ผู้สอน ผศ. นันท์นภัส โตอดิเทพย์

บทที่ 4

การทำรอบ

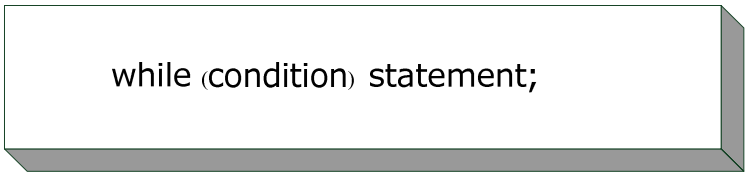
(Loops)

การทำรอบหรือการทำซ้ำ มี 3 แบบคือด้วยคำสั่ง while, do while และ for การทำซ้ำด้วย while และ do while ใช้ในเมื่อไม่ทราบจำนวนครั้งที่ทำซ้ำ การทำซ้ำด้วยคำสั่ง for มักใช้เอทราบจำนวนครั้งที่ทำซ้ำ

คำสั่ง **while**

คำสั่ง while ให้ทำซ้ำจนกระทั่งนิพจน์เงื่อนไขให้ค่าเท็จ โดยตรวจสอบเงื่อนไขก่อนการทำงานครั้งแรก หมายความว่าถ้าเงื่อนไขเป็นเท็จก็จะไม่ทำงานให้ทันที

รูปแบบคำสั่ง

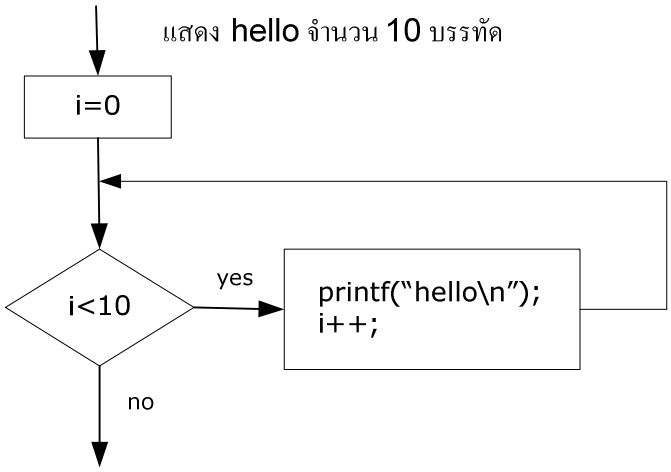



```
i=0;
while (i<10)
{  printf("hello\n");
   i++;
 }
```

แสดง hello จำนวน 10 บรรทัดแสดงปีน flowchart ในรูป 4.1

```
scanf("%d", &x);
while (x)
{  sum = sum +x;
    scanf("%d", &x);   ←——   เป็นการเปลี่ยนแปลงค่าของ x ใหม่
}
```

ในระหว่างที่รับค่า x ที่มีค่าไม่ใช่ 0 (ซึ่งหมายถึงจริง) เข้าทำงานใน body ของคำสั่ง while



รูป 4.1

ซึ่งทำการบวกสะสมค่า x ในตัวแปร sum คำสั่งสุดท้ายของ while ควรเป็นการทำเปลี่ยนแปลงเงื่อนไขใน while มิฉะนั้นเป็นการทำงานลูปไม่จบ (forever loop) และถ้าในครั้งแรกผู้ใช้รับค่าเข้าเป็น 0 เงื่อนไขที่ตรวจสอบเป็นเท็จจะไม่ทำงานใน body ของ while เลย

```
char  name[40] = "I LOVE SILPAKORN."
int i = 0;
while (name[i] != '\0')
     {  putchar(name[i]);
        i++;
      }
```

แสดงแคแรคเตอร์ในสตริง name ในระหว่างที่แคแรคเตอร์ใน name[i] ไม่เท่ากับ '\0' หรือ NULL ให้แสดงแคอรคเตอร์นั้น หรือในระหว่างที่ name[i] เป็นจริงนั่นเอง

```
i=1;
while (i<=10)
 {
        printf ("%d Multiple value i =%d\n"  i, 2*i );
        i++;
 }
```

