# บทที่ 5 การเลือกทำ (Selection)

## วัตถุประสงค์

1) เพื่อให้นักศึกษาเข้าใจการทำงานที่มีการตัดสินใจเลือกทำ

2) เพื่อให้นักศึกษาสามารถเลือกวิธีการเลือกทำที่เหมาะสมในการแก้ปัญหาตัวอย่าง

3) เพื่อให้นักศึกษาสามารถเขียนโปรแกรมที่มีการเลือกทำงานได้

โดยปกติแล้วการทำงานของโปรแกรมภาษา C++ จะทำเรียงคำสั่งจนกว่าจะจบโปรแกรม แต่คำสั่งบางอย่างอาจจะมีการเลือกทำเกิดขึ้น การเลือกทำ (selection) คือการตัดสินใจว่าจะทำคำสั่งใด หรือไม่ทำคำสั่งใดโดยพิจารณาจากเงื่อนไข (condition) ที่กำหนดให้ โดยถ้าเงื่อนไขนั้นเป็นจริง โปรแกรมจะทำคำสั่งที่กำหนด แต่หากเป็นเท็จ โปรแกรมจะไม่ทำตามคำสั่งที่กำหนด อาจจะข้ามคำสั่งนั้นไปเลยหรืออาจจะทำคำสั่งอื่นแทน

เงื่อนไข (condition) จะต้องสามารถถูกประเมินค่าความจริงว่าเป็นจริง (true) หรือเท็จ (false) ได้ ในภาษา C++ ค่าความจริงสามารถแบ่งได้เป็น 2 ประเภทคือ 1) ข้อมูลชนิดบูลีน (Boolean) มีค่า true หรือ false ซึ่งโดยปกติแล้วจะได้จากการเปรียบเทียบค่าของข้อมูลสองตัว และ 2) ข้อมูลชนิดตัวเลข โดยค่าที่จะทำให้เงื่อนไขเป็นจริงคือค่าใดๆ ที่ไม่ใช่ 0 ส่วนค่าที่จะทำให้เงื่อนไขเป็นเท็จนั้นคือค่า 0 ค่าเดียวเท่านั้น นอกจากนี้เงื่อนไขที่จะถูกประเมินสามารถประกอบด้วยเงื่อนไขเดี่ยว (single condition) หรือหลายๆ เงื่อนไขประกอบกัน (compound condition) ก็ได้

## 5.1 การเปรียบเทียบค่าของข้อมูล

ในภาษา C++ เราสามารถใช้เครื่องหมายตามรูปที่ 1 เพื่อเปรียบเทียบค่า 2 ค่า ไม่ว่าจะเป็นตัวแปรหรือค่าตรงตัวเลยก็ได้ โดยค่าที่ได้จะถูกประเมินเป็นจริง (true) หรือเท็จ (false) เท่านั้น ข้อควรระวังในการใช้เครื่องหมายเหล่านี้ก็คือ ในการเขียนโปรแกรม เราใช้เครื่องหมาย >= สำหรับมากกว่าหรือเท่ากับและ <= สำหรับน้อยกว่าหรือเท่ากับ ไม่ใช่เครื่องหมาย ≥ และ ≤ ตามลำดับ นอกจากนี้เราต้องระวังเรื่องเครื่องหมายเท่ากับ (=) ซึ่งในการเปรียบเทียบว่าค่าสองค่าเท่ากันหรือไม่ เราใช้เครื่องหมายเท่ากับ 2 ตัว (==) เสมอ ถ้าเราใช้เครื่องหมายเท่ากับอันเดียว (=) จะหมายความถึงการให้ค่า เช่น x == 5 คือการเปรียบเทียบว่าค่าของ x เท่ากับ 5 หรือไม่ แต่ถ้าเราเขียนว่า x = 5 เป็นการให้ค่า 5 กับตัวแปร x

| | |
|---|---|
| x < y | x is less than y |
| x <= y | x is less than or equal to y |
| x > y | x is greater than y |
| x >= y | x is greater than or equal to y |
| x == y | x is equal to y |
| x != y | x is not equal to y |

**รูปที่ 1** การเปรียบเทียบค่า

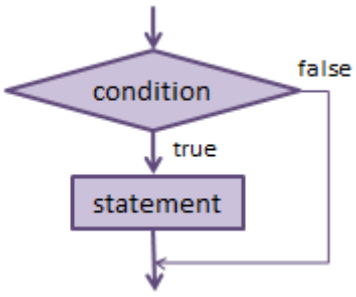
คำสั่งเงื่อนไขที่เราใช้ในภาษา C++ ได้แก่คำสั่ง if, if else, else if และ switch เราจะอธิบายการใช้ไปทีละคำสั่ง

## 5.2 คำสั่ง if

รูปที่ 2 แสดงการใช้คำสั่ง if เราใส่เงื่อนไข (condition) ไว้ในวงเล็บหลัง if เสมอ จากนั้นเราใส่คำสั่ง (statement) ที่เราต้องการให้ทำเมื่อเงื่อนไขเป็นจริงในบรรทัดเดียวกัน หรือคนละบรรทัดก็ได้ จากรูปที่ 3 ซึ่งเป็นผังงานของคำสั่ง if เราจะเห็นได้ว่า ถ้าเงื่อนไขเป็นจริง โปรแกรมจะทำคำสั่งนั้น แต่ถ้าเป็นเท็จก็จะข้ามไปไม่ทำอะไร

```
if(condition) statement;
```

**รูปที่ 2** รูปแบบการใช้คำสั่ง **if**




**รูปที่ 3** ผังงานการทำงานของคำสั่ง **if**

**ตัวอย่างที่ 1**

โจทย์ : จงเขียนโปรแกรมเพื่อรับจำนวนเต็ม 2 ตัว n และ d ตามลำดับ แล้วแสดงออกทางหน้าจอถ้า n หารด้วย d ไม่ลงตัว

1) เอาต์พุต คือให้แสดงว่าจำนวนเต็มที่รับเข้ามานั้นหารกันไม่ลงตัวถ้ามันหารกันไม่ลงตัว แต่ถ้าหารกันลงตัวก็ไม่ต้องแสดงอะไร

2) อินพุต คือจำนวนเต็ม 2 จำนวน กำหนดให้เป็น n และ d ตามลำดับ

3) โจทย์ไม่ได้กำหนดอะไรให้นอกเหนือจากนี้

4) เรารู้ว่า ถ้า n หารด้วย d ไม่ลงตัว จะมีเศษ (ไม่ใช่ 0) เพราะฉะนั้นเราจะใช้ตัวดำเนินการ % ในการตรวจสอบคุณสมบัตินี้ ดังนั้นเราต้องตรวจสอบโดยการใช้คำสั่ง if ว่า ถ้า n % d แล้วได้เศษคือค่าอะไรก็ได้ที่ไม่ใช่ 0 เราก็แสดงข้อความออกทางหน้าจอ ไม่เช่นนั้น ไม่ต้องแสดงอะไรออกมา

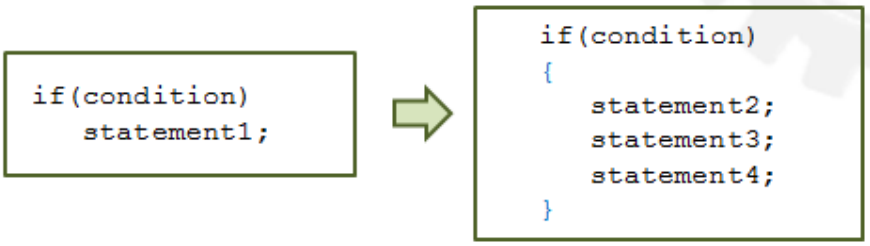
จากข้อมูล 4 ข้อดังกล่าวทำให้เราเขียนโค้ดได้ดังนี้

โค้ด:

```
int main(){
    int n, d;
    cout << "Enter 2 positive integers: ";
    cin >> n >> d;
    if (n % d) cout << n << " is not divisible by " << d << endl;
    return 0;
}
```

**บล็อคของคำสั่งและขอบเขตของตัวแปร (Statement block and variable scope)**

โดยปกติแล้ว การใช้คำสั่ง if แบบไม่มีบล็อคของคำสั่งในโปรแกรมภาษา C++ จะทำคำสั่งเดียวสำหรับแต่ละเงื่อนไขที่เป็นจริง ในกรณีที่เงื่อนไขเป็นจริงและเราต้องการให้โปรแกรมทำคำสั่งหลายคำสั่ง เราจำเป็นต้องเขียนบล็อคของคำสั่ง โดยการใส่เครื่องหมายปีกกา {..} ครอบกลุ่มคำสั่งเหล่านั้นไว้เพื่อให้มันอยู่ในบล็อค (block) เดียวกันและโปรแกรมจะทำคำสั่งทั้งหมดที่อยู่ในบล็อคนั้นเมื่อเงื่อนไขเป็นจริง ตามรูปที่ 4




