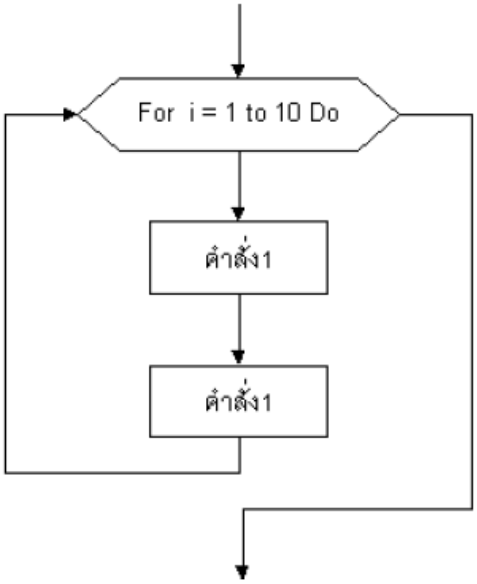
บทที่ 4 คำสั่งควบคุม (Control Statement) (ต่อ)

4.3 คำสั่ง for

คำสั่งวนซ้ำเป็นคำสั่งที่ใช้แก้ปัญหาโจทย์ในลักษณะที่มีการทำงานซ้ำกันหลาย ๆ ครั้ง ซึ่งเขียนในรูปแบบของผังงานได้ดังรูปด้านล่าง



แสดงผังงานของคำสั่ง for

คำสั่ง for เป็นคำ สั่งวนซ้ำในลักษณะที่รู้จำนวนรอบของการวนซ้ำที่แน่นอน โดยแบ่งรูปแบบหลักออกเป็น 3 ส่วน ได้แก่

- ส่วนที่ใช้กำหนดค่าเริ่มต้นหรือกำหนดค่าตัวนับของการวนซ้ำ
- ส่วนที่ตรวจเงื่อนไขการวนซ้ำ
- ส่วนของการจัดการค่าตัวนับของการวนซ้ำ

```
for ( กำหนดค่าตัวนับ ; เงื่อนไขการวนซ้ำ ; จัดการค่าตัวนับ ) {
        คำสั่ง1;
        คำสั่ง2;
}
```

ขั้นตอนของการทำงานเมื่อพบคำ สั่ง for มีดังนี้

1. ทำคำสั่งในการกำหนดค่าตัวนับ
2. ตรวจสอบเงื่อนไขการวนซ้ำ หากเป็นเท็จจะหยุดและออกจากการทำงานของคำสั่ง for ไปทำงานคำสั่งหลังจากนั้น
3. กรณีเงื่อนไขการวนซ้ำเป็นจริงจะทำคำสั่งในขอบเขตของ for นั้น คือภายใต้เครื่อง { } จนกระทั่งหมด และไปทำคำสั่งจัดการค่าตัวนับ ซึ่งอาจจะเป็นการเพิ่มค่าหรือลดค่าตัวนับ หลังจากนั้นจะกลับทำ ตรวจสอบเงื่อนไขการวนซ้ำในขั้นตอนที่ 2 ทำ เช่นนี้เรื่อยไปจนกระทั่งเงื่อนไขการวนซ้ำเป็นเท็จ

หากคำสั่งที่ต้องทำ ในการวนซ้ำ มีเพียง 1 คำสั่ง รูปแบบการเขียนจะเขียนเครื่องหมาย { } ครอบคำ สั่งนั้นไว้หรือไม่ก็ได้ แต่ถ้ามีคำสั่งที่ต้องทำซ้ำมากกว่า 1 คำ สั่ง จะต้องมีเครื่องหมาย { } แสดงขอบเขตของการทำวนซ้ำเสมอ พิจารณาตัวอย่างที่ 4.9

```
#include <stdio.h>
void main( ) {
    int i, number;
    float average, sum=0.0f;
    for ( i=0 ; i < 5 ; i++) {
        printf("Enter number %d : ", i+1);
        scanf("%d", number);
        sum += number;
    }
    average = sum / 5;
    printf("Average is %.2f", average);
}
```

```
ผลการทำงานของโปรแกรม
   Enter number 1 : 10
   Enter number 2 : 20
   Enter number 3 : 30
   Enter number 4 : 40
   Enter number 5 : 50
   Average is 30.00
```

ในการทำงานของคำ สั่ง for สิ่งสำคัญที่ผู้ใช้จะต้องรู้คือ ทำอย่างไรจึงจะวนทำซ้ำได้เท่ากับจำนวนรอบที่ต้องการ จากตัวอย่างที่ 4.9 มีการใช้ตัวแปร i เป็น ตัวนับจำนวนรอบของการทำซ้ำ ในที่นี้ต้องการให้มีการทำซ้ำทั้งหมด 5 รอบ ไล่ขั้นตอนการทำงานของโปรแกรมได้ดังนี้

