หน่วยที่ 6 โครงสร้างแบบทำซ้ำ

สาระสำคัญ

ในการเขียนโปรแกรมนอกจากจะเขียนโปรแกรมโครงสร้างแบบลำดับ และโครงสร้างแบบมีเงื่อนไขแล้ว ในบางครั้งจำเป็นจะต้องเขียนโปรแกรมโครงสร้างแบบทำซ้ำ ทั้งนี้เนื่องจากการเขียนโปรแกรมในบางครั้งจะต้องมีการให้ทำซ้ำหลายๆ ครั้ง เช่น ให้แสดงชื่อนักเรียน 100 ครั้ง ให้หาผลบวกเลขตั้งแต่ 1-10 หรือให้ทำไปเรื่อยๆ จนกว่าจะกดเลข 0 เป็นต้น

ในการทำซ้ำลักษณะนี้ จำเป็นจะต้องมีการเขียนโปรแกรมโครงสร้างแบบทำซ้ำ ซึ่งมี 4 รูปแบบ คือ คำสั่งทำซ้ำแบบ while คำสั่งทำซ้ำแบบ do-while คำสั่งทำซ้ำแบบ for และคำสั่งทำซ้ำแบบ for ซ้อน ในการเลือกใช้โครงสร้างแบบทำซ้ำแต่ละรูปแบบ ขึ้นอยู่กับลักษณะของงานและผลลัพธ์ที่ต้องการ

จุดประสงค์การเรียนรู้

1. อธิบายรูปแบบการใช้คำสั่ง while, do-while และ for ได้
2. วิเคราะห์ความเหมือนและความแตกต่าง ระหว่างคำสั่งทำซ้ำแบบ while และ do-while ได้
3. วิเคราะห์ความเหมือนและความแตกต่าง ระหว่างคำสั่งทำซ้ำแบบ while และ for ได้
4. วิเคราะห์ความเหมือนและความแตกต่าง ระหว่างคำสั่งทำซ้ำแบบ do-while และ for ได้
5. อธิบายเหตุผลในการเลือกใช้คำสั่งทำซ้ำแบบ while, do-while และ for ได้
6. สามารถประยุกต์ใช้คำสั่งทำซ้ำแบบ while ในการเขียนโปรแกรมภาษาซีได้
7. สามารถประยุกต์ใช้คำสั่งทำซ้ำแบบ for ในการเขียนโปรแกรมภาษาซีได้
8. สามารถประยุกต์ใช้คำสั่งทำซ้ำแบบ do-while ในการเขียนโปรแกรมภาษาซีได้
9. สามารถประยุกต์ใช้คำสั่งทำซ้ำแบบ for ซ้อน ในการเขียนโปรแกรมภาษาซีได้

สาระการเรียนรู้

6.1 คำสั่งทำซ้ำแบบ while

6.2 คำสั่งทำซ้ำแบบ do-while

6.3 คำสั่งทำซ้ำแบบ for

6.4 คำสั่งทำซ้ำแบบ for ซ้อน

นักเขียนโปรแกรมไม่สามารถปฏิเสธได้ว่า การพัฒนาโปรแกรมในปัจจุบันนี้ มีความสลับซับซ้อนมากขึ้น เนื่องจากระบบงานมีความซับซ้อนและทันสมัยมากขึ้นนั่นเอง ดังนั้นการนำคอมพิวเตอร์มาใช้ในการปฏิบัติงานแทนมนุษย์หรือการนำคอมพิวเตอร์มาช่วยในการผ่อนแรงของมนุษย์ นักเขียนโปรแกรมจะต้องพัฒนาโปรแกรม ให้มีความละเอียดและทันสมัยมากขึ้นตามไปด้วย ทั้งนี้เพื่อให้โปรแกรมที่ผ่านการพัฒนาแล้ว สามารถใช้งานได้จริง การพัฒนาโปรแกรมในปัจจุบัน ไม่สามารถหลีกเลี่ยงการเขียนโปรแกรมแบบโครงสร้าง การเขียนโปรแกรมแบบโครงสร้างที่สำคัญ คือการเขียนโปรแกรมโครงสร้างแบบทำซ้ำ โดยในหน่วยการเรียนนี้จะขอกล่าวถึง คำสั่งที่เป็นโครงสร้างแบบทำซ้ำ 4 รูปแบบ คือ คำสั่งทำซ้ำแบบ while คำสั่งทำซ้ำแบบ do-while คำสั่งทำซ้ำแบบ for และคำสั่งทำซ้ำแบบ for ซ้อน