แสดงค่า i (จาก 1 ถึง 10) และค่า 2*i ที่จอภาพ เริ่มเซตค่า i เป็น 1 และเพิ่มค่า i รอบละ 1 จนกระทั่งค่าถึงเป็น 11 (เงื่อนไขเป็นเท็จ) จึงเลิกทำงานตามคำสั่งในลูป

คำสั่ง do..while

คำสั่ง do..while ให้ทำซ้ำจนกระทั่งนิพจน์เงื่อนไขให้ค่าเท็จเช่นเดียวกับ while แต่จะตรวจสอบเงื่อนไขหลังการทำงานในส่วน body

รูปแบบคำสั่ง

```
do {
      statements;
   } while (condition);
```

```
i=0;
do
   {  printf("hello\n");
       i++;
    } while (i<10)  ;
```

แสดง hello จำนวน 10 บรรทัด แสดงเป็น flowchart ในรูป 4.2

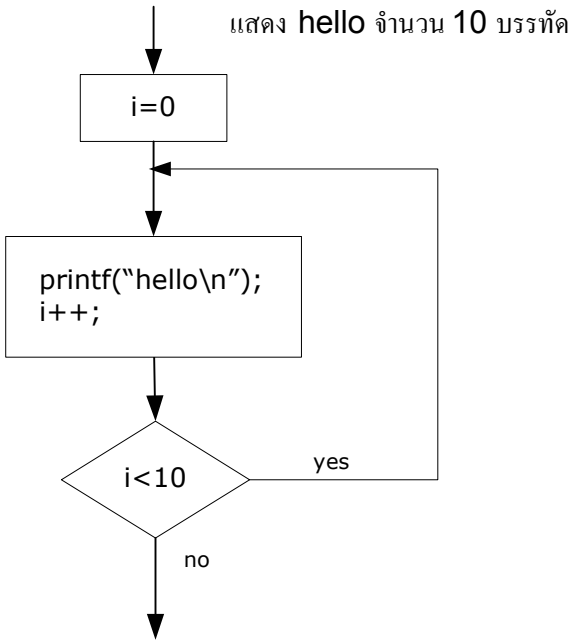
```
do
  {  scanf("%d", &x);
     if (x)  sum = sum +x;

   }
```

รับค่า x และบวกสะสมในตัวแปร sum

```
int i=1;
do {
     printf("%d Multiple value   i = %d\n ", i, 2*i);
     i++;
} while (i<=10);
```

ลูป do-while แสดงค่าของตัวแปร i และค่า 2*i จาก 1..10



รูป 4.2

คำสั่ง **for**

คำสั่ง for ทำซ้ำตามตัวแปรที่กำหนดเป็นเงื่อนไข ส่วนเงื่อนไขของคำสั่งมี 3 ส่วนคือ

1. ส่วนแรก เป็นค่าเริ่มต้นของตัวแปรลูป (loop variable)

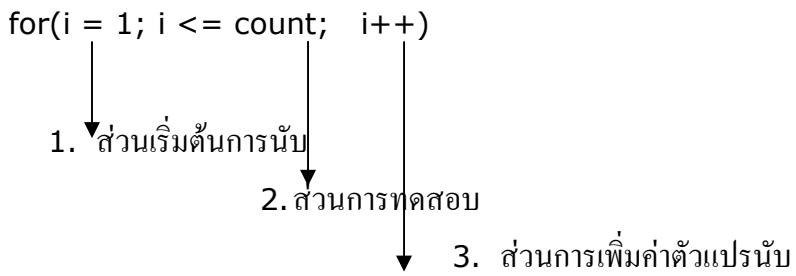
2. ส่วนที่สอง เป็นส่วนตรวจสอบ จะจบการทำซ้ำเมื่อค่าในส่วนนี้เป็นเท็จ

3. ส่วนที่สาม เป็นคำสั่งที่ทำงานทุกครั้งที่ส่วน body ของลูปทำงานเสร็จ ส่วนนี้มักเป็นส่วนที่เพิ่มค่าเป็นส่วนที่เปลี่ยนแปลงค่าในนิพจน์เงื่อนไขในส่วนที่สอง

รูปแบบคำสั่ง

```
for (initialization; condition; increment) statement;
```

เช่น การหาผลบวกของจำนวน 1.. 20 กำหนดค่าเริ่มต้นของผลรวมเป็น 0.0 ในตัวแปร total ใช้ตัวแปร i เป็นการนับจำนวนครั้งของการบวกสะสม ดังนั้นส่วนเงื่อนไขของคำสั่ง for เป็นดังนี้