1. กำหนดให้ i มีค่าเริ่มต้นที่ 0
2. แล้วตรวจสอบว่า i มีค่าน้อยกว่า 5 เงื่อนไขเป็นจริง มีรับข้อมูลตัวแรกและทำการบวกเลขที่รับเข้ามานั้นเก็บไว้ในตัวแปร sum

3. เพิ่มค่า i ขึ้น 1 เพราะฉะนั้น i มีค่าเท่ากับ 1
4. แล้วตรวจสอบว่า i มีค่าน้อยกว่า 5 เงื่อนไขเป็นจริง มีรับข้อมูลตัวที่ 2 และทำการบวกเลขที่รับเข้ามานั้นเก็บไว้ในตัวแปร sum
5. เพิ่มค่า i ขึ้น 1 เพราะฉะนั้น i มีค่าเท่ากับ 2
6. แล้วตรวจสอบว่า i มีค่าน้อยกว่า 5 เงื่อนไขเป็นจริง มีรับข้อมูลตัวที่ 3 และทำการบวกเลขที่รับเข้ามานั้นเก็บไว้ในตัวแปร sum
7. เพิ่มค่า i ขึ้น 1 เพราะฉะนั้น i มีค่าเท่ากับ 3
8. แล้วตรวจสอบว่า i มีค่าน้อยกว่า 5 เงื่อนไขเป็นจริง มีรับข้อมูลตัวที่ 4 และทำการบวกเลขที่รับเข้ามานั้นเก็บไว้ในตัวแปร sum
9. เพิ่มค่า i ขึ้น 1 เพราะฉะนั้น i มีค่าเท่ากับ 4
10. แล้วตรวจสอบว่า i มีค่า น้อยกว่า 5 เงื่อนไขเป็นจริง มีรับข้อมูลตัวที่ 5 และทำการบวกเลขที่รับเข้ามานั้นเก็บไว้ในตัวแปร sum
11. เพิ่มค่า i ขึ้น 1 เพราะฉะนั้น i มีค่าเท่ากับ 5
12. ตรวจสอบเงื่อนไข i มีค่าน้อยกว่า 5 พบว่าเงื่อนไขเป็นเท็จ ก็จะจบการทำงานภายในคำ สั่ง for

พิจารณาค่าของการวนซ้ำต่อไปนี้ ว่าค่าเริ่มต้นของตัวนับการทำซ้ำเป็นเท่าใด ทำซ้ำทั้งหมดกี่รอบและค่าสุดท้ายของตัวนับที่ออกจากการทำซ้ำเป็นเท่าใด จากตารางด้านล่าง

ตัวอย่างหาค่าของการทำงานวนซ้ำด้วยคำสั่ง for

คำสั่ง	ค่าตัวนับเริ่มต้น	จำนวนรอบ	ค่าตัวนับสุดท้าย
for (i = 1; i <= 5; i++)	1	5	6
for (j = 53; j <= 57; j++)	53	5	58
for (k = 10; k > 5; k- -)	10	5	5
for (a = 2; a < 20; a += 2)	2	9	20
for (a = 2; a < 20; a *= 2)	2	4	32

จะเห็นว่าคำสั่งส่วนต่าง ๆ ของคำสั่ง for สามารถเขียนได้หลายรูปแบบ แต่มีสุดท้ายที่ต้องการคือการควบคุมจำนวนรอบของการทำซ้ำให้ได้เท่ากับจำนวนที่ผู้เขียนโปรแกรมต้องการ

สิ่งที่ต้องระวังในการเขียนก็คือ

```
for (i = 0; i < 5; i++);
        printf("Hello\n");
```

การใส่เครื่องหมาย ; ต่อท้ายคำสั่ง for เครื่องหมาย ; เป็นคำ สั่งที่เรียกว่า Null Statement นับเป็นคำสั่ง 1 คำสั่ง โดยที่คำสั่งนี้จะไม่ทำอะไรเลย เพราะฉะนั้นในตัวอย่างดังกล่าว จะมีการวนทำซ้ำโดยไม่ทำอะไรเลย 5 รอบ แล้วจึงพิมพ์ข้อความ Hello เพียงข้อความเดียว แทนที่จะพิมพ์ข้อความ Hello 5 ครั้ง

นอกจากนี้ที่พบเห็นบ่อย ๆ คือการใช้เครื่องหมาย ; ต่อท้ายคำ สั่ง if ซึ่งจะให้ผลเช่นเดียวกับตัวอย่างข้างต้น เช่น

```
if (a > 5);
        printf("Hello");
```

จากการใส่เครื่องหมายดังกล่าวคอมไพเลอร์จะตีความได้ว่า ตรวจสอบว่า a มีค่า มากกว่า 5 หรือไม่ ถ้ามากกว่าก็ไม่ต้องทำอะไร (ทำ Null Statement คือ ;) แล้ว ไปพิมพ์ข้อความว่า Hello แทนที่จะทำการตรวจสอบว่าถ้า a มีค่ามากกว่า 5 ให้พิมพ์ข้อความว่า Hello