6.1 คำสั่งทำซ้ำแบบ while

เป็นคำสั่งโครงสร้างแบบทำซ้ำที่จะมีการทดสอบเงื่อนไขก่อนทำคำสั่งในลูป โดยหากเงื่อนไขที่อยู่หลัง while เป็นจริง จะทำคำสั่งในลูป และย้อนกลับมาทดสอบเงื่อนไขที่อยู่หลัง while ไปเรื่อยๆ จนกว่าเงื่อนไขที่อยู่หลัง while จะเป็นเท็จ จึงจะออกนอกลูป คำสั่งทำซ้ำแบบ while มีรูปแบบการใช้คำสั่งดังนี้

```
while(เงื่อนไข)
{
        คำสั่งที่ 1;
        …….
        คำสั่งที่ n;
}
คำสั่งให้ทำในลำดับถัดไป;
```

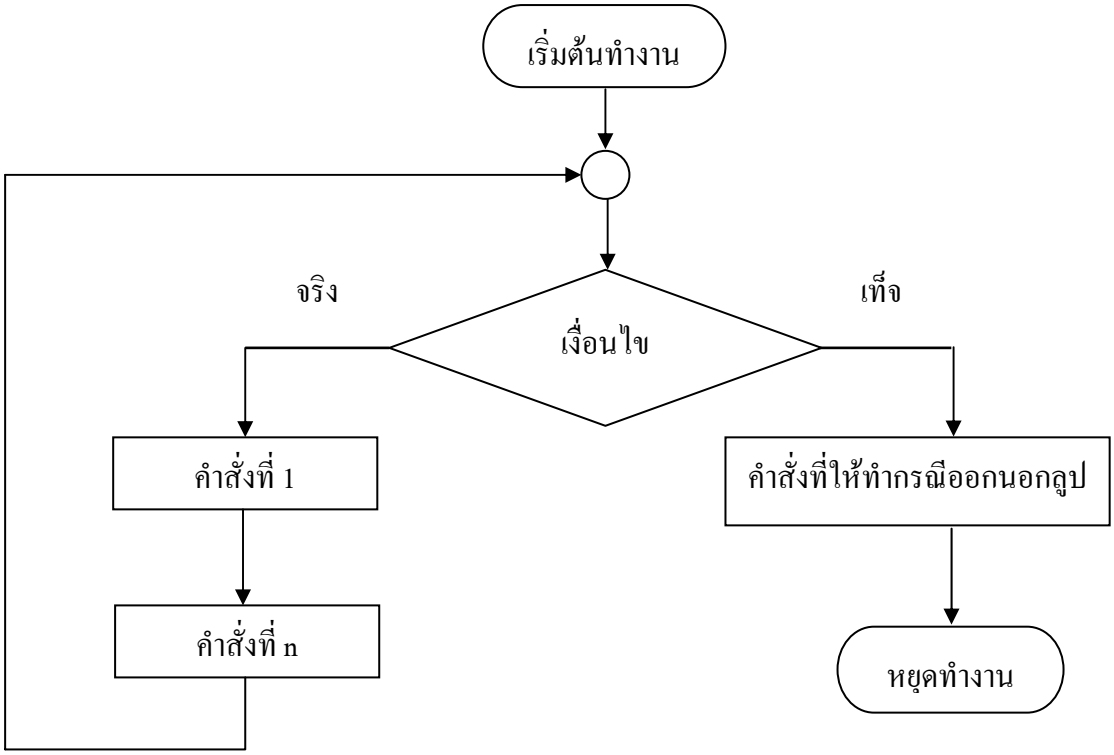
ภาพที่ 6.1 รูปแบบคำสั่งทำซ้ำแบบ while

จากภาพที่ 6.1 อธิบายได้ดังนี้

เงื่อนไข	หมายถึง นิพจน์ที่นำมาใช้ในการทดสอบ
คำสั่งที่ 1..คำสั่งที่ n	หมายถึง คำสั่งที่ให้ทำกรณีผลการทดสอบเงื่อนไขที่อยู่หลัง while มีค่าเป็นจริง

คำสั่งให้ทำในลำดับถัดไป หมายถึง คำสั่งที่ให้ทำกรณีเงื่อนไขที่นำมาทดสอบหลัง while มีค่าเป็นเท็จ หรือเมื่อทำคำสั่งกรณีที่เงื่อนไขเป็นจริงภายในลูปเรียบร้อยแล้ว

ผังงานรูปแบบคำสั่งทำซ้ำแบบ while มีรูปแบบดังนี้



ภาพที่ 6.2 ผังงานรูปแบบคำสั่งทำซ้ำแบบ while

