บทที่ 4

การตรวจสอบเงื่อนไขและการทำซ้ำ

ในการเขียนโปรแกรมนั้น นอกเหนือจากการเขียนในลักษณะเป็นชุดคำสั่งเรียงลำดับจากบนลงล่างแล้ว ในบางครั้งเรายังจำเป็นต้องเขียนโปรแกรมที่มีลักษณะต่อไปนี้

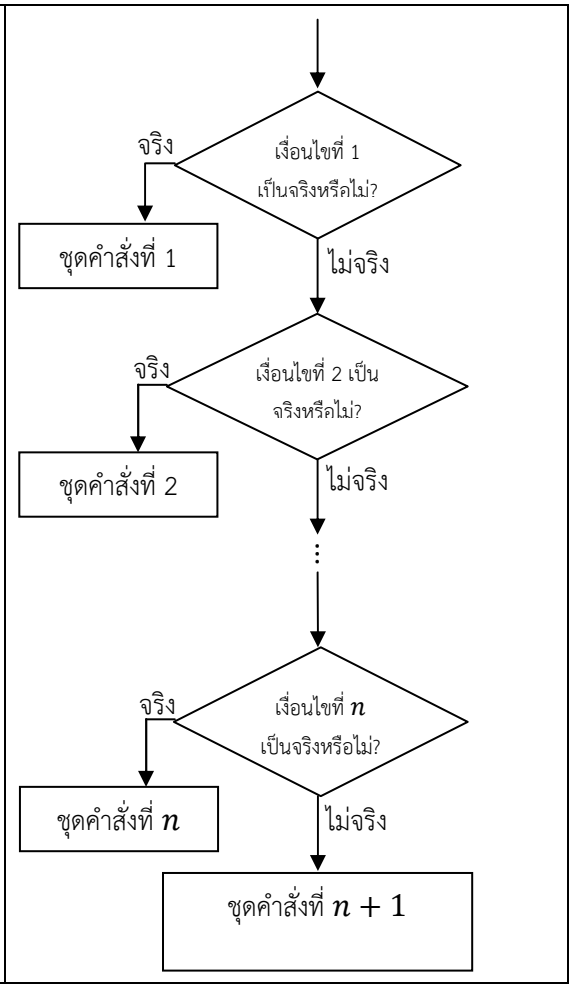
1. โปรแกรมที่ดำเนินการตามชุดคำสั่งที่ที่ขึ้นอยู่กับเงื่อนไข (condition) บางอย่าง เช่น ถ้าเงื่อนไขที่ว่า "x มีค่ามากกว่า 1 หรือไม่?" มีค่าตรรกะเป็นจริง แล้วโปรแกรมจะทำตามชุดคำสั่ง A แต่ถ้าเงื่อนไขดังกล่าวไม่เป็นจริง แล้วโปรแกรมจะทำตามชุดคำสั่ง B แทน เป็นต้น
2. โปรแกรมที่ดำเนินการตามชุดคำสั่งหนึ่ง ๆ ซ้ำไปซ้ำมาหลาย ๆ ครั้ง โดยที่จำนวนครั้งที่ทำซ้ำนั้นอาจเป็นจำนวนที่มีค่าแน่นอน เช่น ทำตามชุดคำสั่ง A เป็นจำนวน 10 รอบ เป็นต้น หรือจำนวนครั้งที่ทำซ้ำนั้นขึ้นอยู่กับเงื่อนไขบางอย่าง เช่น ตราบใดที่ค่าของตัวแปร x ยังมากกว่าหรือเท่ากับ 0 อยู่ ให้ทำตามชุดคำสั่ง A เป็นต้น