ตัวอย่างการใช้คำสั่ง for เพิ่มเติม คือคำสั่ง for สามารถใช้ตัวดำเนินการคอมมา (,) ประกอบภายในคำสั่ง หรืออาจจะเขียนองค์ประกอบของคำสั่งไม่ครบก็ได้ แสดงดังตัวอย่าง

```
for (i=0, j=0; i<5 && i+j < 10; i++, j+=2) {
    ......
}
a = 0;
for ( ; a < 5 ; ) {
    ......
   a++;
}
```

ตัวอย่างที่ 4.10 โปรแกรมเพื่อหาค่าเฉลี่ยของเลขจำนวนเต็ม N ตัวซึ่งรับจากผู้ใช้

```
#include <stdio.h>
void main( ) {
  int i, number, n;
  float average, sum=0.0f;
  printf("Enter N : ");
  scanf("%d", &n);
  for ( i=0 ; i < n ; i++) {
    printf("Enter number %d : ", i+1);
    scanf("%d", number);
    sum += number;
  }
  average = sum / 5;
  printf("Average is %.2f", average);
}
```

ตัวอย่างที่ 4.11 โปรแกรมเพื่อแสดงผลลัพธ์ของการแปลงเลขฐาน 10 ให้เป็นเลขฐาน 8 และเลขฐาน 16 โดยรับข้อมูลค่าเริ่มต้นและค่าสุดท้ายของช่วงข้อมูลที่ต้องการแปลงค่า

```
#include <stdio.h>
void main( ) {
   int start, end, i;
   printf("Enter start number : ");
   scanf("%d", &start);
   printf("Enter end number : ");
   scanf("%d", &end);
   printf("\n\tDec\t\t\tOctal\t\t\tHexa");
   for ( i=start ; i <= end ; i++) {
      printf("\n\t%3d\t\t\t%5o\t\t\t%4x", i, i, i);
   }
}
```

นอกจากนี้ยังมีคำ สั่งเพิ่มเติมที่จะพบได้เมื่อมีการใช้การทำ งานวนซ้ำในลักษณะต่างทั้งคำ สั่ง for คำ สั่ง while และคำ สั่ง do-while คำสั่งดังกล่าวได้แก่ **คำสั่ง continue** คำ สั่งนี้จะทำ งานคล้ายกับคำ สั่ง break ซึ่งเป็น การบอกให จบการทำงานในขอบเขตที่คำสั่งอยู่ และออกจากคำสั่งวนซ้ำนั้น คำสั่ง continue จะบอกให้จบการทำ งานในขอบเขตที่คำสั่งอยู่เช่นเดียวกัน แต่จะวนกลับไปยังทำคำสั่งวนซ้ำนั้นในรอบถัดไปแทน พิจารณาจากตัวอย่าง โดยที่ให้มีการรับข้อมูลเลขจำนวนเต็ม N จำนวนจากผู้ใช้ ให้หาผลรวมของเลขคู่ที่รับเข้ามา แต่หากผลรวมนั้นมาค่ามากกว่า 200 ให้หยุดการหาผลรวม ถ้าเมื่อใดป้อนเลข 0 เข้ามาให้หยุดการรับข้อมูลทันทีจะได้ว่า

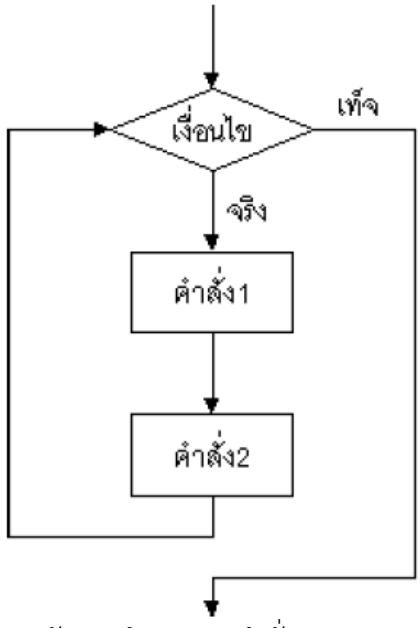
```
printf("Enter N : ");
scanf("%d", &n);
sum = 0;
for (i=0; i < N; i++) {
   printf("Enter number %d : ", i);
   scanf("%d", &number);
   if (number == 0)
      break;
   else if (number % 2 == 0) {
   if (sum > 200)
      continue;
      sum += number;
   }
}
```

เมื่อใดที่ค่า sum มีค่ามากกว่า 200 จะมีการทำคำสั่ง continue ซึ่งมีผลทำให้มีมีการประมวลผลคำสั่งที่เหลือ แต่จะกลับไปทำคำสั่ง i++ และตรวจสอบเงื่อนไขของ i < N ถ้าเงื่อนไขเป็นจริง ก็จะวนทำงานซ้ำ ต่อไป