จากรูปแบบและผังงานรูปแบบคำสั่งทำซ้ำแบบ while คำสั่งที่ให้ทำในกรณีเงื่อนไขเป็นจริง หรือคำสั่งภายในลูป อาจจะมีเพียงคำสั่งเดียวหรือหลายคำสั่งก็ได้ หากมีคำสั่งเดียว ไม่จำเป็นจะต้องเขียนคำสั่งนั้น ใส่ไว้ภายใต้เครื่องหมาย {} แต่หากมีคำสั่งที่ต้องการให้ทำหลายคำสั่ง จะต้องนำคำสั่งเหล่านั้นเขียนไว้ภายใต้เครื่องหมาย {}

ลักษณะสำคัญของคำสั่งทำซ้ำแบบ while

1. โปรแกรมจะทำการตรวจสอบเงื่อนไขก่อนทำคำสั่งในลูปเสมอ
2. โปรแกรมจะทำคำสั่งในลูปก็ต่อเมื่อเงื่อนไขที่ทำการทดสอบมีค่าเป็นจริงเท่านั้น
3. โปรแกรมอาจจะทำหรือไม่ทำคำสั่งในลูปก็ได้

ต่อไปนี้เป็นตัวอย่างการใช้คำสั่งทำซ้ำแบบ while

ตัวอย่างที่ 6.1 การเขียนโปรแกรมแสดงชื่อนักเรียน จำนวน 10 รอบ โดยใช้คำสั่ง while

```
1  #include<stdio.h>
2  #include<conio.h>
3  main()
4  {
5          clrscr();
6          int count = 1;
7          while(count <= 10){
8                  printf(”Miss Thanawan Kiriya\n”);
9                  count++;
10         }
11         getch();
12 }
```

จากตัวอย่างที่ 6.1 อธิบายการทำงานของโปรแกรมได้ดังนี้

บรรทัดที่ 6 ประกาศตัวแปร count มีชนิดเป็น int มีค่าเท่ากับ 1

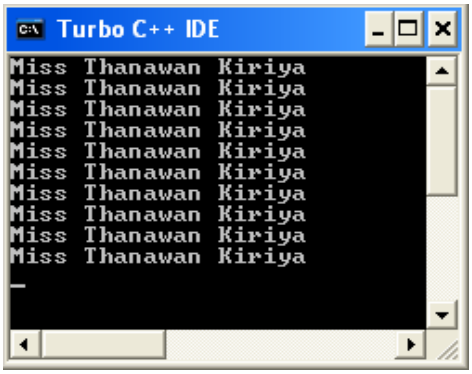
บรรทัดที่ 7 วนรอบทำซ้ำคำสั่งในบรรทัดที่ 8- 9 ตราบใดที่ค่าตัวแปร count น้อยกว่าหรือเท่ากับ 10 จริง แต่ถ้าหากค่าตัวแปร count มีค่ามากกว่า 10 ให้ทำคำสั่งในบรรทัดที่ 11