ในโปรแกรม Scilab มีคำสั่งสำหรับการกำหนดทิศทางของโปรแกรมทั้งสองประเภท ดังนี้

4.1 คำสั่งในการตรวจสอบเงื่อนไข

เราสามารถกำหนดทิศทางของโปรแกรมได้ตามเงื่อนไขที่กำหนด โดยใช้คำสั่ง `if` ซึ่งมีโครงสร้างดังต่อไปนี้

```
if <เงื่อนไขที่ 1 ที่ต้องการตรวจสอบ> then
        ⋮
        (ชุดคำสั่งที่ 1)
        ⋮
elseif <เงื่อนไขที่ 2 ที่ต้องการตรวจสอบ> then
        ⋮
        (ชุดคำสั่งที่ 2)
        ⋮
[เพิ่ม elseif ไปเรื่อย ๆ ถ้ามีเงื่อนไขอื่น ๆ ที่ต้องการตรวจสอบ]
        ⋮
elseif <เงื่อนไขที่ n ที่ต้องการตรวจสอบ> then
        ⋮
        (ชุดคำสั่งที่ n)
        ⋮
else
        ⋮
        ชุดคำสั่งที่ n + 1
        ⋮
end
```



ตัวอย่าง 4.1.1 พิมพ์ชุดคำสั่งต่อไปนี้ลงใน SciNotes แล้วบันทึกเป็นไฟล์ชื่อ `testif.sce`

```
// รับชื่อบุคคลจาก keyboard
name = input("Please enter your name: ", "string")
// รับอายุบุคคลจาก keyboard
age = input("Please enter your age: ")
if age < 0 then
    status = "impossible";
elseif age < 20 then
    status = "too young";
elseif age < 30 then
    status = "so young";
elseif age < 40 then
    status = "quite all right";
else
    status = "too old";
end
printf("Hello %s. You are now %d years old, which is %s.\n", name, age,
status)
```

ให้นักศึกษาทดลอง run โปรแกรมนี้ แล้วสังเกตผลที่ได้เมื่อกรอกข้อมูลที่แตกต่างกัน

<u>หมายเหตุ</u> การระบุเงื่อนไขที่ต้องการตรวจสอบในคำสั่ง `if` นั้น จะต้องให้อยู่ในรูปใดรูปหนึ่งต่อไปนี้

- ค่าตรรกะ ได้แก่ `%T` (จริง) หรือ `%F` (เท็จ)
- การดำเนินการระหว่างค่าตรรกะ เช่น `p & ~q` เมื่อ `p`, `q` เป็นตัวแปรที่เก็บค่าตรรกะ เป็นต้น
- ความสัมพันธ์ เช่น `4 >= 3` เป็นต้น

4.2 คำสั่งในการทำซ้ำ

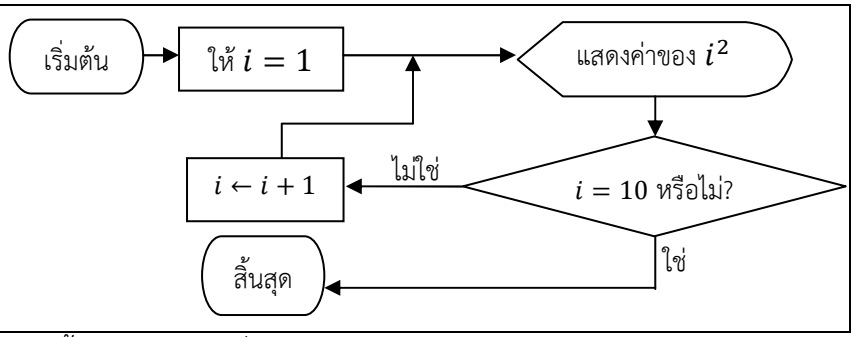
<u>4.2.1 คำสั่งในการทำซ้ำตามจำนวนรอบที่แน่นอน</u>

เราสามารถทำกระบวนงานหนึ่ง ๆ ซ้ำกันหลาย ๆ ครั้งเป็นจำนวนรอบที่แน่นอนได้ โดยใช้คำสั่ง `for` ซึ่งมีโครงสร้างดังนี้

```
for <ตัวแปร>=<ลำดับของค่าที่ต้องการ>
      ⋮
      ชุดคำสั่งที่ต้องการทำซ้ำ ซึ่งอาจอาศัยค่าของตัวแปรข้างต้นหรือไม่ก็ได้
      ⋮
end
```

ตัวอย่าง 4.2.1 พิมพ์ชุดคำสั่งต่อไปนี้ลงใน SciNotes แล้วบันทึกเป็นไฟล์ชื่อ `testfor.sce`

```
for i = 1:10
    disp(i^2)
end
```