4.4 คำสั่ง while

คำ สั่ง while เป็นคำ สั่งวนซ้ำมักใช้ในกรณีที่ต้องทำงานซ้ำกันหลาย ๆ ครั้ง โดยไม่ทราบจำนวนรอบของการทำซ้ำที่แน่นอน ตัวอย่างเช่น ต้องการรับข้อมูลเลขจำนวนเต็มบวกจากผู้ใช้จำนวนหนึ่งเพื่อนำ มาหาค่าเฉลี่ยของตัวเลขที่ป้อนเข้ามาทั้งหมด

การทำงานดังกล่าวจะต้องมีการทำงานวนซ้ำเพื่อรับข้อมูลและหาผลรวมของข้อมูลที่รับเข้ามานั้น ในกรณีเช่นนี้ผู้ที่ใช้งานโปรแกรมซึ่งป้อนจำนวนข้อมูลเข้าสู่ระบบอาจจะป้อนข้อมูลในจำนวนที่ไม่เท่ากัน หากต้องการเขียนโปรแกรมเพื่อให้ทำงานกับผู้ใช้คนใด ๆ มักจะใช้คำ สั่ง while เข้ามาช่วยในการเขียนโปรแกรม ในกรณีตัวอย่างเราทราบว่าข้อมูลที่รับเข้ามาต้องเป็นข้อมูลจำนวนเต็มบวกเท่านั้นจึงจะนำ มาหาค่าเฉลี่ย ซึ่งผู้เขียนโปรแกรมสามารถตั้งเงื่อนไขว่า หากมีการป้อนข้อมูลเป็นเลขจำนวนเต็มลบให้แสดงว่าผู้ใช้ต้องการหยุดการป้อนข้อมูลนั้น การทำงานของคำ สั่ง while สามารถเขียนแสดงด้วยผังงานดังรูปด้านล่าง



ผังการทำงานของคำสั่ง while

หากแทนผังงานดังกล่าวด้วยคำ สั่ง while สามารถเขียนรูปแบบของคำ สั่ง while ได้ดังนี้

```
while ( เงื่อนไข ) {
    คำสั่ง1 ;
    คำสั่ง2 ;
}
```

จากรูปแบบคำสั่งดังกล่าวจะเกิดการทำงานคำสั่ง 1 และคำสั่ง 2 เมื่อมีการตรวจสอบว่าเงื่อนไขเป็นจริง และจะทำงานซ้ำเช่นนี้ไปจนกว่าเงื่อนไขนั้นจะเป็นเท็จ หากคำสั่งที่ทำซ้ำมีมากกว่า 1 คำ สั่งจะต้องใช้เครื่องหมายแสดงขอบเขต คือ { } ครอบคำสั่งที่ต้องการให้ทำซ้ำทั้งหมด แต่ถ้ามีคำสั่งที่ต้องทำซ้ำเพียงคำสั่งเดียว ผู้เขียนโปรแกรมใส่เครื่องหมาย { } หรือไม่ก็ได้

ทั้งนี้ต้องระวังห้ามใส่เครื่องหมาย ; หลังวงเล็บของเงื่อนไข ซึ่งถือเป็นคำสั่ง Null Statement ดังอธิบายในหัวข้อ 4.3 แสดงตัวอย่างโปรแกรมด้วยตัวอย่างที่ 4.12

ตัวอย่างที่ 4.12 โปรแกรมเพื่อหาค่าเฉลี่ยของเลขจำนวนเต็มบวกที่ผู้ใช้ป้อนเข้าสู่ระบบ เมื่อใดที่ผู้ใช้ป้อนเลขจำนวนเต็มลบให้ถือว่าสิ้นสุดการป้อนข้อมูล

```
#include <stdio.h>
void main( ) {
   int number, count;
   float sum, average;
   count = 0;
   sum = 0.0f;
   printf("Enter number : ");
   scanf("%d", &number);
   while ( number >= 0) {
      sum += number;
      count++;
      printf("Enter number : ");
      scanf("%d", &number);
   }
   average = sum / count;
   printf("Average is %.2f", average);
}
```

จากตัวอย่างเนื่องจากจะต้องหาค่าเฉลี่ยของเลขจำนวนเต็มบวกที่ป้อนเข้ามาจึงใช้ตัวแปร count ในการนับจำนวนตัวเลขจำนวนเต็มบวกที่ป้อนเข้ามาทั้งหมด และมีเงื่อนไขในการทำซ้ำคือ number >= 0 ถ้าตราบใดที่ผู้ใช้ป้อนข้อมูลเข้ามามีค่ามากกว่าหรือเท่ากับ 0 ก็จะเกิดการหาค่าผลรวมของข้อมูลนั้น และนับจำนวนข้อมูลที่รับเข้ามา ตลอดจนรับข้อมูลตัวใหม่เข้าตรวจสอบว่าตรงกับเงื่อนไขของการทำซ้ำหรือไม่ จนกว่าจะมีการป้อนข้อมูลน้อยกว่า 0 จึงจะมีการคำนวณหาค่าเฉลี่ย และแสดงผลลัพธ์ทางจอภาพ หากสังเกตรูปแบบการเขียนตัวอย่างดังกล่าว จะเขียนเป็นโครงร่างได้ดังนี้

```
printf("Enter number : ");
scanf("%d", &number);          } กำหนดค่าเริ่มต้นให้ค่าควบคุม
while (number >= 0) {          → เงื่อนไขค่าควบคุมการทำซ้ำ
      ....
      printf("Enter number : ");
      scanf("%d", &number);    } เปลี่ยนแปลงค่าควบคุม
}
```

