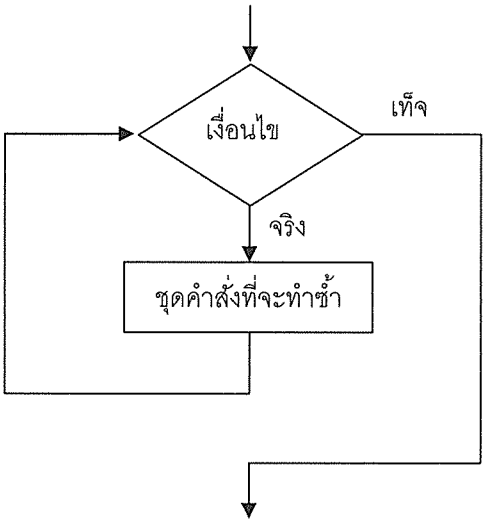
# บทที่ 7

# โครงสร้างแบบวนรอบทำซ้ำ

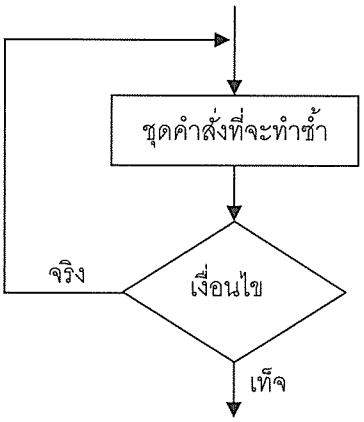
คอมพิวเตอร์มีความสามารถในการทำงานซ้ำๆได้ดี การทำซ้ำจะเรียกว่าลูป ซึ่งมีอยู่หลายประเภท โดยการทำลูปนั้นจะต้องมีการตรวจสอบเงื่อนไขด้วยว่าจะให้ทำซ้ำกี่ขั้นตอน เมื่อใดต้องการให้หยุดทำซ้ำ ในภาษาซีมีคำสั่งลูปการทำซ้ำอยู่ 3 ประเภท ซึ่งจะต้องทำความเข้าใจการใช้งานคำสั่งแต่ละคำสั่งก่อนจึงสามารถเลือกนำมาใช้ได้

โดยทั่วไปแล้ว การทำงานของโปรแกรมคอมพิวเตอร์จะทำงานเรียงตามลำดับ ตั้งแต่สเตตเมนต์แรกจนถึงสเตตเมนส์สุดท้าย แต่เราสามารถให้คอมพิวเตอร์ทำงานซ้ำๆ ที่สเตตเมนต์ชุดหนึ่งได้โดยใช้คำสั่งควบคุมให้ทำงานซ้ำได้ บางครั้งจะเรียกว่า คำสั่งลูป

การสั่งให้คอมพิวเตอร์ทำงานซ้ำๆ นั้นจะต้องมีการควบคุมว่าจะให้โปรแกรมหยุดทำซ้ำเมื่อใด โดยการควบคุมจะใช้การตรวจสอบเงื่อนไขเป็นตัวควบคุมการทำซ้ำ ซึ่งมีอยู่ 2 ประเภท คือ การตรวจสอบก่อนทำซ้ำและการตรวจสอบหลังทำซ้ำ ดังรูป




การทำงานประเภทที่ 1




การทำงานประเภทที่ 2

การทำงานประเภทที่หนึ่งจะตรวจสอบเงื่อนไขก่อนการทำซ้ำ เมื่อโปรแกรมทำงานมาถึงการทำซ้ำจะตรวจสอบเงื่อนไขก่อน ถ้าหากเงื่อนไขเป็นจริงโปรแกรมจะทำซ้ำชุดคำสั่งที่กำหนด จากนั้นจะกลับมาตรวจสอบเงื่อนไขอีก และทำไปเรื่อยๆ จนกว่าเงื่อนไขจะเป็นเท็จจึงจะออกจากการทำซ้ำ