**รูปที่ 4** รูปแบบการใช้บล็อคของคำสั่ง

เพื่อให้เห็นข้อแตกต่างระหว่างการใช้บล็อคและไม่ใช่บล็อคหลังคำสั่ง if เราดูตัวอย่างในรูปที่ 5 ใน Code 1 เราใส่คำสั่ง statement2 ถึง statement4 ไว้ในบล็อค โปรแกรมจะเห็นเป็น 1 ก้อนคำสั่ง เพราะฉะนั้นถ้าเงื่อนไขเป็นจริง ทั้ง 3 คำสั่งจะถูกทำทั้งหมด แต่ในกรณีที่เงื่อนไขเป็นเท็จ จะไม่มีคำสั่งใดเลยถูกทำ (คือมันจะข้ามการทำคำสั่งไปทั้งก้อน)

Code1:

```
if(condition)
{
    statement2;
    statement3;
    statement4;
}
```

Code2:

```
if(condition)
    statement2;
    statement3;
    statement4;
```

**รูปที่ 5** การเปรียบเทียบการใช้บล็อคคำสั่งและการไม่ใช้บล็อคคำสั่งหลังคำสั่ง **if**

ส่วนใน Code2 เรามี 3 คำสั่งเหมือนกันหลังคำสั่ง if แต่มันไม่ได้อยู่ในบล็อคเดียวกัน มีเพียง statement2 เท่านั้นที่เป็นคำสั่งที่ผูกกับเงื่อนไข ถ้าเงื่อนไขเป็นจริง statement2 จะถูกทำ แล้วจากนั้นมันจะทำ statement3 และ statement4 ต่อ แต่มันถือว่าทั้งสองคำสั่งหลังนั้นไม่ได้อยู่ในการกระทำเมื่อเงื่อนไขเป็นจริง กล่าวคือ มันถือเป็นส่วนอื่นของโปรแกรม อย่างไรก็ตาม ใน Code2 นี้เมื่อเงื่อนไขเป็นเท็จ โปรแกรมจะไม่ทำ statement2 แต่จะข้ามไปทำ statement3 และ statement4 ซึ่งถือเป็นส่วนอื่นของโปรแกรมไปเลย ต่างจากผลลัพธ์ของ Code1 ซึ่งจะไม่มีคำสั่งใดเลยที่ถูกทำ ดังที่อธิบายไปก่อนหน้านี้

เราจะเห็นได้ว่าการไม่ใส่ปีกกาหลังคำสั่ง if อาจจะทำให้สับสนและโปรแกรมทำงานผิดได้ ดังนั้นเราจึงแนะนำให้นักศึกษาใส่ปีกกาครอบคำสั่งที่ต้องการทำหลัง if เสมอ แม้ว่าจะมีแค่คำสั่งเดียวก็ตาม

ขอบเขตของตัวแปร (variable scope) เริ่มจากจุดที่ประกาศตัวแปร และไปสิ้นสุดที่บล็อคในสุดที่ตัวแปรนั้นถูกประกาศไว้ (ในขอบเขตของคู่ปีกกาที่ตัวแปรนั้นถูกประกาศอยู่) อย่างเช่นถ้าเราประกาศตัวแปรในฟังก์ชันเมน ตัวแปรนั้นก็จะใช้ได้แค่ในฟังก์ชันเมนเท่านั้น จะเอาไปใช้นอกฟังก์ชันไม่ได้ ในกรณีเดียวกัน ถ้าเราประกาศตัวแปรไว้ในคู่ปีกกาหลังคำสั่ง if ตัวแปรนั้นก็จะใช้ได้แค่ภายในคำสั่ง if เท่านั้น ในหนึ่งโปรแกรมเราสามารถมีตัวแปรหลายตัวที่มีชื่อเดียวกันได้ แต่ตัวแปรเหล่านั้นจะต้องถูกประกาศอยู่ในขอบเขตที่ไม่ต่อเนื่องหรือไม่ซ้อนกัน (disjoint) แต่ในทางปฏิบัติเราไม่แนะนำให้ทำถ้าไม่จำเป็น เพื่อป้องกันความสับสน

ตัวอย่างข้างล่างนี้ แสดงให้เห็นถึงการใช้บล็อคและขอบเขตของตัวแปร บล็อคใหญ่ของโค้ดนี้คือบล็อคของฟังก์ชันเมน ซึ่งมีตัวแปร x และ y ถูกประกาศอยู่ ทำให้ตัวแปรสองตัวนี้สามารถใช้ได้ทุกที่ในฟังก์ชันเมน ส่วนบล็อคเล็กที่อยู่ในฟังก์ชันเมน คือบล็อคสำหรับคำสั่ง if นั่นคือถ้าเงื่อนไขหลัง if เป็นจริงแล้ว โปรแกรมจะ

ทำทุกคำสั่งที่อยู่ในบล็อคหลัง if นี้ ซึ่งในบล็อคนี้มีการประกาศตัวแปร temp ทำให้ตัวแปร temp นั้นสามารถถูกใช้ได้แค่ในบล็อคนี้เท่านั้น เราไม่สามารถใช้ temp ได้ในส่วนอื่นของฟังก์ชันเมน

โค้ด:

```
int main(){
    int x, y;
    cout << "Enter 2 integers: ";
    cin >> x >> y;
    if(x > y) {
          int temp = x;
          x = y;
          y = temp;
    }
    cout << x << " <= " << y << endl;
    return 0;
}
```

**เงื่อนไขประกอบ (Compound condition)**

เงื่อนไขประกอบ (Compound conditions) คือการที่เงื่อนไขในการเลือกทำนั้นมีเงื่อนไขย่อยตั้งแต่ 2 เงื่อนไขขึ้นไป และการประกอบเงื่อนไขเพื่อเลือกทำนั้น เราต้องใช้ตัวดำเนินการทางตรรกะคือ และ (AND: ใช้เครื่องหมาย &&) กับหรือ (OR: ใช้เครื่องหมาย ||) ถ้าเงื่อนไขย่อยต่างๆ เชื่อมกันด้วย AND ทุกเงื่อนไขย่อยนั้นต้องเป็นจริง เพื่อที่จะทำให้เงื่อนไขในการเลือกทำนั้นเป็นจริง ส่วนถ้าเงื่อนไขย่อยต่างๆ เชื่อมกันด้วย OR ถ้ามีแค่ 1 เงื่อนไขย่อยเป็นจริงก็จะทำให้เงื่อนไขในการเลือกทำนั้นเป็นจริง ดังแสดงในรูปที่ 6 โดยกำหนดให้ p และ q เป็นเงื่อนไขย่อย

| p | q | p && q |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

| p | q | p \|\| q |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

| p | !p |
|---|---|
| 0 | 1 |
| 1 | 0 |

**รูปที่ 6** ค่าความจริงสำหรับการประกอบเงื่อนไขเข้าด้วยกันด้วย **&&, ||** และ **!**

นอกจากนี้ยังมีตัวดำเนินการที่เรียกว่า นิเสธ (NOT: ใช้เครื่องหมาย !) อีกด้วย NOT จะดำเนินการกับเงื่อนไข 1 เงื่อนไขย่อย โดยจะทำให้ค่าความจริงของเงื่อนไขนั้นกลับเป็นตรงกันข้าม นั่นคือถ้าเงื่อนไขย่อยนั้นเป็นจริง การใส่ ! เข้าไปข้างหน้าจะทำให้เงื่อนไขสำหรับการเลือกทำนั้นเป็นเท็จ แต่ถ้าเงื่อนไขย่อยนั้นเป็นเท็จ การใส่ ! เข้าไปข้างหน้าก็จะทำให้เงื่อนไขสำหรับการเลือกทำนั้นเป็นจริง

**ตัวอย่างที่ 2**

โจทย์ : จงเขียนโปรแกรมเพื่อรับจำนวนเต็มเข้ามาทางแป้นพิมพ์ 2 ตัว กำหนดให้เป็น n และ d จากนั้นให้ตรวจสอบว่า n และ d มีค่ามากกว่า 0 และแสดงออกจากหน้าจอ

1) เอาต์พุต คือถ้าจำนวนเต็มที่รับเข้ามานั้นมีค่ามากกว่า 0 ทั้ง 2 ตัว ให้แสดงออกทางหน้าจอว่าอินพุตถูกต้อง ไม่เช่นนั้นไม่ต้องแสดงอะไรออกมา
2) อินพุต คือจำนวนเต็ม 2 จำนวน กำหนดให้เป็น n และ d ตามลำดับ
3) โจทย์ไม่ได้กำหนดอะไรให้นอกเหนือจากนี้
4) ไม่มีข้อมูลเพิ่มเติม