ให้นักศึกษาทดลอง run โปรแกรมนี้ แล้วสังเกตผลที่ได้

วิธีการกำหนดลำดับของค่าสำหรับการทำซ้ำ

วิธีที่	สัญลักษณ์	ความหมายทางคณิตศาสตร์
1	$m:d:n$ (เมื่อ $m \le n$ และ $d > 0$)	$m, m+d, m+2d, \dots, m+kd$ เมื่อ k เป็นจำนวนเต็มที่มากที่สุดที่ทำให้ $m+kd \le n$
2	$m:n$ (เมื่อ $m \le n$)	เช่นเดียวกับวิธีที่ 1 แต่ให้ $d = 1$
3	$m:-d:n$ (เมื่อ $m \ge n$ และ $d > 0$)	$m, m-d, m-2d, \dots, m-kd$ เมื่อ k เป็นจำนวนเต็มที่มากที่สุดที่ทำให้ $m-kd \ge n$
4	$[m_1\ m_2 \dots\ m_n]$	$m_1, m_2, \dots, m_n$

ตัวอย่าง 4.2.2 จงแก้ไขไฟล์ `testfor.sce` ให้แสดงค่าต่อไปนี้ทีละบรรทัด

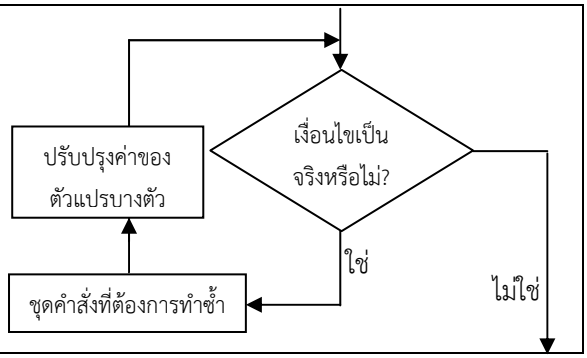
$$19^2, 17^2, 15^2, \dots, 3^2, 1^2$$

วิธีทำ

4.2.2 คำสั่งในการทำซ้ำอย่างมีเงื่อนไข

ในกรณีที่ไม่ทราบจำนวนรอบในการทำซ้ำเป็นที่แน่นอน เราอาจจะสามารถทำกระบวนงานหนึ่ง ๆ ซ้ำกันหลาย ๆ ครั้งได้ ในลักษณะที่ว่าการทำซ้ำนั้นจะยังคงมีอยู่ตราบใดที่เงื่อนไขบางอย่างยังคงเป็นจริง ซึ่งเราสามารถทำได้โดยใช้คำสั่ง `while` ซึ่งมีโครงสร้างดังนี้

```
while <เงื่อนไขที่ต้องการตรวจสอบ>
        ⋮
        (ชุดคำสั่งที่ต้องการทำซ้ำ ตราบใดที่เงื่อนไขข้างต้นยังคงเป็นจริง)
        ⋮
        (มีการปรับปรุงค่าของตัวแปรบางตัว)
end
```



ตัวอย่าง 4.2.3 พิมพ์ชุดคำสั่งต่อไปนี้ลงใน SciNotes แล้วบันทึกเป็นไฟล์ชื่อ `testwhile.sce`

```
// รับค่าของตัวแปร x ทาง keyboard
x = input("Please enter a positive integer: ");
// แสดงข้อความส่วนหัวของตาราง
printf("-----------------------\n")
printf("x\t\ln(x)\n")
printf("-----------------------\n")
// ตราบใดที่ x ยังคงมีค่าเป็นบวก ให้แสดงค่า x และ ln x ทีละบรรทัด
while x>0
    printf("%d\t%.4f\n", x, log(x)) // คำสั่ง log(x) หมายถึงลอการิทึมฐาน e
    x = x-2;                        // ปรับปรุงค่าของตัวแปร x
end
```

ให้นักศึกษาทดลอง run โปรแกรมนี้ แล้วสังเกตผลที่ได้ เมื่อทดลองกับค่าของ x ที่แตกต่างกัน

