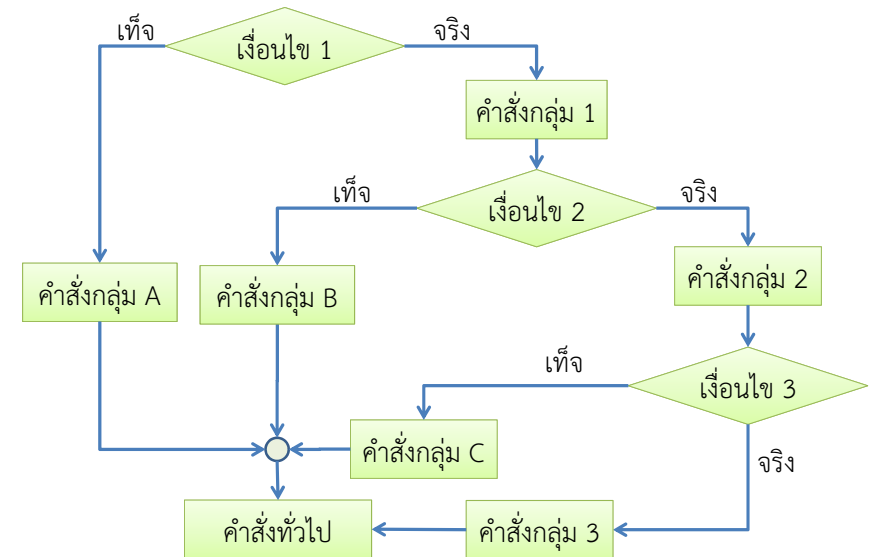


```

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

```

โฟลวชาร์ตของการซ้อน if ไว้ข้างใน (แบบจัดหนัก)





โจทย์ การตัดเกรดในบางมหาวิทยาลัยจะแบ่งออกเป็นสามระดับคือ ตก, ผ่าน, และ ยอดเยี่ยม โดยมีเกณฑ์การตัดเกรดดังนี้ น้อยกว่า 40 คะแนนคือตก (F) ได้ถึง 40 คะแนนแต่น้อยกว่า 80 คะแนนคือผ่าน (P) และได้ 80 คะแนนขึ้นไปคือยอดเยี่ยม (A) จงเขียนโปรแกรมภาษาซีที่รับคะแนนนักศึกษาเข้ามาเป็นเลขทศนิยมและตัดเกรด คะแนนนั้น

ตัวอย่าง

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

คราวนี้จะลองใช้การ
ซ้อน if ไว้ข้างใน



```
void main() {
    float point;
    scanf("%f", &point);

    if(point < 80) {
        if(point >= 40) {
            printf("P");
        } else {
            printf("F");
        }
    } else {
        printf("A");
    }
}
```

ตรรกศาสตร์ของการซ้อน if ไว้ด้านใน



- เงื่อนไขจะเหมือนกับการเชื่อมด้วย AND

```
if(point < 80) {
    if(point >= 40) {
        printf("P");
    } else {
        printf("F");
    }
} ...
```

- จากข้างบนตรง printf("P"); นี้จะถูกกระทำก็ต่อเมื่อ point < 80 && point >= 40 เป็นจริง
- ส่วนตรง else จะเป็นว่า point < 80 && !(point >= 40)

ตัวอย่างโจทย์ เงื่อนไขเลขชวงง



โจทย์ จงเขียนโปรแกรมที่รับค่าจำนวนเต็มจากผู้ใช้ ถ้าเลขนั้นหารด้วยสามลงตัว โปรแกรมจะพิมพ์เลขตัวนั้นออกมาและรับเลขจำนวนถัดมา ถ้าเลขตัวที่สองนั้นหารด้วยสามไม่ลงตัวก็จะพิมพ์ตัวเลขตัวที่สองออกมา แต่ถ้าตัวเลขที่สองหารด้วยสามลงตัว โปรแกรมจะพิมพ์เลขตัวแรกออกมา ในกรณีที่เลขตัวแรกหารด้วยสามไม่ลงตัว โปรแกรมจะพิมพ์เลข -1 และจบการทำงานทันที