ในโจทย์ข้อนี้ เราต้องตรวจสอบทั้ง n และ d ว่ามากกว่า 0 หรือไม่ เพราะฉะนั้นเราต้องใช้เงื่อนไขประกอบโดยการเชื่อมการตรวจสอบโดยเครื่องหมาย && (AND) เพราะเราต้องการให้ทั้งสองตัวแปรนั้นมีค่ามากกว่า 0 จึงจะแสดงออกทางหน้าจอ

โค้ด:

```
int main(){
    int n, d;
    cout << "Enter two positive integers: ";
    cin >> n >> d;
    if(n > 0 && d > 0)
    {
          cout << "inputs ok";
    }
    return 0;
}
```

**การลัดวงจรของการตรวจสอบเงื่อนไขประกอบ (Short circuiting)**

ในการประเมินเงื่อนไขประกอบนั้น โดยปกติแล้วโปรแกรมจะต้องประเมินเงื่อนไขย่อยทุกๆ เงื่อนไข แต่จากรูปที่ 6 เราเห็นได้ว่าในบางกรณีนั้นเราไม่จำเป็นจะต้องหาค่าความจริงในทุกๆ เงื่อนไข นั่นเพราะสำหรับ AND แล้ว ถ้าเงื่อนไขย่อยใดเงื่อนไขย่อยหนึ่งเป็นเท็จ ก็จะทำให้ทั้งเงื่อนไขเป็นเท็จโดยที่ไม่ต้องดูเงื่อนไขย่อยที่เหลือเลย เพราะถ้าเงื่อนไขย่อยที่เหลือเป็นจริงหรือเท็จ เงื่อนไขสำหรับเลือกทำก็จะเป็นเท็จอยู่ดี และเช่นกันสำหรับ OR แล้วถ้าเงื่อนไขย่อยใดเงื่อนไขย่อยหนึ่งเป็นจริง ก็จะทำให้ทั้งเงื่อนไขเป็นจริงโดยที่ไม่ต้องดูเงื่อนไขย่อยที่เหลือเลย เพราะถ้าเงื่อนไขย่อยที่เหลือเป็นจริงหรือเท็จ เงื่อนไขสำหรับเลือกทำก็จะเป็นจริงอยู่ดี เราเรียกวิธีการตรวจสอบเงื่อนไขย่อยอันใดอันหนึ่งและสรุปค่าความเป็นจริงได้เลยว่าการลัดวงจร หรือ short circuiting อย่างไรก็ตาม ถ้าเราตรวจสอบเงื่อนไขย่อยในการเชื่อมด้วย AND แล้วค่านั้นเป็นจริง เรายังต้องทำการตรวจสอบเงื่อนไขที่เหลือต่อไป เช่นเดียวกับถ้าเราตรวจสอบเงื่อนไขย่อยในการเชื่อมด้วย OR แล้วค่านั้นเป็นเท็จ เราก็ต้องตรวจสอบเงื่อนไขที่เหลือต่อไปเพราะเรายังสรุปไม่ได้

**ตัวอย่างที่ 3**

โจทย์ : จากโค้ดต่อไปนี้ กำหนดให้อินพุตจากแป้นพิมพ์คือ 2 และ 0 ตามลำดับ ให้ระบุว่าเราสามารถทำการลัดวงจรได้อย่างไร

โค้ด:

```
int main(){
    int n, d;
    cout << "Enter two positive integers: ";
    cin >> n >> d;
    if(d != 0 && n%d == 0)
    {
          cout << d << " divides " <<n << endl;
    }
    else
    {
          cout << d << " does not divide " << n << endl;
    }
    return 0;
}
```

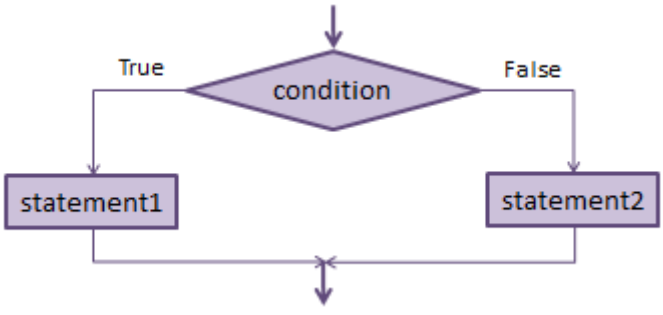
การลัดวงจรจะเกิดขึ้นได้เมื่อมีเงื่อนไขประกอบ ซึ่งในโค้ดนี้คือ d != 0 && n%d == 0 โจทย์กำหนดให้อินพุตคือ 2 และ 0 ตามลำดับ นั่นคือ n เท่ากับ 2 และ d เท่ากับ 0 เราเริ่มตรวจสอบเงื่อนไขอย่างแรก โดยการตรวจสอบว่า d != 0 เรารู้ว่า d เท่ากับ 0 ทำให้เงื่อนไข d != 0 เป็นเท็จ และเมื่อเงื่อนไขย่อยนี้เป็นเท็จแล้วก็ทำให้เงื่อนไขทั้งหมดเป็นเท็จ เพราะเงื่อนไขย่อยทั้งสองเชื่อมกันด้วย && (AND) ถ้าเงื่อนไขย่อยใดเป็นเท็จแล้ว เงื่อนไขทั้งหมดก็จะเป็นเท็จโดยไม่ต้องตรวจสอบเงื่อนไขที่เหลือ เราเรียกสภาพนี้ว่าการลัดวงจร

## 5.3 คำสั่ง if else

การทำงานของคำสั่ง if else นั้นเหมือนกับคำสั่ง if ทุกประการยกเว้นว่า ถ้าเงื่อนไข (ไม่ว่าจะเป็นเงื่อนไขเดี่ยวหรือเงื่อนไขประกอบ) เป็นเท็จ โปรแกรมจะต้องทำคำสั่งหรือบล็อคของคำสั่งอีกชุดหนึ่งแทน รูปแบบของคำสั่งในรูปที่ 7 และผังงานในรูปที่ 8 แสดงให้เห็นว่าถ้าเงื่อนไขเป็นจริง โปรแกรมจะทำ statement1 แต่ถ้าเงื่อนไขเป็นเท็จ โปรแกรมจะทำ statement2 ซึ่งถ้าเป็นคำสั่ง if จะไม่มีการทำ statement2 ถ้าเงื่อนไขเป็นเท็จ ดังแสดงในรูปที่ 2

```
if(condition) statement1;
else statement2;
```

**รูปที่** 7 รูปแบบคำสั่ง **if else**




**รูปที่** 8 ผังงานของคำสั่ง **if else**

**ตัวอย่างที่ 4**

โจทย์ : จงเขียนโปรแกรมเพื่อรับจำนวนเต็ม 2 จำนวน จากนั้นให้แสดงออกหน้าจอว่าจำนวนใดมีค่าน้อยที่สุด

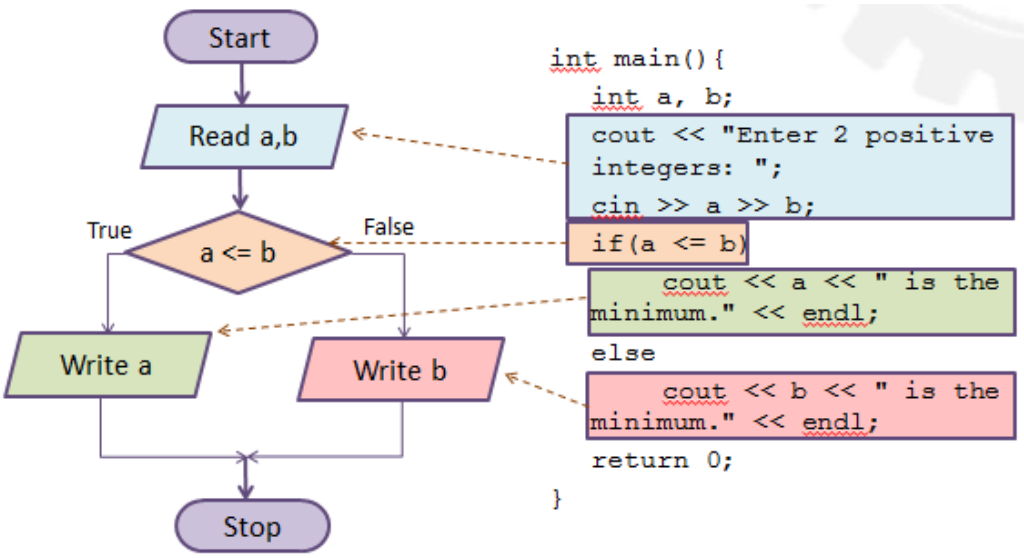
1) เอาต์พุต คือจำนวนที่น้อยที่สุดที่รับเข้ามา นั่นคือถ้า b น้อยกว่า ให้แสดงออกทางหน้าจอว่า b มีค่าน้อยที่สุด ไม่เช่นนั้น ให้แสดงออกทางหน้าจอว่า a มีค่าน้อยที่สุด
2) อินพุต คือจำนวนเต็ม 2 จำนวน กำหนดให้เป็น a และ b
3) โจทย์ไม่ได้กำหนดอะไรให้นอกเหนือจากนี้
4) ไม่มีข้อมูลอะไรเพิ่มเติม