ในการทำงานของคำ สั่ง while จะมีตัวควบคุมการทำซ้ำเสมอ ซึ่งจะต้องมีการกำหนดค่าเริ่มต้นของค่าควบคุม ตรวจสอบเงื่อนไขค่าควบคุม และมีการเปลี่ยนแปลงค่าควบคุม เพื่อนำไปตรวจสอบเงื่อนไขการทำซ้ำ จากขั้นตอนทั้ง 3 พิจารณาเปรียบเทียบกับคำสั่ง for ดังตัวอย่าง

```
for (i = 0; i < 10; i++) {
   …..
}
```

ในคำสั่ง for ดังตัวอย่างมี i เป็นคำสั่งควบคุม มีเงื่อนไขตรวจสอบค่าควบคุม และมีการเปลี่ยนแปลงค่าควบคุมเพื่อนำ ไปสู่การตรวจสอบเงื่อนไขอีกครั้ง ซึ่งสามารถเขียนในรูปแบบของคำ สั่ง while ได้ว่า

```
i = 0;
while ( i < 10 ) {
   …
   i++;
}
```

จะเห็นว่าคำ สั่ง for นั้นสามารถเขียนในรูปแบบของคำสั่ง while ได้เช่นเดียวกัน ต่างกันที่คำสั่ง for มักจะใช้กับการทำงานที่รู้จำนวนรอบของการทำซ้ำที่แน่นอน ส่วนคำสั่ง while สามารถใช้ได้ทั่วไปแต่จะเหมาะกับงานที่ไม่รู้จำนวนรอบการทำซ้ำดังตัวอย่างข้างต้น ตัวอย่างเพิ่มเติมแสดงดังตัวอย่างที่ 4.13

ตัวอย่างที่ 4.13 รับข้อมูลจำนวนเต็มจากผู้ใช้ จนกว่าผู้ใช้จะป้อนเลข -9999 ให้หาว่ามีเลขจำนวนเต็มลบ (ไม่รวม –9999) อยู่กี่จำนวน และมีเลขจำนวนเต็มบวก (รวมทั้งเลขศูนย์) อยู่กี่จำนวน

```
#include <stdio.h>
void main( ) {
    int countPlus, countMinus, num;
    countPlus = countMinus = 0;
    printf("Enter number (-9999 for end) : ");

    scanf("%d", &num);
    while (num != -9999) {
            if (num < 0)
                 countMinus++;
            else
                 countPlus++;
            printf("Enter number (-9999 for end) : ");
           scanf("%d", &num);
     }
    printf("Number less than zero = %d, more than or equal to zero = %d",
                                                        countMinus, countPlus);
}
```

ในกรณีของการใช้คำสั่งวนซ้ำต่าง ๆ สิ่งที่ต้องระวังคือการนำ เอาเลขจำนวนจริงมาเป็นค่าควบคุมการวนซ้ำ พิจารณาจากตัวอย่าง เนื่องจากเรารู้ว่า 1/3 + 1/3 + 1/3 มีค่าเท่ากับ 1 เราจึงจะใช้วิธีการดังกล่าวมาเป็นตัวควบคุมการทำงานวนซ้ำ เช่น

```
float x, y;
x = 1.0f;
y = 1/3;
while (x != y) {
   printf("\n55555");
   y += 1/3;
}
```

หากทดลองเรียกใช้โปรแกรมดังกล่าว จะเกิดการพิมพ์ค่า 55555 ไปเรื่อย ๆ ทำ ให้เกิดสิ่งที่เรียกว่าการวนซ้ำแบบอนันต์ หรือการวนซ้ำแบบไม่รู้จบ ทั้งนี้เกิดจากการหาของภาษาซี 1 และ 3 เป็นจำนวนเต็ม เมื่อนำ มาหารจะได้ค่าเท่ากับ 0 สามารถทดลองง่าย ๆ ด้วยคำสั่งเงื่อนไข

```
x = 1.0f;
y = (1/3) + (1/3) + (1/3);
if (x == y)
   printf("Equal");
else
   printf("Not equal");
```

คำตอบที่ได้คือ Not equal เพราะฉะนั้นหากใช้เลขจำนวนจริงและการคำนวณเป็นเป็นสิ่งควบคุม การวนซ้ำหรือนำ ไปเปรียบเทียบเงื่อนไขจะต้องระมัดระวังในการเขียนโปรแกรม แสดงตัวอย่างเพิ่มเติมในตัวอย่างที่ 4.14 และ 4.15