วิเคราะห์ โปรแกรมต้องตรวจสอบเลขตัวที่หนึ่งก่อน และจึงจะตัดสินใจว่าจะรับเลขถัดมาอีกหรือไม่ ถ้ารับก็ต้องตรวจสอบเลขตัวที่สองต่อด้วย เราจะเห็นได้ว่าการรับและตรวจค่าตัวเลขตัวที่สอง เป็นสิ่งที่เลือกทำ และอาจจะไม่เกิดขึ้นก็ได้ ดังนั้นการรับค่าและตรวจเลขตัวที่สองจึงควรอยู่ใน if

การตรวจว่าสิ่งไหนเป็นการเลือกทำ เป็นการวิเคราะห์ที่ใช้ได้เสมอ



```
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");
}
```



ในกรณีที่เงื่อนไขประกอบด้วยเงื่อนไขย่อย เชื่อมด้วย && หรือ ||

→ โปรแกรมภาษาซีจะทำการตรวจเงื่อนไข จากซ้ายไปขวา แค่เพียงพอที่จะสรุปค่าความจริงของเงื่อนไขรวมได้ เช่น

- ถ้ามี p && q โปรแกรมจะตรวจ p ก่อน ซึ่งหาก p เป็นเท็จ เรารู้ได้แน่เลย ว่า p && q ต้องเป็นเท็จแน่ ๆ (ความรู้จากเรื่องตรรกศาสตร์ ม. 4) ดังนั้น โปรแกรมจะไม่พิจารณา q แต่จะสรุปค่าความจริงและไม่ทำคำสั่งที่อยู่ใน if
- จากตัวอย่างเดิม ถ้า p เป็นจริง การจะสรุปค่าความจริงจะต้องตรวจ q ด้วย ดังนั้นโปรแกรมก็ต้องทำการตรวจค่าความจริงของ q ด้วย
- ถ้ามี p || q โปรแกรมจะตรวจ p ก่อน ถ้าหาก p เป็นจริงแล้ว โปรแกรมจะสรุปได้เลยว่า p || q เป็นจริงแน่นอน และจะไม่ตรวจค่า q แต่จะทำคำสั่งใน if เลย

Short Circuit กับการพิจารณาค่าความจริง



- การพิจารณาค่าความจริงแบบที่ภาษาซีทำนั้น จะเห็นได้ว่าการข้ามขั้นตอนการตรวจค่าความจริงบางอย่างได้ด้วย
→ การข้ามการตรวจค่าความจริงแบบนี้เรียกว่า short circuit
- Short circuit ทำให้มีการทำงานที่ลดลง แต่มันก็มีผลข้างเคียงบางอย่างเกิดขึ้นตามมาด้วย
- การมี short circuit ไม่ได้หมายความว่าโปรแกรมจะให้ผลเหมือนเดิมทุกประการ

ตัวอย่าง short circuit (1)



ตามหลักตรรกศาสตร์การสลับ p && q ไปเป็น q && p ไม่ได้ทำให้ค่าความจริงเปลี่ยน แต่ในภาษาซีมันอาจจะมีเรื่องไม่คาดคิดเกิดขึ้นได้

```
if(3 / 2 == 0 && 5 / 0 == 7) {
    printf("Check Point 1\n");
} else {
    printf("Check Point 2\n");
}
```

โปรแกรมนี้พิมพ์คำว่า Check Point 2

```
if(5 / 0 == 7 && 3 / 2 == 0) {
    printf("Check Point 1\n");
} else {
    printf("Check Point 2\n");
}
```

ส่วนอันนี้จะแครช เพราะเกิดการหารด้วยศูนย์

ตัวอย่าง short circuit (2)



ตามหลักตรรกศาสตร์การสลับ $p \parallel q$ ไปเป็น $q \parallel p$ ไม่ได้ทำให้ค่าความจริงเปลี่ยน แต่ในภาษาซีมันอาจจะมีเรื่องไม่คาดคิดเกิดขึ้นได้

```
if(3 / 2 != 0 || 5 / 0 == 7) {  
    printf("Check Point 1\n");  
} else {  
    printf("Check Point 2\n");  
}
```

โปรแกรมนี้พิมพ์คำว่า Check Point 1

```
if(5 / 0 == 7 || 3 / 2 != 0) {  
    printf("Check Point 1\n");  
} else {  
    printf("Check Point 2\n");  
}
```

ส่วนอันนี้อาจแครช เพราะมีการหารด้วยศูนย์