การทำงานประเภทที่สองจะตรวจสอบเงื่อนไขภายหลังการทำคำสั่ง คือโปรแกรมจะทำชุดคำสั่งที่ต้องการทำซ้ำก่อน จากนั้นจะตรวจสอบเงื่อนไข ถ้าเงื่อนไขเป็นจริงจะทำชุดคำสั่งที่ต้องการทำซ้ำต่อไป และกลับมาตรวจสอบเงื่อนไขอีก เมื่อเงื่อนไขเป็นเท็จโปรแกรมจะออกจากการทำซ้ำ

โดยทั่วไปการทำซ้ำนั้นมักจะมีตัวแปรอยู่ภายในลูปการทำซ้ำด้วย โดยตัวแปรนี้จะมีค่าเปลี่ยนไปแต่ละครั้งเมื่อมีการทำซ้ำ และจะใช้ตัวแปรนี้มาตรวจสอบเงื่อนไข ในภาษาซีมีคำสั่งให้ทำงานซ้ำอยู่ 3 รูปแบบ ขึ้นอยู่กับการตรวจสอบเงื่อนไขของการทำซ้ำ คือ

1. คำสั่ง for
2. คำสั่ง while
3. คำสั่ง do…while

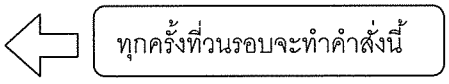
## 7.1 การวนรอบทำซ้ำด้วยคำสั่ง for

การทำซ้ำแบบ for หรือ ลูป for จะเป็นการให้โปรแกรมทำซ้ำจนกว่าค่าตัวแปรจะครบตามที่ตั้งไว้ เริ่มแรกโปรแกรมจะกำหนดค่าเริ่มต้นให้กับตัวแปรเริ่มต้น จากนั้นจะทำสเตตเมนต์โดยรูปแบบของคำสั่งเป็นดังนี้

```
for ( ค่าเริ่มต้น ; เงื่อนไข ; เพิ่มค่าหรือลดค่า )
    Statement
```