จากข้อมูล 4 ข้อดังกล่าวทำให้เราเขียนโค้ดโดยใช้เครื่องหมาย <= ในการเปรียบเทียบค่าได้ดังนี้

โค้ด:

```
int main(){
	int a, b;
	cout << "Enter 2 positive integers: ";
	cin >> a >> b;
	if(a <= b)
	{
		cout << a << " is the minimum." << endl;
	}
	else
	{
		cout << b << " is the minimum." << endl;
	}
	return 0;
}
```

เราสามารถเขียนผังงานของโค้ดนี้ได้ตามรูปที่ 9 เราจะเห็นส่วนต่างๆ ของผังงานเทียบกับโค้ด คือถ้า a <= b เป็นจริงเราจะแสดงค่า a ออกทางหน้าจอ (ทางซ้ายของผังงาน) ถ้าเป็นเท็จโปรแกรมจะแสดงค่า b ออกทางหน้าจอ (ทางขวาของผังงาน)




**รูปที่ 9** ผังงานโปรแกรมหาค่าอินพุตที่น้อยที่สุด

**ตัวอย่างที่ 5**

โจทย์ : จงเขียนโปรแกรมเพื่อรับอักขระผ่านทางแป้นพิมพ์ จากนั้นให้แสดงออกหน้าจอว่านักศึกษาคนนี้ได้ลงทะเบียนเรียนหรือไม่ ถ้านักศึกษาใส่ y หรือ Y เข้ามา แปลว่านักศึกษาคนนี้ได้ลงทะเบียนเรียน ไม่เช่นนั้นให้แปลว่านักศึกษาคนนี้ไม่ได้ลงทะเบียนเรียน

1) เอาต์พุต คือข้อความว่านักศึกษาลงทะเบียนเรียนหรือไม่
2) อินพุต คืออักขระ 1 ตัว
3) โจทย์กำหนดเงื่อนไขของการลงทะเบียนเรียน
4) ไม่มีข้อมูลอะไรเพิ่มเติม

โจทย์ข้อนี้มีอินพุตเป็นอักขระซึ่งเป็นตัวเล็กหรือตัวใหญ่ก็ได้ เพราะฉะนั้นเราจึงสร้างเงื่อนไขประกอบเปรียบเทียบอักขระที่รับเข้ามากับทั้ง y เลยและ Y ใหญ่ ถ้าตรงกับอักขระตัวใดตัวหนึ่ง (OR) ก็ให้แสดงออกว่านักศึกษาคนนี้ลงทะเบียน ไม่เช่นนั้น (else) ก็แสดงออกทางหน้าจอว่าไม่ได้ลงทะเบียน

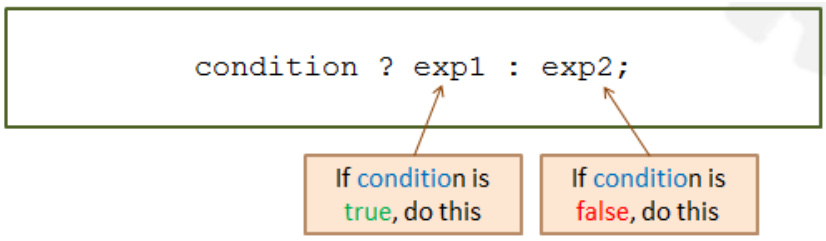
โค้ด:

```
int main(){
    char ans;
    cout << "Are you enrolled? (y/n) ";
```

```
    cin >> ans;
    if(ans == 'y' || ans == 'Y') cout << "Yes!" << endl;
    else cout << "No." << endl;
    return 0;
}
```

**ตัวดำเนินการสร้างนิพจน์แบบเงื่อนไข (Conditional expression operator)**

เราสามารถเขียนโค้ดให้ทำแบบคำสั่ง if else ได้โดยใช้ตัวดำเนินการสร้างนิพจน์แบบเงื่อนไข (conditional expression operator) ตามรูปแบบคำสั่งในรูปที่ 10 ในการเขียนคำสั่ง if else เรากำหนดว่าถ้าเงื่อนไข (condition) เป็นจริง เราจะทำ statement1 แต่ถ้าเป็นเท็จ เราจะทำ statement2 ในการใช้ตัวดำเนินการสร้างนิพจน์แบบเงื่อนไขก็เช่นกัน แต่ต่างกันตรงที่จะไม่มีคำว่า if else และผลลัพธ์ของคำสั่งจะเป็นนิพจน์ (ไม่ใช่ statement) ตามเงื่อนไขที่กำหนด ดังนั้นในการใช้ตัวดำเนินการสร้างนิพจน์แบบเงื่อนไข เราจะต้องมีตัวแปรมารองรับค่า เช่น int x = (2>3) ? 100 : 200; หรือว่าจะเขียนเป็นนิพจน์สำหรับคำสั่ง cout ก็ได้ เช่น cout << (2>3) ? 100 : 200;




**รูปที่ 10** รูปแบบการใช้ตัวดำเนินการสร้างนิพจน์แบบเงื่อนไข

ในการเขียนการเลือกทำด้วยตัวดำเนินการสร้างนิพจน์แบบเงื่อนไขนั้น เราจะเขียนเงื่อนไขไว้ข้างหน้าสุด ตามด้วยเครื่องหมายปรัศนี (?) จากนั้นเราจะใส่นิพจน์หากเงื่อนไขเป็นจริงลงไปในตำแหน่ง exp1 ตามด้วยเครื่องหมายโคลอน (:) แล้วตามด้วย exp2 ซึ่งเป็นนิพจน์หากเงื่อนไขเป็นเท็จ (เพื่อให้ง่าย เราจะอ่านว่าเงื่อนไขเป็นจริงหรือไม่ ถ้าจริงให้ค่าเท่ากับนิพจน์ที่อยู่ข้างหน้าเครื่องหมาย : ถ้าไม่จริงให้ค่าเท่ากับนิพจน์ที่อยู่ข้างหลังเครื่องหมาย :)

**ตัวอย่างที่ 6**

โจทย์ : จงเขียนโปรแกรมเพื่อรับจำนวนเต็ม 2 จำนวน จากนั้นให้แสดงออกทางหน้าจอว่าจำนวนใดมีค่าน้อยที่สุด โดยใช้ตัวดำเนินการสร้างนิพจน์แบบเงื่อนไขในการเปรียบเทียบและให้ค่า

1) เอาต์พุต คือการระบุว่าจำนวนเต็มใดที่รับเข้ามานั้นน้อยที่สุด นั่นคือถ้า b น้อยกว่า ให้แสดงออกทางหน้าจอว่า b มีค่าน้อยที่สุด ไม่เช่นนั้น ให้แสดงออกทางหน้าจอว่า a มีค่าน้อยที่สุด
2) อินพุต คือจำนวนเต็ม 2 จำนวน กำหนดให้เป็น a และ b
3) โจทย์กำหนดให้ใช้ตัวดำเนินการสร้างนิพจน์แบบเงื่อนไข
4) ไม่มีข้อมูลอะไรเพิ่มเติม

จากข้อมูล 4 ข้อดังกล่าวทำให้เราเขียนโค้ดได้ดังนี้

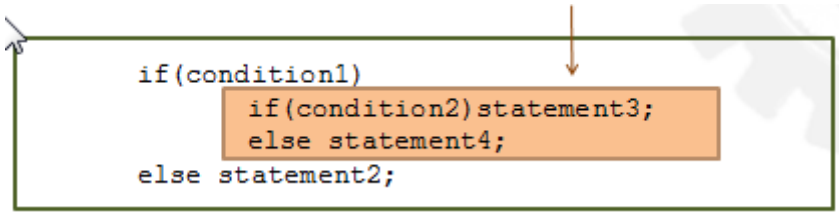
โค้ด:

```
int main(){
      int a, b;
      cout << "Enter 2 positive integers: ";
      cin >> a >> b;
      cout << (a <= b ? a : b) << " is the minimum." << endl;
      return 0;
}
```

เราจะเห็นได้ว่าถ้าเราใช้ตัวดำเนินการสร้างนิพจน์แบบเงื่อนไขแล้วเราสามารถเปรียบเทียบค่า 2 ค่าและแสดงค่าที่ต้องการในคำสั่ง cout คำสั่งเดียวได้เลย ไม่จำเป็นต้องใช้คำสั่ง if else