ตัวอย่างที่ 4.14 เขียนโปรแกรมเพื่อรับค่าเลขจำนวนจริงจากผู้ใช้ 1 จำนวน และให้ลบออกด้วย 0.16523 อยากทราบว่าจะต้องลบออกกี่ครั้ง เลขที่รับเข้ามานั้นจึงจะมีค่าเข้าใกล้ 0 มากที่สุด (เข้าใกล้ทางบวก) และค่าที่เข้าใกล้ 0 นั้นมีค่าเท่าใด

```
#include <stdio.h>
#define MINUS_VALUE 0.16523
void main( ) {
   int count=0;
   float num, ;
   printf("Enter number : ");
   scanf("%f", &num);
   while (num > 0) {
      num -= MINUS_VALUE;
      count++;
   }
   printf("It's need %d times, value is %f", count-1, num+MINUS_VALUE);
}
```

จากตัวอย่างมีการใช้คำสั่ง #define เป็นการบอกให้รู้ว่าหากเมื่อใดเจอข้อความที่อยู่ตรงกลาง จะกำหนดให้มีค่าเท่ากับค่าที่อยู่ทางขวาสุด เช่น #define MINUS_VALUE 0.16523 แปลได้ว่าเมื่อใดที่คำสั่งในโปรแกรมมีการอ้างถึงคำว่า MINUS_VALUE ให้แทนที่ข้อความนั้นด้วย 0.16523 เพราะฉะนั้นเมื่อพบคำสั่ง

num -= MINUS_VALUE;

จะมีค่าเท่ากับคำ สั่ง

num -= 0.16523;

ในตัวอย่างที่ 4.14 เมื่อต้องการพิมพ์ทราบคำตอบจะต้องมีการเพิ่มค่า num และลดค่า count เนื่องจากในมีการทำวนซ้ำ ตรวจสอบไว้ว่า num > 0 ตราบใดที่ค่า num มีค่ามากกว่า 0 จะทำ งานไปเรื่อย ๆ และจะจบก็ต่อมีค่า num มีค่ามากกว่าหรือเท่ากับ 0 แต่คำตอบต้องการค่า num ที่เข้าใกล้ 0 ทางบวก เพราะฉะนั้นจึงต้องมีการเพิ่มค่าให้ num เป็นสถานะก่อนสุดท้าย คือ บวกด้วย MINUS_VALUE และลดค่า count ลง 1

ตัวอย่างที่ 4.15 โปรแกรมเพื่อรับคะแนนสอบของนักเรียนให้อยู่ในช่วง 10 ถึง 20

```
#include <stdio.h>
#define YES 1
#define NO 0
void main( ) {
    int num, correct=NO;
    printf("Enter number : ");
    scanf("%d", &num);
    while (!correct) {
        if (num >= 10 && num <= 20)
            correct = YES;
        else {
            printf("Error data !!!!\n");
            printf("Enter number : ");
            scanf("%d", &num);
        }
    }
}
```

ในตัวอย่างที่ 4.15 สามารถนำ ไปประยุกต์ใช้กับการตรวจสอบค่าของข้อมูลที่รับเข้ามาให้อยู่ในช่วงที่กำหนดไว้ ในที่นี้ใช้ตัวแปร correct เป็นตัวควบคุมการวนซ้ำโดยกำหนดเริ่มต้นให้เท่ากับ NO คือ 0 เมื่อตรวจสอบเงื่อนไข **!correct** จะได้ค่า 1 ทำ ให้การวนซ้ำเป็นจริง จะเข้าไปทำ คำ สั่งใน while หากตรวจสอบได้ว่า

```
printf("Enter number : ");
scanf("%d", &num);
while (num < 10 || num > 20) {
        printf("Error data !!!!\n");
        printf("Enter number : ");
        scanf("%d", &num);
}
```

ในการเขียนโปรแกรมคำสั่งอาจจะประกอบขึ้นมาจากคำสั่งต่าง ๆ หลายคำสั่งทำ งานด้วยกัน ทั้งคำสั่งทั่วไป คำสั่งเงื่อนไข และคำสั่งวนซ้ำ พิจารณาตัวอย่างเพิ่มเติมของการใช้คำสั่งร่วมกันหลายรูปแบบ จากตัวอย่างที่ 4.16