***ตัวอย่าง เช่น***

```
for ( i=1 ;  i<5 ; i++ )
    Statement;
```



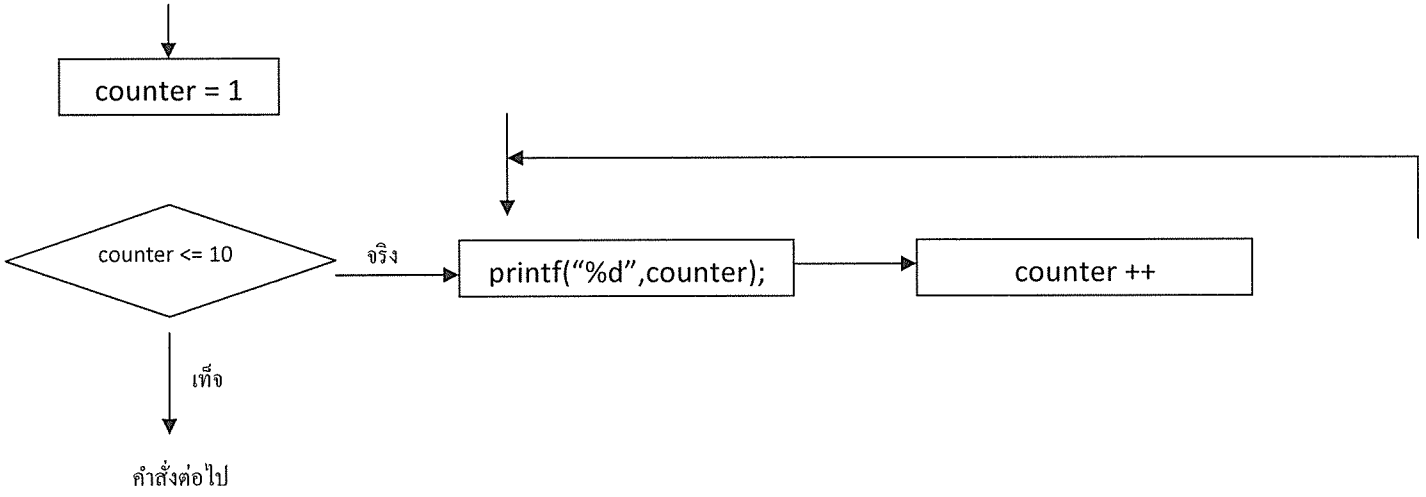

แรกเริ่มโปรแกรมจะใส่ค่าเริ่มต้น 1 ลงในตัวแปร i จากนั้นจะทดสอบเงื่อนไขว่าเงื่อนไขเป็นจริงหรือไม่ ถ้าเป็นจริงจะทำสเตตเมนต์ และจะเพิ่มค่า i ขึ้นหนึ่งค่า

ตัวอย่างที่ 1

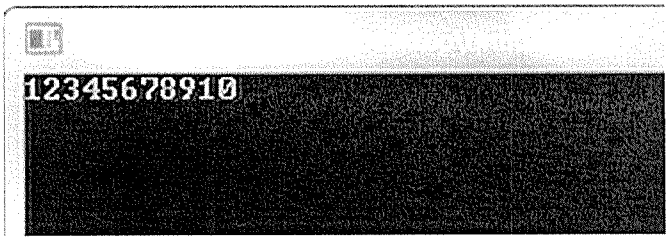
```
for  (counter = 1 ; counter <= 10 ; counter++)
        printf ( “%d” , counter);
```

โปรแกรมจะพิมพ์ค่า counter ตั้งแต่ 1 ถึง 10 โดยเริ่มแรกจะใส่ค่าให้กับตัวแปร counter ซึ่งเป็นตัวแปรเริ่มต้น จากนั้นจะตรวจสอบเงื่อนไขว่า counter น้อยกว่า 10 จริงหรือไม่ ถ้าจริงจะพิมพ์ค่า countฟ และเพิ่มค่า counter ขึ้นอีกหนึ่งค่า จากนั้นจะตรวจสอบเงื่อนไขใหม่

จากการทำงานข้างต้นสามารถเขียนเป็นผังงานได้ดังนี้



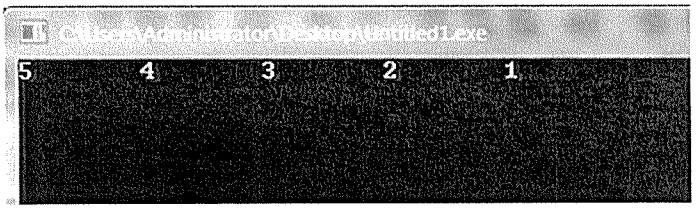

```
#include<stdio.h>
#include<conio.h>
main()
{
    int counter;
    for  (counter = 1 ; counter <= 10 ; counter++)
   printf ( "%d" , counter);
    getch();
}
```



ตัวอย่างที่ 2

เมื่อรันโปรแกรมจะทำให้คอมพิวเตอร์ลดค่า 5 จนถึง 1 ดังต่อไปนี้

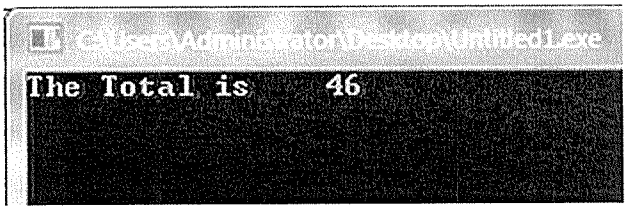
```
#include<stdio.h>
#include<conio.h>
main()
{
    int x;
    for  ( x=5; x>0 ; x-- )
         printf ( "%d\t" , x);
    getch();
}
```