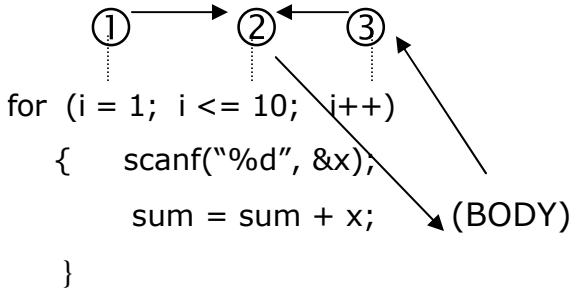


มีลำดับการทำงานดังนี้

1. i=1;
2. ตรวจสอบเงื่อนไขในส่วนที่สอง if (i <=10) เป็นจริงหรือ เท็จ
3. ถ้าจริงทำคำสั่งในส่วน body ของ for คือ

   รับค่า x จากผู้ใช้

   บวกสะสมใน sum
4. i++ เพิ่มค่า i
5. ไปทำในข้อ 2 ตรวจสอบเงื่อนไข



แสดงเป็น flowchart ในรูป 4.3

สังเกตุว่าส่วนการเซตค่าเริ่มต้นถูกกระทำเพียงครั้งเดียว และส่วน body ของคำสั่ง for อาจไม่ได้ถูกทำงานเลย เมื่อพบว่าเงื่อนไขเป็นเท็จ

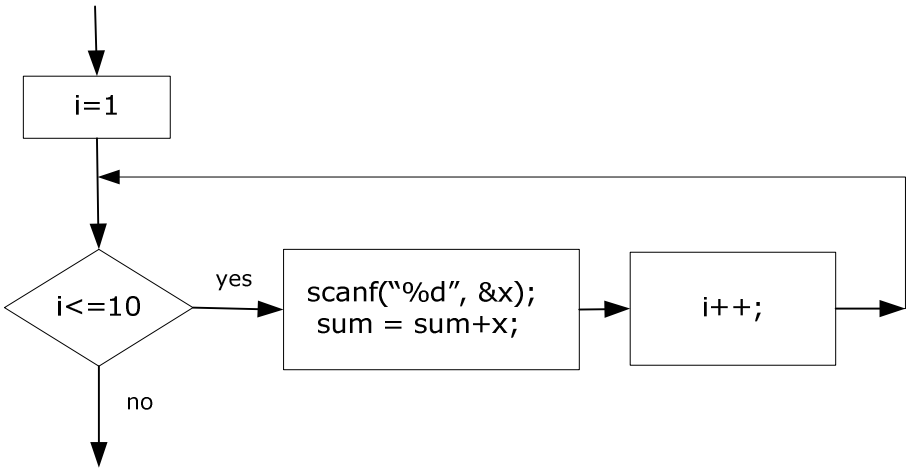
ส่วนของโปรแกรม

```
void main()
{
     const  int count = 20;
     float total = 0.0;
```

```
    int i;

    for(i = 1; i <= count; i++)
            total += i;
    printf("sum from 1..%d  = %f\n", count, total);
    printf("average from 1..%d  = %f\n", count, total/count);

}
```



รูป 4.3

ในแต่ละส่วนของคำสั่ง for อาจมีคำสั่งมากกว่า 1 คำสั่งได้ คั่นด้วยเครื่องหมาย , จากตัวอย่างนี้คือการเซตค่าเริ่มต้นให้กับตัวแปร x,y และส่วนการเพิ่มค่ามี 2 คำสั่งคือเพิ่มค่า x และลดค่า y อีกหนึ่ง จำนวนครั้งที่ทำงานซ้ำในลูป for คือ 50 ครั้ง จากค่า x = 0 ถึงค่า x = 49 ค่าสุดท้ายของตัวแปร y คือ 1

```
int count = 0;
for( x=0, y=50; x < 50; x++, y--)
  count++;
```