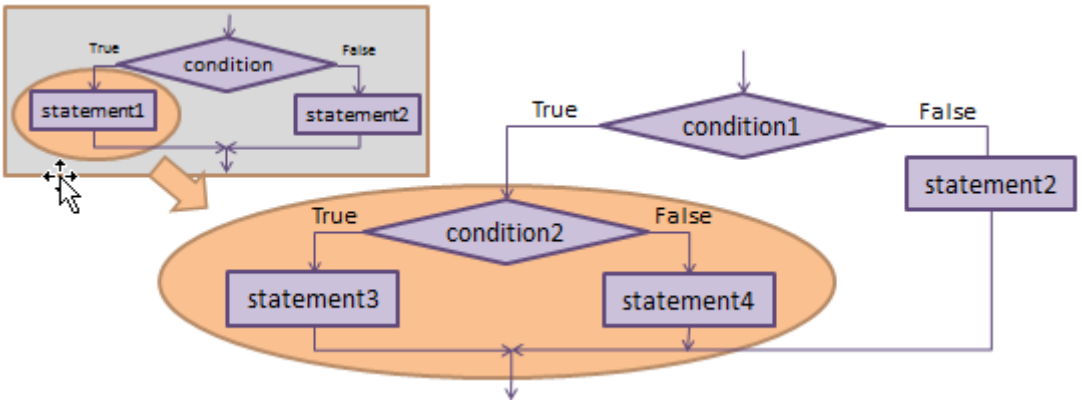
## 5.4 การเลือกทำแบบซ้อนกัน (Nested selection)

เวลาที่เราใช้คำสั่งเพื่อเลือกทำ เราสามารถซ้อนคำสั่งการเลือกทำเข้าไปได้อีกชั้นหนึ่ง โดยแทนที่คำสั่ง (statement) ต่างๆ ตามรูปที่ 11 เราเรียกการซ้อนคำสั่งการเลือกทำเข้าไปว่า nested selection




**รูปที่ 11** รูปแบบคำสั่งของการเลือกทำแบบซ้อนกัน

จากรูปที่ 12 เราจะเห็นการทำงานของการเลือกทำแบบซ้อนกัน นั่นคือเมื่อเงื่อนไข condition1 เป็นจริง แทนที่โปรแกรมจะทำ statement1 (เหมือนที่ทำในคำสั่ง if และ if else) มันจะทำการตรวจสอบเงื่อนไขอีกชุดหนึ่ง นั่นคือ condition2 ถ้า condition2 เป็นจริง โปรแกรมก็จะทำ statement3 ถ้า condition2 เป็นเท็จ ก็จะทำ statement4 แต่ถ้าเงื่อนไขแรก condition1 เป็นเท็จ โปรแกรมจะทำ statement2 เลย และจะไม่มีการตรวจสอบเงื่อนไข condition2

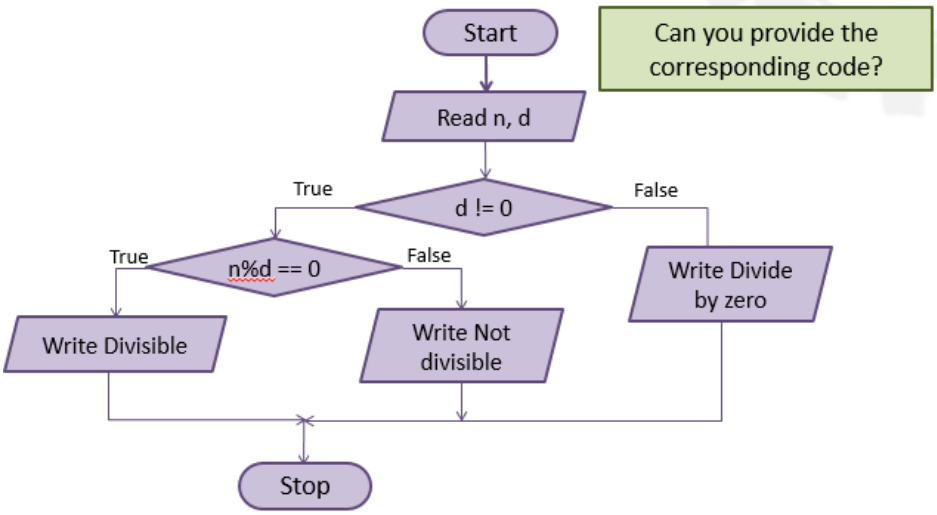



**รูปที่ 12** ผังงานของการเลือกทำแบบซ้อนกัน

เราใช้การเลือกทำแบบซ้อนกันในกรณีที่เราต้องการตรวจสอบหลายๆ เงื่อนไขก่อนที่จะทำคำสั่งใดคำสั่งหนึ่ง เช่น เราต้องการตรวจสอบว่าผู้ใช้งานเป็นนักศึกษาหรือไม่ ถ้าใช่ก็ไปตรวจสอบอีกว่าผู้ใช้งานเป็นนักศึกษาปี 4 หรือไม่ ถ้าใช่ก็ให้ลงทะเบียนได้ ถ้าไม่ใช่นักศึกษาปี 4 ก็ไม่ให้ลงทะเบียน แต่ถ้าผู้ใช้งานไม่ได้เป็นนักศึกษาก็อาจจะไม่ให้เข้าใช้ระบบเลย เป็นต้น นอกจากนี้ เราสามารถใส่การเลือกทำแบบซ้อนกันเข้าไปในคำสั่ง if หรือ if else แบบไม่จำกัด แต่ต้องระวังในการเขียนโปรแกรมเพราะถ้าใส่เข้ามากไป อาจจะทำให้งงได้ ดังนั้นการใส่ปีกกาเพื่อกำหนดขอบเขตของคำสั่ง if หรือ if else แต่ละบล็อคสำคัญมาก เพื่อความไม่สับสนเราแนะนำให้ใส่ปีกกาครอบไว้เสมอ แม้หลัง if หรือ else จะมีคำสั่งเดียวก็ตาม

**ตัวอย่างที่ 7**

โจทย์: จากผังงานที่กำหนดให้ในรูปที่ 13 จงเขียนโปรแกรมด้วยคำสั่งแบบ nested selection



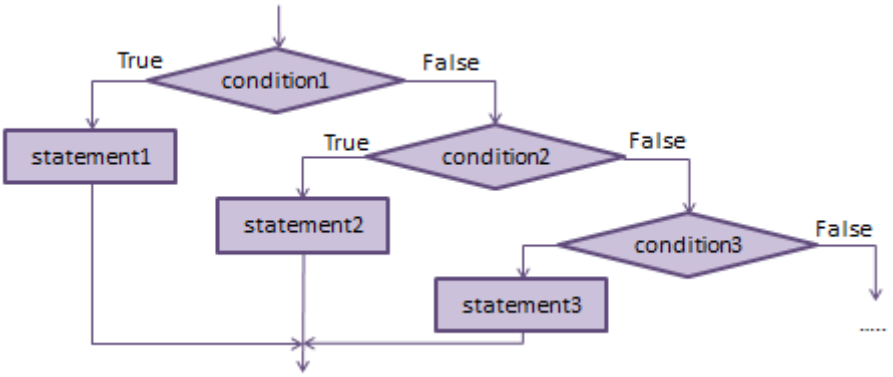

**รูปที่ 13** ผังงานของโจทย์

โค้ด:

```
int main(){
    int n,d;
    cin >> n >> d;
    if(d != 0){
          if(n%d == 0){
                cout << n << " is divisible by " << d << endl;
          }
          else cout << n << " is not divisible by " << d << endl;
    }
    else cout << n << " is divided by 0" << endl;
    return 0;
}
```

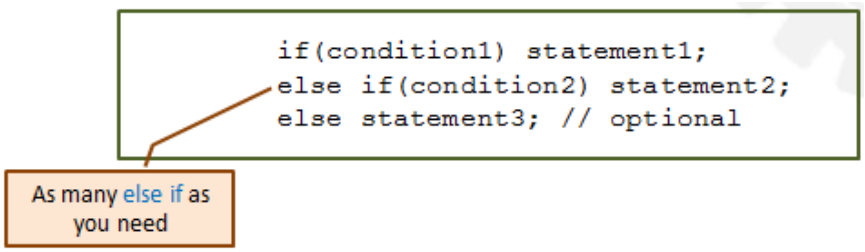
## 5.5 คำสั่ง else if

ในกรณีที่เราต้องการตรวจสอบเงื่อนไขหลายๆ เงื่อนไข อาจจะทำให้การเขียนโปรแกรมโดยใช้คำสั่ง if else แบบซ้อนกันมีความซับซ้อนและอ่านยาก เช่นในโปรแกรมที่ให้ผู้ใช้เลือกภาษา 5 ภาษา ถ้าผู้ใช้เลือกภาษาใด ให้โปรแกรมแสดงคำในภาษานั้นออกมา เราจะต้องมี if else ซ้อนกันถึง 4-5 ชั้นด้วยกัน ดังนั้นเราจึงต้องอาศัยคำสั่ง else if เข้ามาช่วย การทำงานของคำสั่ง else if นั้นคือการตรวจสอบเงื่อนไขต่างๆ ไปเรื่อยๆ จนกว่าจะเจอเงื่อนไขที่เป็นจริงแล้วทำคำสั่งของเงื่อนไขนั้น จากนั้นก็จบคำสั่งการเลือกทำชุดนั้นไปเลย โดยไม่ต้องตรวจสอบเงื่อนไขที่เหลืออีก ดังแสดงในรูปที่ 14