ตัวอย่างที่ 3 ตัวอย่างโปรแกรม การใช้โครงสร้าง for ในการหาผลรวม

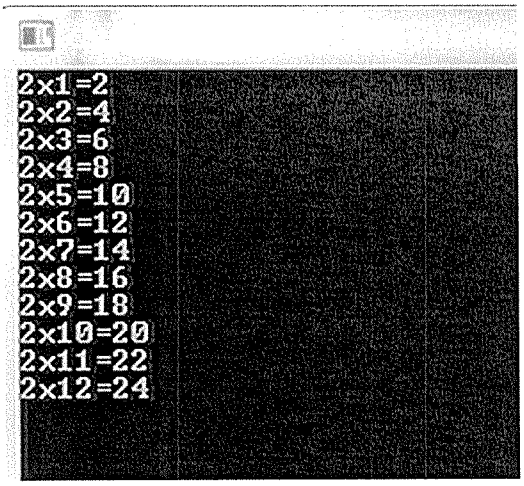
```
#include<stdio.h>
#include<conio.h>
main()
{
    int ctr;
    int total = 0;
    for ( ctr=10 ; ctr<=13 ; ctr++ )
        {
            total=total+ctr;
         }
         printf("The Total is \t%d",total);
    getch();
}
```

*** ถ้าหากต้องการให้สเตตเมนต์ที่จะทำงานซ้ำเป็นสเตตเมนต์รวม จะต้องเขียนสเตตเมนต์ในเครื่องหมาย { และเครื่องหมาย }



ตัวอย่างที่ 4 ตัวอย่างการเขียนโปรแกรมให้แสดงเป็นตารางสูตรคูณ

```
#include<stdio.h>
#include<conio.h>
main()
{
    int x;
    for (x=1;x<=12;x++)
        printf("2x%d=%d\n",x,2*x);
    getch();
}
```




## 7.2 ลูป while Statement

ประโยคคำสั่งลูปแบบ While จะใช้ให้โปรแกรมทำงานซ้ำโดยตรวจสอบเงื่อนไขก่อน ถ้าเงื่อนไขเป็นจริงจะทำซ้ำ และจะวนรอบจนกว่าเงื่อนไขจะเป็นเท็จ ลูปนี้จะต่างกับลูป for เพราะจำนวนครั้งที่ทำซ้ำจะไม่แน่นอนขึ้นอยู่กับเงื่อนไข รูปแบบของคำสั่งเป็นดังนี้

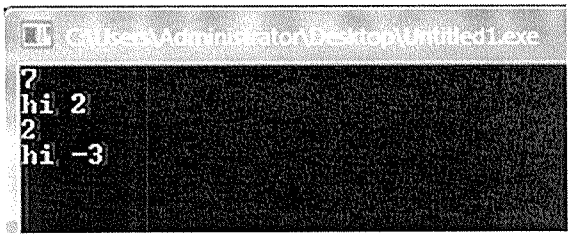
```
while ( ตรวจสอบเงื่อนไข )
        Statement;
```

ในการใช้คำสั่งนี้จะเริ่มต้นด้วยคำว่า while และตรวจสอบเงื่อนไข จากนั้นจะตามด้วยสเตตเมนต์ที่จะทำงานในการตรวจสอบเงื่อนไขนั้นจะใช้ตัวดำเนินการเปรียบเทียบแบบบูลีน ตัวอย่างการใช้งานเป็นดังชุดคำสั่งต่อไปนี้

```
#include<stdio.h>
#include<conio.h>
main()
{
    int n = 7;
    while (n >= 0)
    {
        printf("%d\n",n);
        n = n - 5;
        printf("hi %d\n",n);
    }
    getch();
}
```