บรรทัดที่ 8 แสดงคำว่า Miss Thanawan Kiriya

บรรทัดที่ 9 เพิ่มค่าตัวแปร count อีก 1

บรรทัดที่ 11 รอรับการกดอักขระผ่านทางแป้นพิมพ์ 1 ครั้ง

จากตัวอย่างที่ 6.1 แสดงผลการรันโปรแกรมได้ดังนี้



ภาพ 6.3 ผลการรันโปรแกรมแสดงชื่อนักเรียน จำนวน 10 รอบ โดยใช้คำสั่ง while

ตัวอย่างที่ 6.2 การเขียนโปรแกรมทำซ้ำบวกเลข 1-10

```
1  #include<stdio.h>
2  #include<conio.h>
3  main(){
4          clrscr();
5          int num = 1, sum = 0;
6          while(num <= 10){
7                  sum += num;
8                  num++;
9          }
10         printf(”sum of 1-10 = %d\n”,sum);
11         getch();
12 }
```

จากตัวอย่างที่ 6.2 อธิบายการทำงานของโปรแกรมได้ดังนี้

บรรทัดที่ 5 ประกาศตัวแปร num และตัวแปร sum มีชนิดเป็น int มีค่าเท่ากับ 1 และ 0 ตามลำดับ

บรรทัดที่ 6 วนรอบทำซ้ำคำสั่งในบรรทัดที่ 7-8 ตราบใดที่ค่าตัวแปร num น้อยกว่าหรือเท่ากับ 10 แต่ถ้าหากค่าตัวแปร num มีค่ามากกว่า 10 ให้ทำคำสั่งในบรรทัดที่ 10-11

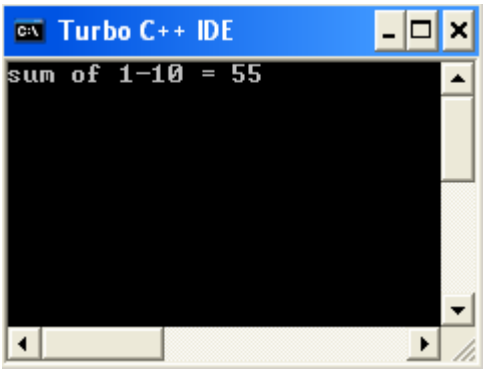
บรรทัดที่ 7 คำนวณค่าตัวแปร sum โดยนำค่าตัวแปร sum บวกด้วยค่าตัวแปร num นำผลลัพธ์เก็บไว้ในตัวแปร sum

บรรทัดที่ 9 เพิ่มค่าตัวแปร num อีก 1

บรรทัดที่ 10 แสดงคำว่า sum of 1-10 ตามด้วยค่าตัวแปร sum โดยตัวแปร sum มีชนิดเป็น int จึงใช้คู่กับ %d

บรรทัดที่ 11 รอรับการกดอักขระผ่านทางแป้นพิมพ์ 1 ครั้ง

จากตัวอย่างที่ 6.2 แสดงผลการรันโปรแกรมได้ดังนี้



ภาพ 6.4 ผลการรันโปรแกรมทำซ้ำบวกเลข 1-10

ตัวอย่างที่ 6.3 การเขียนโปรแกรมทำซ้ำบวกเลขคู่ระหว่าง 1-10

```
1  #include<stdio.h>
2  #include<conio.h>
3  main()
4  {
5          clrscr();
6          int num = 2, sum = 0;
7          while(num <= 10){
8                  sum += num;
9                  num += 2;
10         }
11         printf(”sum of even number 1-10 = %d\n”,sum);
12         getch();
13 }
```