ตัวอย่างที่ 4.16 โปรแกรมเพื่อรับข้อมูลจากผู้ใช้ N คน โดยที่ผู้ใช้แต่ละคนป้อนเลขจำนวนเต็มเข้าสู่ระบบได้ตามความพอใจจนกว่าจะป้อนเลข 9999 ให้หาค่าเฉลี่ยของเลขที่ผู้ใช้แต่ละคนป้อน และหาค่าเฉลี่ยของเลขที่ป้อนเข้าสู่ระบบทั้งหมด และหาว่าผู้ใช้แต่ละคนป้อนเลขจำนวนเต็มที่อยู่ในช่วง 0 ถึง 30 คนละกี่ตัวและหาว่ามีการป้อนเลขเลขจำนวนเต็มลบที่อยู่ในช่วง –1 ถึง –30 กี่ตัวโดยผู้ใช้ทั้งหมดกี่ตัว

```
#include <stdio.h>
#include <conio.h>
void main() {
   int n, num, countOne, countAll, countSpecOne, countSpecAll, i;
   float sumOne, sumAll, averageOne, averageAll;
   clrscr();
   countAll = countSpecAll = 0;
   sumAll = 0.0f;
   printf("Enter N : ");
   scanf("%d", &n);
   for (i = 1; i <= n; i++) {
      countOne = countSpecOne = 0;
      sumOne = 0.0f;
      printf("\n\nFor user no. %d\n", i);
      printf("Enter number : ");
     scanf("%d", &num);
     while (num != 9999) {
         countOne++;
         sumOne = sumOne + num;
         if (num >= 0 && num <= 30)
            countSpecOne++;
        else if (num <= -1 && num >= -30)
            countSpecAll++;
        printf("Enter number : ");
       scanf("%d", &num);
     }
```

```
      averageOne = sumOne / countOne;
      printf("Average for user no. %d is %.2f", i, averageOne);
      printf("\nNumber of data between 0 and 30 is %d", countSpecOne);
      countAll = countAll + countOne;
      sumAll = sumAll + sumOne;
     }
   averageAll = sumAll / countAll;
   printf("\n\nAverage for all user is %.2f", averageAll);
   printf("\nNumber of data between -1 and -30 is %d", countSpecAll);
   getch();
}
```

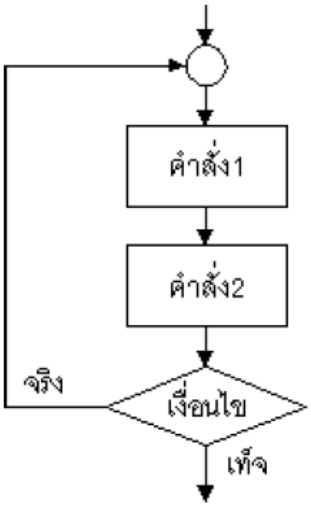
จากตัวอย่างสิ่งที่ต้องระวังคือการหาค่าตัวนับต่าง ๆ ว่าเป็นตัวนับสำหรับงานใด และสำหรับผู้ใช้แต่ละคนหรือว่าสำหรับข้อมูลทั้งหมด หากโปรแกรมซับซ้อนควรจะเริ่มจากการแก้ปัญหาย่อย ๆ ทีละข้อ โดยอาจจะแยกเป็นขั้นตอนคือ

1. รับข้อมูลค่า N คือ จำนวนผู้ใช้ก่อน สร้างการวนซ้ำสำหรับการทำ งานของผู้ใช้ N คน อาจจะใช้ คำสั่ง for หรือ คำ สั่ง while ก็ได้
2. พิจารณาการทำ งานของผู้ใช้แต่ละคน รับข้อมูลของผู้ใช้แต่ละคน ซึ่งต้องรับข้อมูลจนกว่าจะป้อน เลข 9999 จึงจะหยุดการทำ งาน ซึ่งสามารถใช้คำ สั่ง while มาช่วยในการทำ งาน
3. แก้ปัญหาที่โจทย์ต้องการทีละอย่าง โดยอาจจะเริ่มจากหาค่าเฉลี่ยของผู้ใช้แต่ละคน
4. หาค่าเฉลี่ยของผู้ใช้ทั้งหมด
5. หาค่าที่อยู่ในช่วง 0 ถึง 30 สำหรับผู้ใช้แต่ละคน
6. หาค่าที่อยู่ในช่วง –1 ถึง 30 สำหรับผู้ใช้ทั้งหมด

การแบ่งงานเป็นปัญหาย่อยแล้วแก้ปัญหาย่อยเหล่านั้น จะช่วยให้การเขียนโปรแกรมทำได้ง่ายขึ้น ซึ่งเป็นหลักการของการแบ่งแยกและเอาชนะ (Divide and Conquer) ซึ่งใช้ทั่วไปในงานเขียนโปรแกรม และสามารถเขียนเป็นงานย่อยซึ่งจะกล่าวถึงในเรื่องของฟังก์ชันต่อไป

4.5 คำสั่ง do-while

คำสั่ง do-while เป็นคำ สั่งวนซ้ำอีกรูปแบบหนึ่ง แสดงผังงานดังรูปด้านล่าง



แสดงผังงานของการทำงานคำสั่ง do-while

จากรูปจะเห็นว่าการทำงานของคำสั่ง do-while จะต้องมีการทำงานคำสั่ง1 และคำสั่ง 2 เสมอ หลังจากนั้น จะมีการตรวจสอบเงื่อนไข หากเงื่อนไขเป็นจริง ก็จะกลับไปทำ คำสั่งใน do-while อีก จนกว่าเงื่อนไขนั้นจะเป็นเท็จ เขียนในรูปแบบของคำสั่งได้ว่า

```
do {
    คำสั่ง1 ;
    คำสั่ง2 ;
} while ( เงื่อนไข ) ;
```