**รูปที่ 14** ผังงานการทำงานของคำสั่ง **else if**

รูปแบบของคำสั่ง else if นั้นแสดงในรูปที่ 15 เราจะเห็นได้ว่าคำสั่ง else if นั้นใช้ร่วมกับคำสั่ง if และเขียนเรียงกันลงมา ทำให้อ่านง่ายกว่าการใช้คำสั่ง if else แบบซ้อนกัน ในการใช้คำสั่ง else if นั้น ลำดับของเงื่อนไขมีความสำคัญมากและเราต้องวางเรียงอย่างระมัดระวัง เพราะถ้าโปรแกรมเจอว่าเงื่อนไขใดเป็นจริง ก็จะทำคำสั่งของเงื่อนไขนั้นเงื่อนไขเดียว จะไม่ตรวจสอบเงื่อนไขอื่นๆ ที่อยู่ทีหลังต่อไป แต่จะจบการเลือกทำไปเลย




**รูปที่ 15** รูปแบบการใช้คำสั่ง **else if**

**ตัวอย่างที่ 8** โปรแกรมตัดเกรด

โจทย์ : จงเขียนโปรแกรมเพื่อรับค่าคะแนนของนักศึกษาผ่านทางแป้นพิมพ์ จากนั้นให้แสดงเกรดสำหรับนักศึกษาคนนั้นตามตารางที่กำหนดให้ข้างล่างนี้ออกทางหน้าจอ

| Score | Grade |
|---|---|
| ≥ 80 | A |
| ≥ 70 && < 80 | B |
| ≥ 60 && < 70 | C |
| ≥ 50 && < 60 | D |
| < 50 | F |

1) เอาต์พุต คือเกรดของนักศึกษา
2) อินพุต คือคะแนนของนักศึกษา
3) โจทย์กำหนดตารางคะแนนและเกรดสำหรับช่วงคะแนนนั้นๆ
4) ไม่มีข้อมูลอะไรเพิ่มเติม

โจทย์ข้อนี้จะต้องมีการตรวจสอบหลายเงื่อนไข โดยเอาคะแนนที่รับเข้ามานั้นไปเปรียบเทียบกับคะแนนในช่วงของเกรดต่างๆ เช่น ถ้าคะแนนที่รับเข้ามานั้นมากกว่าหรือเท่ากับ 80 ก็จะได้เกรด A แต่ถ้าคะแนนน้อยกว่า 80 แต่ (และ) มากกว่าหรือเท่ากับ 70 ก็จะได้เกรด B เราสามารถใช้คำสั่ง else if มาช่วยได้ เนื่องจาก 1) มีการตรวจสอบเงื่อนไขหลายครั้งและ 2) การตรวจสอบนั้นขึ้นอยู่กับการเรียงลำดับก่อนหลังได้

โค้ด:

```
int main(){
      float score;
      char grade;
      cout << "Enter a score (0-100): ";
      cin >> score;

      if(score >= 80) grade = 'A';
      else if(score >= 70) grade = 'B';
      else if(score >= 60) grade = 'C';
      else if(score >= 50) grade = 'D';
      elsegrade = 'F';
```

```
    cout << "Score = " << score << ", grade = " << grade
    << endl;
    return 0;
}
```

จากโค้ดที่แสดง เราจะเห็นได้ว่าเมื่อเราใช้คำสั่ง else if หลังจากการเปรียบเทียบเงื่อนไขแรก (score >= 80) และถ้ามันเป็นเท็จแล้ว เราจะรู้ว่าคะแนนที่ใส่เข้ามานั้นต้องน้อยกว่า 80 แน่นอน เพราะฉะนั้นในการเปรียบเทียบเงื่อนไขที่สอง เราไม่จำเป็นต้องเปรียบเทียบว่า score < 80 หรือไม่อีก เราจึงเปรียบเทียบแค่ score >= 70 เช่นเดียวกันกับในเงื่อนไขถัดมา เราไม่จำเป็นต้องเปรียบเทียบว่า score < 70 หรือไม่อีก เพราะถ้าโปรแกรมทำการเปรียบเทียบเงื่อนไขนี้ นั่นแสดงว่าเงื่อนไขก่อนหน้านี้ทั้งหมดเป็นเท็จนั่นเอง (นั่นคือคะแนนต้องน้อยกว่า 70 แน่นอน)

**ตัวอย่างที่ 9** เครื่องคิดเลข

โจทย์ : จงเขียนโปรแกรมเพื่อรับค่า 3 ค่าผ่านทางแป้นพิมพ์ เป็นค่าจำนวนจริง 2 จำนวน a และ b และ ตัวอักขระเครื่องหมายทางคณิตศาสตร์ (+, -, *, /) op ตามลำดับ จากนั้นให้แสดงผลการคำนวณ a op b ออกทางหน้าจอ เช่น รับค่า 1 2 และ + ให้แสดงค่า 3 ออกทางหน้าจอ

1) เอาต์พุต คือผลการคำนวณ a op b
2) อินพุต คือจำนวนจริง a และ b และอักขระ + - * หรือ / ตามลำดับ
3) โจทย์ไม่ได้กำหนดอะไรเพิ่มเติม
4) ไม่มีข้อมูลอะไรเพิ่มเติม

จากข้อมูลทั้ง 4 ข้อซึ่งค่อนข้างตรงไปตรงมา เราสามารถเขียนโค้ดได้ดังนี้

โค้ด:

```
int main(){
    float a, b, result;
    char op;
```

```
        cout << "Enter a and b: ";
        cin >> a >> b;

        cout << "Enter op (+ -* /): ";
        cin >> op;

        if(op == '+') result = a+b;
        else if(op == '-') result = a-b;
        else if(op == '*') result = a*b;
        else if(op == '/') result = a/b;

        cout << a << " " << op << " " << b << " = "
        << result << endl;

        return 0;
}
```

**ข้อแตกต่างของคำสั่ง else if และคำสั่ง if หลายคำสั่งเรียงกัน**

เราจะสังเกตได้ว่า รูปแบบของการใช้คำสั่ง else if กับการใช้คำสั่ง if หลายๆ คำสั่ง (มี if หลายๆ ตัวเรียงกันลงมา) นั้นใกล้เคียงกันมากดังรูปที่ 16 อย่างไรก็ตาม ข้อแตกต่างของทั้งสองคำสั่งนี้คือ ในชุดคำสั่ง else if นั้น จะมีคำสั่งของเงื่อนไขเดียวเท่านั้นที่จะถูกทำ นั่นคือ เมื่อโปรแกรมไล่ตรวจสอบเงื่อนไขจากบนลงล่างแล้ว โปรแกรมจะทำคำสั่งของเงื่อนไขแรกที่เราพบว่าเป็นจริงเท่านั้น แล้วจะไม่ตรวจสอบเงื่อนไขอื่นอีก ในทางตรงกันข้าม สำหรับการใช้คำสั่ง if หลายๆ คำสั่งต่อกัน โปรแกรมจะตรวจสอบทุกเงื่อนไข ถ้าเงื่อนไขไหนเป็นจริงก็จะทำคำสั่งของเงื่อนไขนั้น ทำให้โปรแกรมสามารถทำคำสั่งได้หลายคำสั่ง

Code #1:

```
if(condition1) st1;
else if(condition2) st2;
else if(condition3) st3;
else if(condition4) st4;
```

Code #2:

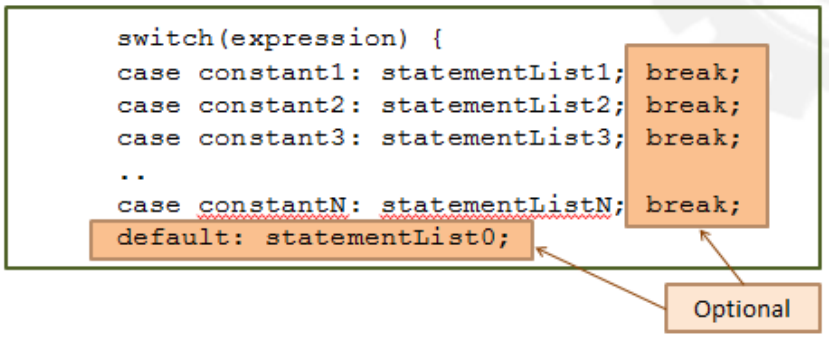
```
if(condition1) st1;
if(condition2) st2;
if(condition3) st3;
if(condition4) st4;
```