หมายเหตุ

1. การระบุเงื่อนไขในคำสั่ง `while` มีลักษณะเช่นเดียวกับการระบุเงื่อนไขในคำสั่ง `if`
2. ในการใช้คำสั่ง `while` นั้น จะต้องมีการปรับปรุงค่าของตัวแปรที่เกี่ยวข้องกับเงื่อนไขที่ระบุไว้เสมอ เพราะมิฉะนั้นแล้วอาจทำให้เงื่อนไขไม่มีทางเป็นเท็จ ซึ่งจะทำให้โปรแกรมติดอยู่ในการทำซ้ำตลอดไป

4.2.3 คำสั่งอื่น ๆ ที่ใช้ในการควบคุมการทำซ้ำ

คำสั่ง `break`

คำสั่ง `break` เป็นคำสั่งที่ใช้ได้ทั้งในคำสั่ง `for` และคำสั่ง `while` ใช้ในกรณีที่เราต้องการออกจากการทำซ้ำก่อนกำหนด กล่าวคือ

- ในกรณีการทำซ้ำโดยใช้คำสั่ง `for` : คำสั่ง `break` จะบังคับให้ออกจากการทำซ้ำทันที โดยไม่ต้องรอให้ตัวแปรที่กำหนดไว้ในคำสั่ง `for` นั้นมีค่าเป็นตัวสุดท้ายของลำดับที่กำหนด
- ในกรณีการทำซ้ำโดยใช้คำสั่ง `while` : คำสั่ง `break` จะบังคับให้ออกจากการทำซ้ำทันที โดยไม่ต้องรอให้เงื่อนไขที่ระบุไว้ในคำสั่ง `while` นั้นเป็นเท็จ

ตัวอย่าง 4.2.4 พิมพ์ชุดคำสั่งต่อไปนี้ลงใน SciNotes แล้วบันทึกเป็นไฟล์ชื่อ `testbreak.sce`

```
for i=1:10
    disp(i^2)
    if i==5 then
        break;
    end
end

i = 20;
while i>0
    disp(sqrt(i));
    if i<15 then
        break;
    end
    i = i-2;
end
```

ให้นักศึกษาทดลอง run โปรแกรมนี้ แล้วสังเกตผลที่ได้

คำสั่ง `continue`

คำสั่ง `continue` เป็นคำสั่งที่ใช้ได้ทั้งในคำสั่ง `for` และคำสั่ง `while` ใช้ในกรณีที่เราต้องการวนกลับไปที่ตอนต้นของชุดคำสั่งที่เราต้องการให้ทำซ้ำใหม่ทันที อย่างไรก็ตาม การเรียกใช้คำสั่งนี้ภายในการทำซ้ำด้วยคำสั่ง `for` และ `while` จะได้ผลลัพธ์ที่แตกต่างกัน ดังนี้

- ในกรณีการทำซ้ำโดยใช้คำสั่ง `for` : คำสั่ง `continue` จะทำให้เราวนกลับไปที่ตอนต้นของชุดคำสั่งที่เราต้องการให้ทำซ้ำ **โดยจะปรับค่าของตัวแปรให้เป็นค่าถัดไปจากค่าปัจจุบัน** (ขึ้นอยู่กับการกำหนดลำดับของค่าสำหรับตัวแปรดังกล่าว)
- ในกรณีการทำซ้ำโดยใช้คำสั่ง `while` : คำสั่ง `continue` จะทำให้เราวนกลับไปที่ตอนต้นของชุดคำสั่งที่เราต้องการให้ทำซ้ำ **โดยไม่ปรับปรุงค่าของตัวแปร** ดังนั้น การใช้งานคำสั่ง `continue` ในกรณีนี้จึงต้องใช้ความระมัดระวังเป็นอย่างมาก เพราะอาจทำให้เงื่อนไขไม่มีทางเป็นเท็จ ซึ่งจะทำให้โปรแกรมติดอยู่ในการทำซ้ำตลอดไปได้

ตัวอย่าง 4.2.5 พิมพ์ชุดคำสั่งต่อไปนี้ลงใน SciNotes แล้วบันทึกเป็นไฟล์ชื่อ `testcont.sce`

```
for i=1:2:15
    if i<7 then
        continue;
    else
        disp(i^2);
    end
end

j = 1;
while j <= 10
    disp(j);
    if j == 5 then
        continue;
    else
        j = j + 1;
    end
end
```

ให้นักศึกษาทดลอง run โปรแกรมนี้ แล้วสังเกตผลที่ได้