จากตัวอย่างเดิม เมื่อเพิ่มการตรวจสอบเงื่อนไขในการทำซ้ำ ถ้าค่าของ x เท่ากับค่าของ y ให้ขึ้นไปที่ส่วนหัวของ for ดังนั้นในกรณีนี้จะไม่นับครั้งในตัวแปร count ซึ่งเกิดขึ้นกรณีเดียวคือเมื่อ x และ y มีค่า 25 ดังนั้นค่าของตัวแปร count หลังจบการทำงานของลูปเป็น 49

```
int count = 0;
for( x=0, y=50; x < 50;  x++, y--)
 {  if (x==y)  continue;
   count++;
  }
```

เมื่อเปลี่ยนแปลงโปรแกรมด้วยการตรวจสอบว่า ถ้าค่าของ x เท่ากับค่าของ y ให้ออกจากการทำซ้ำด้วยคำสั่ง break ส่วนของโปรแกรมนี้จึงมีการทำซ้ำจำนวน 25 ครั้ง

```
int count = 0;
for( x=0, y=50; x < 50; x++, y--)
  {  if (x==y)  break;
   count++;
   }
```

คำสั่ง **break**

คำสั่ง break ใช้เพื่อออกจากลูปการทำซ้ำของคำสั่ง while, do..while, for หรือคำสั่ง switch ไปยังคำสั่งแรกถัดจาก loop หรือ switch นั้น คำสั่ง switch ใช้คำสั่ง break เพื่อออกจากการตรวจสอบค่าตัวแปร เมื่อได้พบค่าและดำเนินการบางอย่างที่ต้องการแล้ว มิฉะนั้นจะทำงานไหลต่อมาจนครบทุกเงื่อนไขที่เหลือ

คำสั่ง **continue**

คำสั่ง continue เพื่อกระโดดไปยังส่วนควบคุมของลูป

ในคำสั่ง while ไปที่ส่วนการทดสอบเงื่อนไข

ในคำสั่ง do..while ไปที่ส่วนการทดสอบเงื่อนไข

ในคำสั่ง for ไปที่ส่วนการการเพิ่มค่า และทำงานตาม iteration ของลูปต่อไป

```
count=0;
sum=0;

  for(i = 1; i <= 10;  i++)
  {    sum = sum + i;
       continue;
       count++;
   }
```

```
        for(i = 1; i <= count;   i++)
{   sum = sum + i;
                break;
                count++;
            }
```

ถ้ารับค่า x 5 , 7 , 10, 15

```
    scanf("%d",&x);
     while (x)
     {   if (x%3==0)  continue;
        sum += x;
        scanf("%d",&x);
     }
```

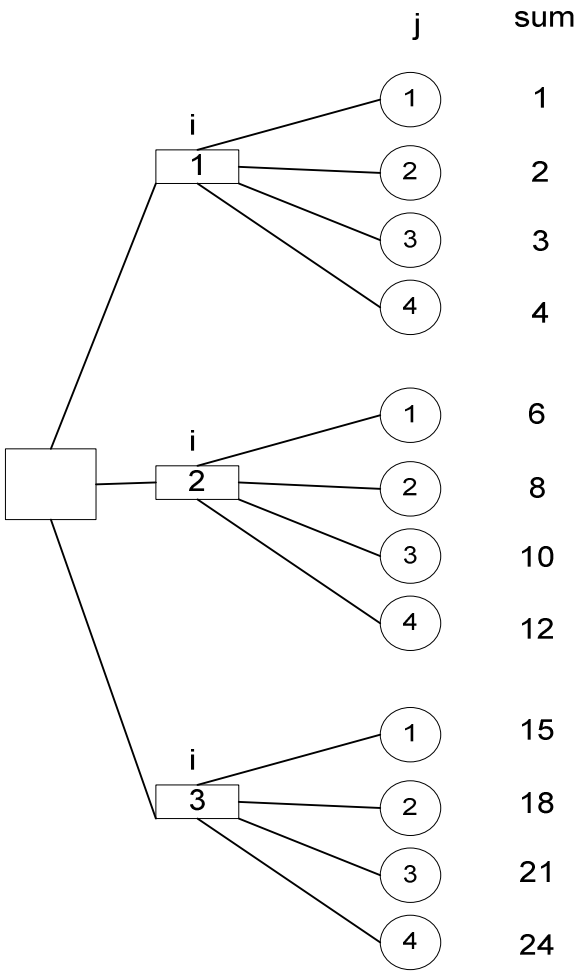
ผลการทำงานของคำสั่งนี้เป็นเช่นไร และค่าในตัวแปร sum ล่าสุดเป็นเท่าไร