ตรวจสอบเงื่อนไข ถ้าผลลัพธ์เป็นจริง

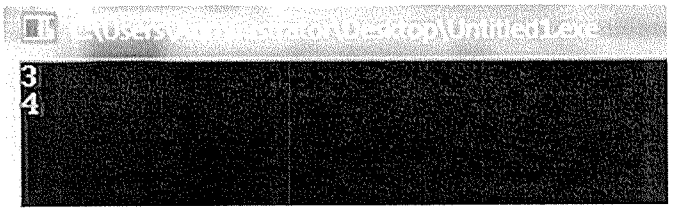
ทำกลุ่มคำสั่ง




ในลูปแรก n มีค่าเท่ากับ 7 ทำให้เงื่อนไขเป็นจริง โปรแกรมจะทำงานในลูปซึ่งจะทำให้ n มีค่าเป็น 2 ต่อมา โปรแกรมตรวจสอบเงื่อนไขเพื่อทำลูปที่สอง พบว่าเงื่อนไขเป็นจริงเมื่อทำลูปที่สองทำให้ n มีค่าเป็น -3 เมื่อตรวจสอบเงื่อนไขพบว่าเป็นเท็จจึงไม่ทำลูปที่สาม การทำงานจึงจบแค่ลูป 2

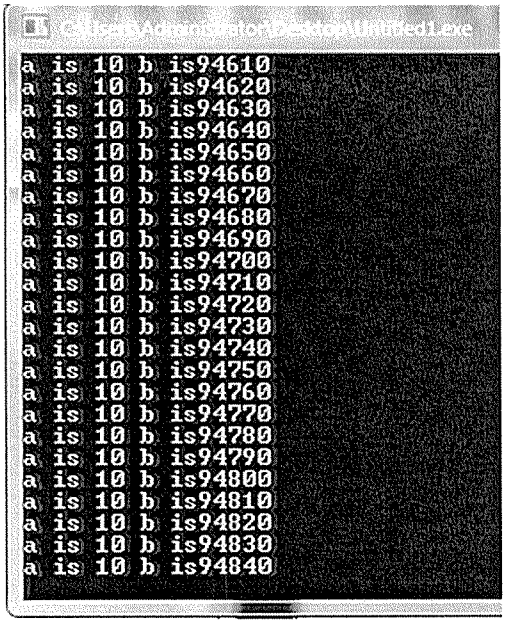
ตัวอย่างที่ 1

```
#include<stdio.h>
#include<conio.h>
main()
{
    int c = 3;
    while (c < 5)
    {
        printf("%d\n",c);
        c++;
    }
    getch();
}
```



ตัวอย่างที่ 2

```
#include<stdio.h>
#include<conio.h>
main()
{
    int a=10;
    int b=20;
    while (a > 5)
    {
        printf("a is %d b is%d\n",a,b);
        b=b+a;
    }
    getch();
}
```




เนื่องจากต้องมีการกลับมาตรวจสอบเงื่อนไขทุกครั้งหลักจากทำการประมวลผลหนึ่งรอบ ***จึงมีความสำคัญอย่างยิ่งที่ภายในชุดคำสั่งจะต้องมีประโยคที่ทำหน้าที่ในการเปลี่ยนแปลงค่าของตัวแปรที่ใช้ในการตรวจสอบเงื่อนไข*** มิเช่นนั้นโปรแกรมจะทำการประมวลผลโดยไม่มีที่สิ้นสุด ก่อให้เกิดความผิดพลาดที่เรียกว่า การทำซ้ำแบบไม่สิ้นสุด ดังตัวอย่างข้างต้นนี้

## 7.3 ลูป do...while

คำสั่งลูปแบบนี้จะทำการตรวจสอบเงื่อนไขภายหลังการทำงานในลูป โดยโปรแกรมจะทำลูปซ้ำไปเรื่อยๆ ถ้าเงื่อนไขเป็นจริงจะทำโปรแกรมซ้ำต่อไป จนกระทั่งเงื่อนไขที่เปรียบเทียบอยู่นั้นเป็นเท็จจึงจะหยุดทำ เนื่องจากลูปแบบนี้จะตรวจสอบเงื่อนไขหลังจากทำลูป จึงทำให้ประโยคในลูปถูกทำหนึ่งครั้งเสมอ ซึ่งต่างจากลูปแบบอื่นๆ