**รูปที่ 16** ข้อแตกต่างระหว่างคำสั่ง else if และ if หลายคำสั่งต่อกัน

จากรูปที่ 16 ถ้าเรากำหนดให้เงื่อนไข condition2 และ condition4 เป็นจริง การใช้คำสั่ง else if ทางซ้ายมือนั้นจะตรวจสอบแค่เงื่อนไข condition1 และ condition2 เมื่อพบว่า condition2 เป็นจริงก็จะทำ st2 แล้วจบการเลือกทำไปโดยไม่ตรวจสอบ condition3 และ condition4 อีก ส่วนการใช้คำสั่ง if หลายๆ คำสั่งต่อกันนั้นจะตรวจสอบทุกเงื่อนไขคือทั้งเงื่อนไข condition1 – condition4 เมื่อพบว่าเงื่อนไข condition2 และ condition4 เป็นจริง ก็จะทำทั้ง st2 และ st4 มันไม่ทำ st1 และ st3 เพราะว่า condition1 และ condition3 เป็นเท็จ

## 5.6 คำสั่ง switch

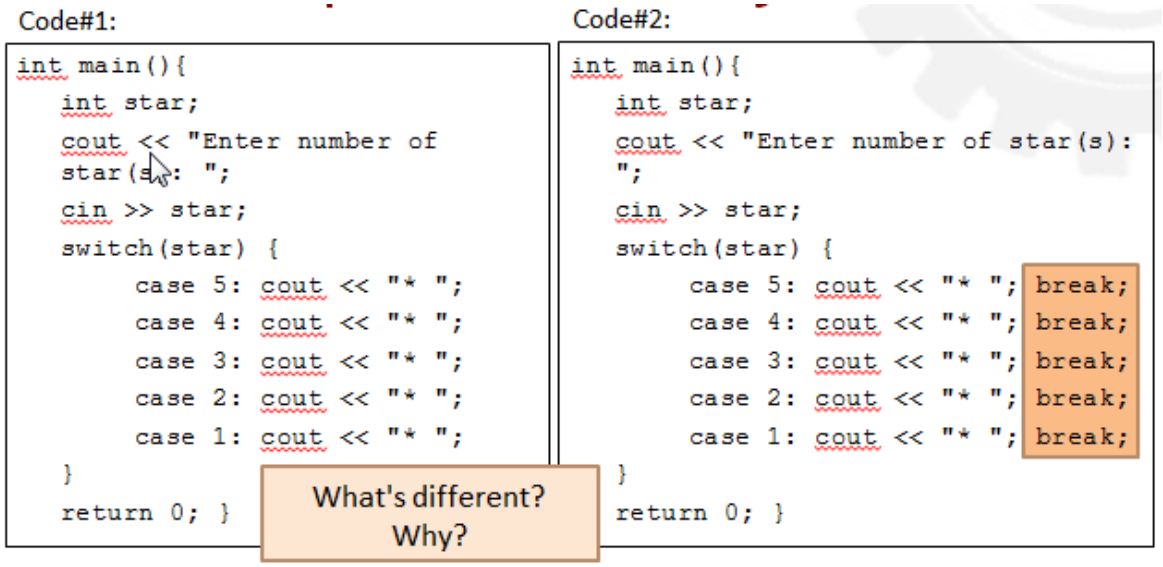
คำสั่ง switch เป็นคำสั่งการเลือกทำอีกคำสั่งหนึ่ง ซึ่งเราสามารถใช้แทนคำสั่ง else if ได้ในบางกรณี และในรูปแบบที่ค่อนข้างแตกต่างออกไป ในการทำคำสั่ง else if เราใช้ค่าความเป็นจริงของเงื่อนไขซึ่งส่วนมากจะอยู่ในรูปของการเปรียบเทียบค่าต่างๆ เช่น มากกว่าหรือน้อยกว่า แต่ในคำสั่ง switch นั้น เราจะทำการเปรียบเทียบนิพจน์ (expression) ซึ่งส่วนมากจะเป็นค่าของตัวแปรกับค่าคงที่ (constant) ว่าเท่ากันหรือไม่ ถ้าเท่ากันก็ให้ทำคำสั่ง โดยเริ่มต้นที่กรณีนั้น และทำไปเรื่อยๆ จนจบคำสั่ง switch ยกเว้นจะเจอคำสั่ง break ก่อน (ดูการอธิบายคำสั่ง break เพิ่มเติม) แต่ถ้าค่าของนิพจน์และค่าคงที่ไม่เท่ากัน ก็ให้เปรียบเทียบนิพจน์นั้นกับค่าคงที่ตัวถัดไปเรื่อยๆ จนจบคำสั่ง รูปแบบการใช้คำสั่ง switch แสดงในรูปที่ 17




**รูปที่ 17** รูปแบบการใช้คำสั่ง **switch**

ในกรณีที่ไม่มีค่าคงที่ใดที่เท่ากับค่าของนิพจน์ เราสามารถเขียนกรณีการทำโดยปริยาย (default) ไว้แล้วโปรแกรมก็จะทำคำสั่งในกรณี default แทน ซึ่งเทียบได้กับ else ในคำสั่ง if else เช่นจากรูปที่ 17 ถ้าค่านิพจน์ไม่ตรงกับค่า constant1 ถึง constantN เลย โปรแกรมจะทำ statementList0 ใน default แทน ให้สังเกตว่าก่อนคำว่า default นั้นเราไม่ต้องใส่คำว่า case เพราะ default แปลว่าตัวเลือกอัตโนมัติ หรือค่าโดยปริยายอยู่แล้ว

นอกจากนี้ เรายังมีคำสั่ง break ซึ่งแปลว่าหยุด ถ้าเราต้องการให้โปรแกรมทำแค่ชุดคำสั่งในกรณีที่ค่าเท่ากับนิพจน์เท่านั้น เราจำเป็นต้องใส่คำสั่ง break เมื่อจบชุดคำสั่งในกรณีนั้นๆ ไม่เช่นนั้นโปรแกรมจะทำชุดคำสั่งของกรณีต่อมาทั้งหมด เช่นจากรูปที่ 17 ถ้านิพจน์มีค่าเท่ากับ constant3 แล้ว และไม่มีการใส่ break ไว้ โปรแกรมจะทำ statementList3, statementList4 ไปเรื่อยๆ จนถึง statementList0 เลย (จบคำสั่ง switch) เรามาดูตัวอย่างในรูปที่ 18 เพื่อเปรียบเทียบกรณีที่มี break และไม่มี break



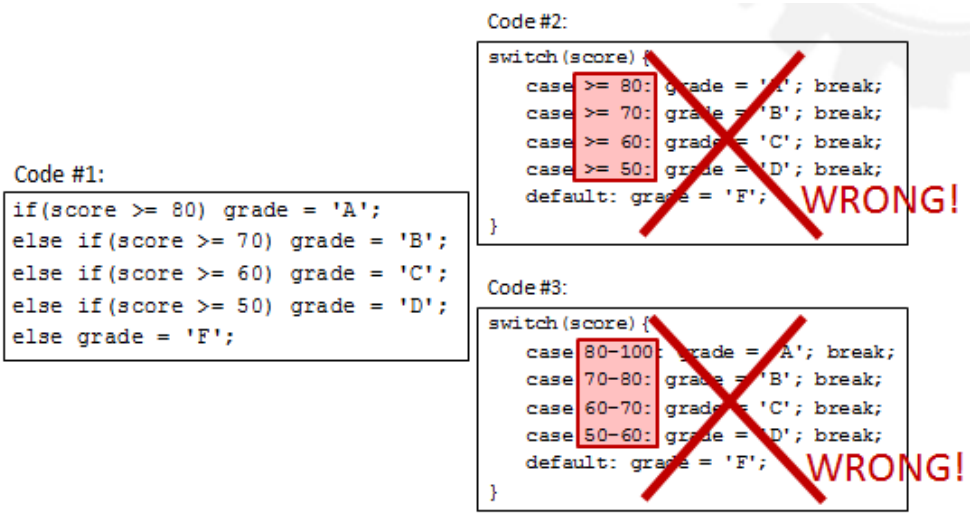

**รูปที่ 18** เปรียบเทียบการใช้คำสั่ง switch แบบมีคำสั่ง break และไม่มีคำสั่ง break

รูปที่ 18 เป็นการใช้คำสั่ง switch เพื่อแสดงเครื่องหมายดอกจัน * ออกทางหน้าจอ โดยโปรแกรมรับอินพุตเป็นเลขจำนวนเต็ม (star) ผ่านทางหน้าจอ จากนั้นคำสั่ง switch จะตรวจสอบว่าอินพุต star มีค่าเท่ากับค่าใดใน 5, 4, 3, 2 หรือ 1 ใน Code#1 ทางซ้าย ถ้ามีค่าเท่ากับ 5 โปรแกรมจะพิมพ์ * ออกมา 5 ครั้ง คือเริ่มจาก case 5 จากนั้นจึงมาแสดง * หลัง case 4, 3, 2, และ 1 ตามลำดับ เพราะเราไม่มีคำสั่ง break ในสักกรณี โปรแกรมจึงทำต่อเรื่อยมา ถ้าในกรณีที่ star มีค่าเท่ากับ 3 โปรแกรมจะเริ่มเลือกทำตั้งแต่ case 3 ซึ่งจะทำให้โปรแกรมพิมพ์ * ออกมาเป็นจำนวน 3 ตัว