ผลการทำงานจะเป็นการทำซ้ำไม่รู้จบ หลังจากได้บวกสะสมไป 3 จำนวนแล้วค่า sum ล่าสุดเป็น 22 เมื่อรับค่า 15 ซึ่งหารด้วย 3 ลงตัวให้ไปที่ continue ตรวจสอบเงื่อนไขอีก ก็เป็นจริงตลอดไป

Nested for

บ่อยครั้งมากที่การทำงานพบว่าต้องใช้คำสั่ง for ซ้อน for เช่นบวกสะสมค่า i ใน sum ในการท้ำซ้ำ for สองชั้น ชั้นที่หนึ่งรันด้วย i ชั้นที่สองรันด้วย j

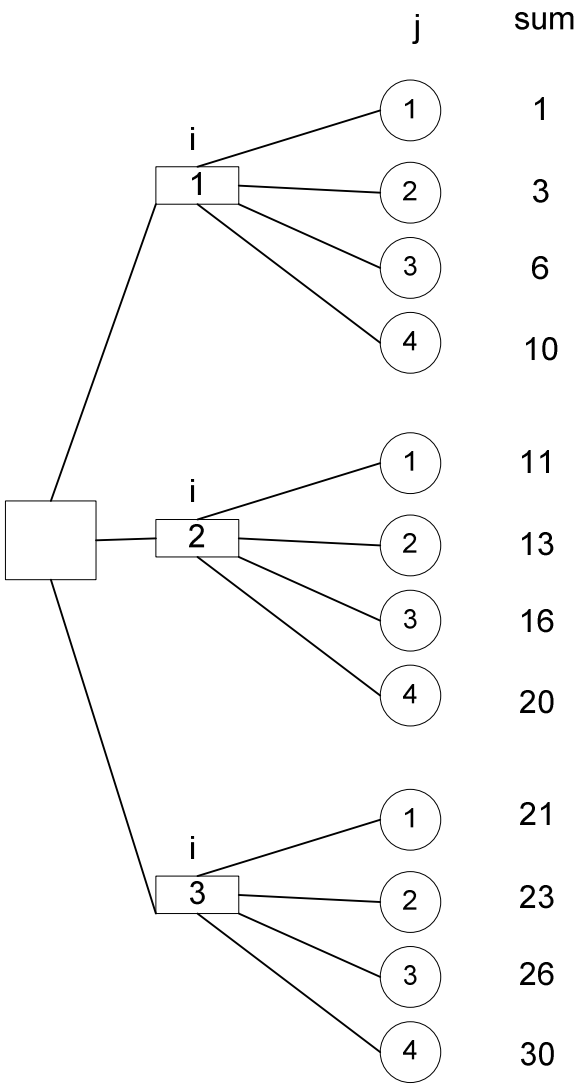
```
        for (i=1; i<=3; i++)
                for (j=1; j<=4; j++)
                        sum = sum + i;
```



การทำงานในรอบสองต้องจาก j =1 ถึง 4 ครบจึงเปลี่ยนค่าในรอบนอกคือ i จาก 1 ถึง 3 ดังรูป ค่าในตัวแปร sum คือ 24

คำสั่ง for นี้บวกสะสม ค่า j ใน sum คำตอบเป็น 30 ดังแสดงรูป

```
for (i=1; i<=3; i++)
                for (j=1; j<=4; j++)
                        sum = sum + j;
```



สรุป

การทำงานเป็นรอบหรือการทำซ้ำใช้การตรวจสอบเงื่อนไขเพื่อทดสอบนิพจน์เงื่อนไขว่าเป็นจริงหรือเป็นเท็จ โดยที่ถ้าเงื่อนไขเป็นจริงทำงานรอบให้อีก ถ้าเงื่อนไขเป็นเท็จหยุดการทำงานซ้ำนั้น แม้ว่าคำสั่ง for จะดูเหมือนเป็นการทำซ้ำเท่าจำนวนครั้งที่กำหนด แต่เป็นการตรวจสอบเงื่อนไขที่จะทำซ้ำหรือไม่อยู่นั่นเอง การเพิ่มค่าในคำสั่ง for ไม่จำเป็นต้องเป็นการเพิ่มค่าเป็นจำนวนเต็ม