คำสั่ง do-while จะต่างกับคำสั่ง while ตรงที่ตะมีการทำงานคำสั่งก่อนตรวจสอบเงื่อนไข ในขณะที่คำสั่ง while จะมีการตรวจสอบเงื่อนไขก่อนการทำงานเสมอ พิจารณาจากตัวอย่างการรับข้อมูลให้อยู่ในช่วงที่ต้องการ สมมติให้อยู่ในช่วงตั้งแต่ 10 ถึง 20 ซึ่งเป็นที่แน่นอนว่าจะต้องมีการรับข้อมูลเข้ามาก่อน จากนั้นจะตรวจสอบว่าข้อมูลถูกต้องหรือไม่ หากเขียนในรูปแบบของคำ สั่ง do-while จะได้ว่า

```
do {
        printf("Enter number (between 10 and 20) : ");
        scanf("%d", &num);
} while (num < 10 || num > 20);
```

จากตัวอย่างจะมีการรับข้อมูล แล้วตรวจสอบว่าข้อมูลอยู่นอกช่วงที่กำหนด คือ น้อยกว่า 10 หรือมากกว่า 20 ถือว่าข้อมูลไม่ถูกต้องให้ไปรับข้อมูลใหม่จนกว่าจะถูกต้อง พิจารณาตัวอย่างที่ 4.17 และ 4.18

ตัวอย่างที่ 4.17 รับข้อมูลจำนวนเต็มจากผู้ใช้ และหาค่าเฉลี่ยของข้อมูลที่รับเข้ามา จนกว่าผู้ใช้จะป้อนเลข 9999

```
#include <stdio.h>
void main( ) {
    int num, count=0;
    float sum=0.0f, average;
    do {
        printf("Enter number : "):
        scanf("%d", &num);
        if (num != 9999) {
            count++;
            sum += num;
        }
    } while (num != 9999);
    average = sum / count;
    printf("Average is %.2f", average);
}
```

ตัวอย่างที่ 4.18 รับข้อมูลจำนวนเต็มจากผู้ใช้ จะหยุดรับข้อมูลเมื่อผลรวมของข้อมูลนั้นมีค่ามากกว่า200 ให้ว่ามีการรับข้อมูลทั้งหมดกี่จำนวน

```
#include <stdio.h>
void main( ) {
   int num, sum=0, count=0;
   do {
      printf("Enter number : "):
      scanf("%d", &num);
      sum += num;
      count++;
   } while (sum <= 200);
   printf("Enter number %d times", count);
}
```

คำ สั่ง do-while สามารถนำ ไปประยุกต์ใช้ได้เหมือนกับคำ สั่ง while หรือ for แต่มักจะใช้คำสั่งนี้ในการตรวจสอบความถูกต้องของข้อมูล หรือใช้ในการสั่งให้โปรแกรมทำงานวนซ้ำจนกว่าผู้ใช้ต้องการให้หยุดการทำงาน จาก

ตัวอย่างที่ได้แสดงมาทั้งหมดจะเห็นว่าเป็นการทำงานเพียงครั้งเดียวก็จบการทำงานของโปรแกรมหากต้องการทำงานนั้นอีกครั้ง ก็จะต้องเรียกใช้งานโปรแกรมนั้นใหม่ สามารถใช้คำสั่ง do-while มาช่วยได้ดังตัวอย่างที่ 4.19

ตัวอย่างที่ 4.19 รับข้อมูลจำนวนเต็มจากผู้ใช้ จะหยุดรับข้อมูลเมื่อผลรวมของข้อมูลนั้นมีค่ามากกว่า 200 ให้หาว่ามีการรับข้อมูลทั้งหมดกี่จำนวน เมื่อโปรแกรมทำงานเสร็จให้ทำงานวนรับข้อมูลต่อไป โดยขึ้นข้อความสอบถามว่า Exit program (Y/N) ? หากผู้ใช้ป้อนข้อมูล y หรือ Y ให้หยุดการทำงานของโปรแกรม แต่ถ้าป้อนข้อมูลอื่น ๆ ให้เรียกโปรแกรมเดิมกลับมาทำงานอีกครั้ง

```
#include <stdio.h>
void main( ) {
   int num, sum=0, count=0;
   char ch;
   do {
      do {
           printf(“Enter number : “):
          scanf(“%d”, &num); flushall( );
          sum += num;
          count++;
       } while (sum <= 200);
      printf(“Enter number %d times”, count);
      printf(“\n\nExit program (Y/N) ? “);
      scanf(“%c”, &ch);
   } while (ch == ‘y’ || ch == ‘Y’);
}
```