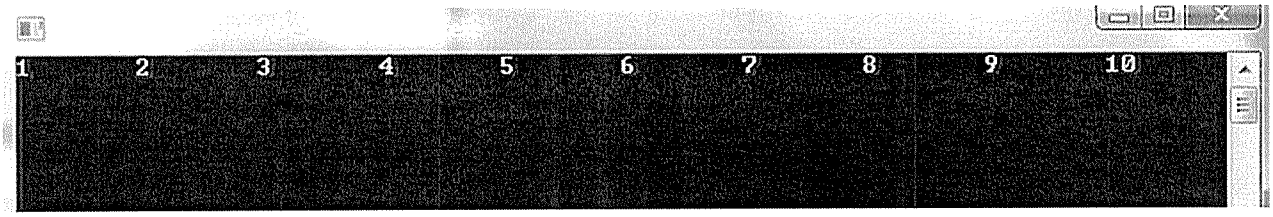
รูปแบบคำสั่งเป็นดังนี้

```
do
  {
      Statement;
  } while ( ตรวจสอบเงื่อนไข );
```

ตัวอย่างที่ 1

```
#include<stdio.h>
#include<conio.h>
main()
{
    int counter = 1;
    do
    {
       printf("%d\t",counter);
    }while (++counter <= 10);
    getch();
}
```

โปรแกรมนี้เป็นตัวอย่างการใช้ลูป do…while พิมพ์ตัวเลขตั้งแต่ 1 ถึง 10



## 7.4 คำสั่ง break และ continue

จากตัวอย่างที่ผ่านมาเราได้ทดลองใช้คำสั่ง break มาบ้างแล้ว โดยคำสั่งนี้สามารถใช้งานร่วมกับ while, for, do...while หรือ switch ได้ สำหรับคำสั่งที่ทำงานตรงข้ามกับคำสั่ง break คือคำสั่ง continue ซึ่งสามารถใช้ใน while, for, do...while ได้เช่นกัน

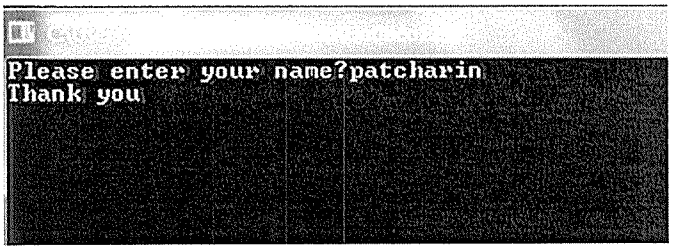
⊙ เมื่อโปรแกรมทำงานมาจนถึงคำสั่ง break จะหยุดการประมวลผลแล้วออกจากโครงสร้างนั้นทันทีและจะไปทำคำสั่งที่ตามหลังโครงสร้างนั้น

⊙ เมื่อโปรแกรมทำงานมาจนถึงคำสั่ง continue จะทำโครงสร้างนั้นต่อไปโดยไม่ทำคำสั่งที่ตามหลัง continue

ตัวอย่างที่ 1 โปรแกรมแสดงการใช้คำสั่ง break

```
#include<stdio.h>
#include<conio.h>
main()
{
   char name[15];
   int a=1;
   do
   {
      printf("Please enter your name?");
      scanf("%s",&name);
      break;
      a++;
   }while (a<5);
   printf("Thank you");
    getch();
}
```

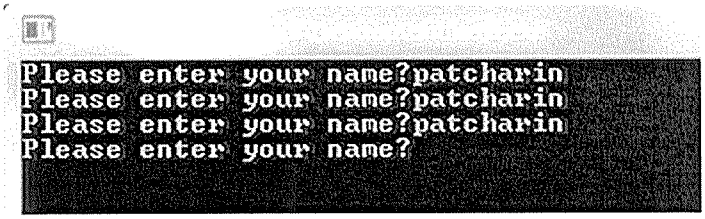
จะสังเกตว่า เมื่อประมวลผลมาจนถึงคำสั่ง break; โปรแกรมจะออกจากโครงสร้าง do..while ทันทีแล้วจึงมาประมวลผลคำสั่งบรรทัด printf("Thank you");