จากตัวอย่างที่ 6.3 อธิบายการทำงานของโปรแกรมได้ดังนี้

บรรทัดที่ 6 ประกาศตัวแปร num และตัวแปร sum มีชนิดเป็น int มีค่าเท่ากับ 2 และ 0 ตามลำดับ

บรรทัดที่ 7 วนรอบทำซ้ำคำสั่งในบรรทัดที่ 8-9 ตราบใดที่ค่าตัวแปร num น้อยกว่าหรือเท่ากับ 10 แต่ถ้าหากค่าตัวแปร num มีค่ามากกว่า 10 ให้ทำคำสั่งในบรรทัดที่ 11-12

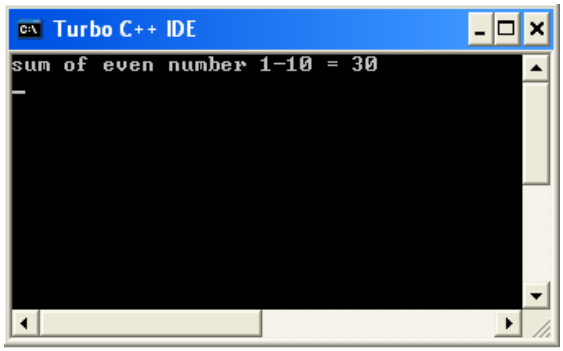
บรรทัดที่ 8 คำนวณค่าตัวแปร sum โดยนำค่าตัวแปร sum บวกด้วยค่าตัวแปร num นำผลลัพธ์เก็บไว้ในตัวแปร sum

บรรทัดที่ 9 เพิ่มค่าตัวแปร num อีก 2

บรรทัดที่ 11 แสดงคำว่า sum of even number 1-10 ตามด้วยค่าตัวแปร sum โดยตัวแปร sum มีชนิดเป็น int จึงใช้คู่กับ %d

บรรทัดที่ 12 รอรับการกดอักขระผ่านทางแป้นพิมพ์ 1 ตัว

จากตัวอย่างที่ 6.3 แสดงผลการรันโปรแกรมได้ดังนี้



ภาพที่ 6.5 ผลการรันโปรแกรมทำซ้ำบวกเลขคู่ระหว่าง 1-10

ตัวอย่างที่ 6.4 การเขียนโปรแกรมทำซ้ำแสดงเลขระหว่าง 1-10 ที่หารด้วย 2 ลงตัว

```
1  #include<stdio.h>
2  #include<conio.h>
3  main()
4  {
5          clrscr();
6          int num = 1;
7          while(num <= 10)
8          {
9                  if((num % 2) == 0)
10                         printf(”number is %d\n”,num);
11                 num++;
12         }
13         getch();
14 }
```

จากตัวอย่างที่ 6.4 อธิบายการทำงานของโปรแกรมได้ดังนี้

บรรทัดที่ 6 ประกาศตัวแปร num มีชนิดเป็น int มีค่าเท่ากับ 1

บรรทัดที่ 7 วนรอบทำซ้ำคำสั่งในบรรทัดที่ 9-11 ตราบใดที่ค่าตัวแปร num น้อยกว่าหรือเท่ากับ 10 แต่ถ้าหากค่าตัวแปร num มีค่ามากกว่า 10 ให้ทำคำสั่งในบรรทัดที่ 13

บรรทัดที่ 8 คำนวณค่าตัวแปร sum โดยนำค่าตัวแปร sum บวกด้วยค่าตัวแปร num นำผลลัพธ์เก็บไว้ในตัวแปร sum

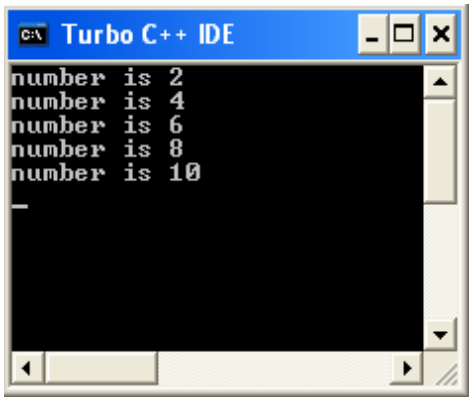
บรรทัดที่ 9 ถ้าค่าตัวแปร num หารด้วย 2 มีเศษเท่ากับ 0 ให้ทำคำสั่งในบรรทัดที่ 10