สำหรับ Code#2 ทางขวานั้น เราใส่ break ท้ายทุกกรณี ถ้า star มีค่าเท่ากับ 5 โปรแกรมจะแสดง * ใน case 5 แล้วเจอ break ก็จะหยุดทำและออกจากคำสั่ง switch ไปเลย ทำให้โปรแกรมแสดง * ได้แค่หนึ่งตัวเท่านั้น เช่นเดียวกับถ้า star มีค่าเท่ากับ 3 โปรแกรมจะแสดง * ใน case 3 แล้วเจอ break จึงจะออกจากคำสั่ง switch ไปเลย ทำให้โปรแกรมแสดง * แค่ตัวเดียวเหมือนกัน

**ข้อแตกต่างของคำสั่ง switch และคำสั่ง else if**

อย่างที่กล่าวไปแล้ว เราจะเห็นได้ว่า ในการใช้ switch นั้น เราจะเปรียบเทียบค่าในนิพจน์กับค่าคงที่ว่าเท่ากันหรือไม่เท่านั้น จะไม่มีการเปรียบเทียบว่ามากกว่าหรือน้อยกว่าเหมือนในคำสั่ง else if ไม่เช่นนั้นโปรแกรมจะคอมไพล์ไม่ผ่านและรันไม่ได้ เพราะเป็นการผิดรูปแบบของคำสั่ง switch




**รูปที่ 19** ข้อแตกต่างของคำสั่ง **switch** และคำสั่ง **else if**

แล้วถ้าเราต้องการแปลงจาก code #1 ในรูปที่ 19 คือมีการเปรียบเทียบว่าน้อยกว่าหรือมากกว่าหรือเปรียบเทียบเป็นช่วงค่าโดยใช้คำสั่ง switch เราจะต้องทำอย่างไร เราสามารถใช้คุณสมบัติการหารเลขจำนวนเต็มเข้ามาช่วย ดังตัวอย่างต่อไปนี้

**ตัวอย่างที่ 10**

โจทย์ : จงเขียนโปรแกรมเพื่อรับค่าคะแนนของนักศึกษาผ่านทางแป้นพิมพ์ จากนั้นแสดงเกรดสำหรับนักศึกษาคนนั้นตามตารางที่กำหนดให้ข้างล่างนี้ออกทางหน้าจอ โดยใช้คำสั่ง switch

| Score | Grade |
|---|---|
| ≥ 80 | A |
| ≥ 70 && < 80 | B |
| ≥ 60 && < 70 | C |
| ≥ 50 && < 60 | D |
| < 50 | F |

1) เอาต์พุต คือเกรดของนักศึกษา
2) อินพุต คือคะแนนของนักศึกษา

3) โจทย์กำหนดตารางคะแนนและเกรดสำหรับคะแนนนั้นๆ และให้ใช้คำสั่ง switch

4) ไม่มีข้อมูลอะไรเพิ่มเติม

ในตัวอย่างนี้เราไม่สามารถใช้คำสั่ง else if ได้ เราต้องใช้คำสั่ง switch เราจึงไม่สามารถเปรียบเทียบโดยใช้เครื่องหมาย <= หรือ >= ได้ เราจึงต้องแปลงช่วงคะแนนให้เป็นค่าคงที่โดยใช้คุณสมบัติของเลขจำนวนเต็มที่เมื่อหารกันแล้วจะปัดเศษลง ดังนั้นถ้าเราหารค่าคะแนน 0-100 ด้วย 10 เราจะได้ค่าคงที่จาก 0-10 เช่น 79/10 ได้ 7 ส่วน 72/10 ก็ได้ 7 เช่นกัน เพราะฉะนั้นคะแนนในช่วง 70-79 หาร 10 ก็จะเป็นค่า 7 เท่ากัน หรือคะแนนในช่วง 40-49 หาร 10 ก็จะได้ค่า 4 เท่ากันเช่นกัน ดังนั้นเราสามารถใช้ค่าคงที่จากการหาร 10 คือ 0-10 เหล่านี้เป็นค่าคงที่ในแต่ละ case ของคำสั่ง switch สำหรับคะแนน 0-100 แล้วให้ค่าคงที่เหล่านี้เป็นแต่ละเคสของคำสั่ง switch

โค้ด:

```
int main(){
	float score;
	char grade;
	cout << "Enter a score (0-100): ";
	cin >> score;

	switch((int)score/10){
		case 10:
		case 9:
		case 8: grade = 'A'; break;
		case 7: grade = 'B'; break;
		case 6: grade = 'C'; break;
		case 5: grade = 'D'; break;
		default: grade = 'F';
	}

	cout << "Score = " << score << ", grade = " << grade <<
endl;
	return 0;
}
```

**ตัวอย่างที่ 11** เครื่องคิดเลข

โจทย์ : จงเขียนโปรแกรมเพื่อรับค่า 3 ค่าผ่านทางแป้นพิมพ์ เป็นค่าจำนวนจริง 2 จำนวน a และ b และ ตัวอักขระเครื่องหมายทางคณิตศาสตร์ (+, -, *, /) op ตามลำดับ จากนั้นให้แสดงผลการคำนวณ a op b ออกทางหน้าจอ เช่น รับค่า 1 2 และ + ให้แสดงค่า 3 ออกทางหน้าจอ (1+2 = 3) กำหนดให้ใช้คำสั่ง switch

1) เอาต์พุต คือผลการคำนวณ a op b
2) อินพุต คือจำนวนจริง a และ b และอักขระ + - * หรือ / ตามลำดับ
3) โจทย์ไม่ได้กำหนดอะไรเพิ่มเติม
4) ไม่มีข้อมูลอะไรเพิ่มเติม

โค้ด:

```
int main(){
      float a, b, result;
      char op;
      cout << "Enter a and b: ";
      cin >> a >> b;

      cout << "Enter op (+ -* /): ";
      cin >> op;

      switch(op){
            case '+': result = a+b; break;
            case '-': result = a-b; break;
            case '*': result = a*b; break;
            case '/': result = a/b; break;
      }

      cout << a << " " << op << " " << b << " = "
      << result << endl;

      return 0;
}
```

**สรุปสิ่งที่ควรรู้**

1) การเขียนการเลือกทำ (selection) นั้น โปรแกรมจะเลือกทำคำสั่งถ้าเงื่อนไขเป็นจริง และอาจจะทำคำสั่งอื่นหากเงื่อนไขเป็นเท็จ
2) เงื่อนไขมี 2 ชนิดคือ เงื่อนไขเดี่ยว และเงื่อนไขประกอบ เงื่อนไขประกอบจะเชื่อมกันด้วยตัวดำเนินการ && || หรือ !
3) คำสั่งสำหรับการเลือกทำในภาษา C++ ได้แก่ if, if else, else if, และ switch
4) ข้อควรระวังในการเปรียบเทียบในเงื่อนไข
    a. ใช้ >= ไม่ใช่ ≥
    b. ใช้ <= ไม่ใช่ ≤
    c. ใช้ != ไม่ใช่ ≠
    d. ใช้ == ไม่ใช่ = โดยเครื่องหมาย = ใช้ในการให้ค่า ไม่ใช่ใช้เปรียบเทียบ
5) คำสั่ง if และ else if จะทำ 1 บล็อคคำสั่งที่เงื่อนไขเป็นจริงลำดับแรกเท่านั้นเสมอ ส่วนถ้าเราใช้ if หลายๆ คำสั่งเรียงกันลงมา โปรแกรมจะตรวจสอบและทำทุกบล็อคคำสั่งที่เงื่อนไขหลัง if เป็นจริง
6) สำหรับคำสั่ง switch โปรแกรมจะตรวจสอบค่าของนิพจน์ว่าตรงกับค่าคงที่ของกรณีใดแล้วทำกรณีนั้น เราไม่สามารถตรวจสอบค่าเป็นช่วงได้ ได้แค่ค่าคงที่เท่านั้น
7) สำหรับคำสั่ง switch ถ้าเราเขียนล้อคำสั่ง else แล้ว เราต้องใส่คำสั่ง break เสมอ ไม่เช่นนั้นโปรแกรมจะทำคำสั่งของกรณีอื่นๆ ที่ตามมาด้วย