ตัวอย่างที่ 2 โปรแกรมแสดงการใช้คำสั่ง continue

```
#include<stdio.h>
#include<conio.h>
main()
{
   char name[15];
   int a=1;
   do
   {
      printf("Please enter your name?");
      scanf("%s",&name);
      continue;
      a++;   <-------- ไม่ทำ
   }while (a<5);
   printf("Thank you");
    getch();
}
```

จะสังเกตว่า เมื่อประมวลผลมาจนถึงคำสั่ง continue; โปรแกรมจะไม่ทำคำสั่งที่อยู่หลัง continue; ซึ่งก็คือ a++ แต่โปรแกรมจะไปตรวจสอบว่า a < 5 จริงหรือไม่ แล้วแสดง Please enter your name? ออกมาเรื่อยๆ



## แบบฝึกหัดที่ 7.1

1. จงหาผลลัพธ์จากการประมวลผลโปรแกรมต่อไปนี้

```
#include<stdio.h>
#include<conio.h>
main()
{
   int m = 10;
   int n = 20;
   while(m==10)
   {
      m=m+10;
   }
   printf("M=%d",m);
   getch();
}
```

ผลลัพธ์

2. จงหาผลลัพธ์จากการประมวลผลโปรแกรมต่อไปนี้

```
#include<stdio.h>
#include<conio.h>
main()
{
   int a = 5;
   int b = 10;
   while(a != 5)
   {
      printf("Hi");
   }
   printf("Hello");
   getch();
}
```

ผลลัพธ์

3. จงหาผลลัพธ์จากการประมวลผลโปรแกรมต่อไปนี้

```
#include<stdio.h>
#include<conio.h>
main()
{
   int m = 2;
   int n = 3;
   do
   {
      printf("This is easy");
   }while((m==2)&&(n!=3));
   getch();
}
```

ผลลัพธ์

4. จงหาผลลัพธ์จากการประมวลผลโปรแกรมต่อไปนี้

```
#include<stdio.h>
#include<conio.h>
main()
{
   int a = 1;
   int b = 2;
   do
   {
      printf("Good morning\n");
      b--;
      break;
   } while(b>1);
   printf("Thank you");
   getch();
}
```

ผลลัพธ์

5. จงหาผลลัพธ์จากการประมวลผลโปรแกรมต่อไปนี้

```
#include<stdio.h>
#include<conio.h>
main()
{
    int m = 2;
    int n = 4;
   for (m=2;m<(n+n);m++)
   {
      continue;
      printf("%d\n",n);
   }
   printf("The last value of m is%d",m);
   getch();
}
```

ผลลัพธ์

6. จงหาผลลัพธ์จากการประมวลผลโปรแกรมต่อไปนี้

```
#include<stdio.h>
#include<conio.h>
main()
{
    int total=0;
    int counter=0;
    while (counter < 5)
    {
        total = total+20;
        counter++;
    }
    printf("The total is %d",total);
    getch();
}
```

ผลลัพธ์

7. จงหาผลลัพธ์จากการประมวลผลโปรแกรมต่อไปนี้

```
#include<stdio.h>
#include<conio.h>
main()
{
    int x;
    for  (x = 12 ; x <= 15 ; x++)
    printf ( "%d\n" ,x);
    getch();
}
```

ผลลัพธ์

8. จงหาผลลัพธ์จากการประมวลผลโปรแกรมต่อไปนี้

```
#include<stdio.h>
#include<conio.h>
main()
{
    int x;
    for  (x = 20 ; x >= 15 ; x--)
    printf ( "%d\n" ,x);
    getch();
}
```

ผลลัพธ์

## แบบฝึกหัดที่ 7.2

1. จงเขียนโปรแกรมแสดงแม่สูตรคูณ โดยให้รับค่าตัวเลขเข้ามาทางคีย์บอร์ดแล้วให้แสดงแม่สูตรคูณทางจอภาพ ดังนี้