ผู้สอน ผศ. นันท์นภัส โตอดิเทพย์

แบบฝึกหัด

1. เหตุใดผลของคำสั่งทำรอบนี้ จึงทำงานไม่รู้จบ

```
while (1)  printf ("Hello \n");
```

2. จากคำสั่งนี้ แสดงข้อความ "Hello" กี่ครั้ง

```
int i=0;
do
{
        printf("Hello \n ");
} while (i <10)
```

3. จากคำสั่งนี้ แสดงข้อความ "Hello" กี่ครั้ง

```
int i=0;
do
{
        printf("Hello \n ");   i++;
} while (i <10);
```

4. จงใช้คำสั่ง for แสดงข้อความ "Hello" จำนวน 10 ครั้ง

5.

ก. จงบอกความแตกต่างของ

```
while (x) {          }
```

และ

```
do
{

 } while (x)
```

ข. ถ้า i มีค่า 11 การทำงานของลูปทั้งสองเป็นอย่างไร

```
while (i<=10)
{   printf("Loop Body\n");
    i++;
}
```

```
do {
  printf("Loop Body\n");
        i++;
} while (i<=10);
```

6.จงบอกผลของคำสั่ง

```
unsigned char c;
while ((c=getch()) != '.')
    printf("\t%c %d\n", c,c);
```

7. ถ้าเปลี่ยนแปลงโปรแกรมในข้อ 1 ดังนี้ ผลลัพธ์จะต่างจากเดิมหรือไม่ เพราะเหตุใด

```
unsigned char c;
while (c=getch()!= '.')
    printf("\t%c %d\n", c,c);
```

8. จงบอกผลของคำสั่ง

```
int i=0;
while (i < 256 )
 {  putchar(i);
    i++);
 }
```

9. เหตุใดโปรแกรมนี้ จึงทำงานไม่รู้จบ

```
unsigned char c = 0;
while (c < 256 )
{ putchar(c);
   c++);
  }
```

10. จงเปลี่ยนการทำงานในข้อ 9 ด้วยคำสั่ง for

11. จงใช้คำสั่ง for เพื่อแสดงผลตัวเลข 1..5 ดังนี้

```
1
12
123
1234
12345
```

12. จงใช้คำสั่ง for ให้ได้ผลลัพธ์ดังนี้

```
*
**
***
****
*****
```

13. จงเขียนคำสั่ง for (1 ชั้น) เพื่อพิมพ์เลข 1 .. 50 ให้พิมพ์บรรทัดละ 5 ค่า

14. ก. จงอธิบายคำสั่ง for

```
for (x=0, y=0; x <20, y < 9; x++, y+=3)
    printf("%d %d\n", x, y);
```

ข. จงแสดงผลที่ได้ของคำสั่ง for ในข้อ ก

ค. การทำงานของ for นี้ได้ผลเหมือนในข้อ ก หรือไม่

```
for (x=0, y=0; x <20 && y < 9; x++, y+=3)
        printf("%d %d\n", x, y);
```

ง. จงแสดงผลการทำงานของ

```
for (x=0, y=0; x <20 || y < 9; x++, y+=3)
        printf("%d %d\n", x, y);
```

15. จงตอบค่าของ total เมื่อจบการทำงานของโปรแกรมนี้

```
void main()
{   const int count=100;
    float total = 0.0;
    int i;
        for(i = 1; i <= count; i++)   total += ( count % 3 );
        printf("result = %f\n", total);
}
```

16. จงหาข้อผิดพลาดของโปรแกรม ที่ไม่สามารถหาค่าเฉลี่ยได้ อย่างน้อย 2 แห่ง

```
void  main()
{   int   x, count = 10;
          double total=0.0;
     while (x)
     {  scanf("%d",&x);
         total += x;
      }
```

```
    printf("%8.2f", total/count);
}
```

17. จงเขียนโปรแกรมรับเข้าตัวเลขจำนวนจริง นับจำนวนที่รับเข้า หาค่าเฉลี่ย ค่าสูงสุด และค่าต่ำสุดของค่าที่รับเข้า

%%%%%%%%%%%%%%%%%%%%%%%%%