บรรทัดที่ 10 แสดงคำว่า number is ตามด้วยค่าตัวแปร num โดยตัวแปร num มีชนิดเป็น int จึงใช้คู่กับ %d

บรรทัดที่ 11 เพิ่มค่าตัวแปร num อีก 1

บรรทัดที่ 13 รอรับการกดอักขระผ่านทางแป้นพิมพ์ 1 ตัว

จากตัวอย่างที่ 6.4 แสดงผลการรันโปรแกรมได้ดังนี้



ภาพที่ 6.6 โปรแกรมทำซ้ำแสดงเลขระหว่าง 1-10 ที่หารด้วย 2 ลงตัว

ตัวอย่างที่ 6.5 การเขียนโปรแกรมทำซ้ำแสดงสูตรคูณของเลขที่ป้อนผ่านทางแป้นพิมพ์

```
1  #include<stdio.h>
2  #include<conio.h>
3  main()
4  {
5          clrscr();
6          int num, count = 1;
7          printf(”input integer number 2-12 : ”);
8          scanf(”%d”,&num);
9          while(count <= 12)
10         {
11                 printf(”%d  x %d = %d\n”,num,count,num*count);
12                 count++;
13         }
14         getch();
15 }
```

จากตัวอย่างที่ 6.5 อธิบายการทำงานของโปรแกรมได้ดังนี้

บรรทัดที่ 6 ประกาศตัวแปร num และตัวแปร count มีชนิดเป็น int มีค่าเท่ากับ 1

บรรทัดที่ 7 แสดงคำว่า input integer number 2-12 : ออกทางจอภาพ

บรรทัดที่ 8 รับค่าตัวแปร num ผ่านทางแป้นพิมพ์ โดยตัวแปร num มีชนิดเป็น int จึงใช้คู่กับ %d

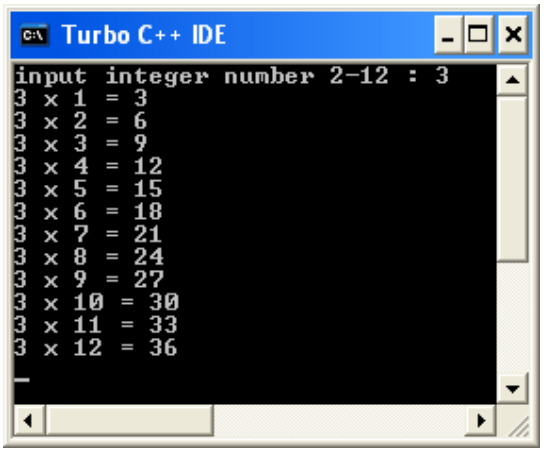
บรรทัดที่ 9 วนรอบทำซ้ำคำสั่งในบรรทัดที่ 11-12 ตราบใดที่ค่าตัวแปร count น้อยกว่าหรือเท่ากับ 12 แต่ถ้าหากค่าตัวแปร count มีค่ามากกว่า 12 ให้ทำคำสั่งในบรรทัดที่ 14

บรรทัดที่ 11 แสดงค่าตัวแปร num, เครื่องหมาย x, ค่าตัวแปร count, เครื่องหมาย = และตามด้วยค่าตัวแปร num คูณด้วยค่าตัวแปร count โดยตัวแปร num, count และผลคูณของตัวแปร num กับ count มีชนิดเป็น int จึงใช้คู่กับ %d

บรรทัดที่ 12 เพิ่มค่าตัวแปร count อีก 1

บรรทัดที่ 13 รอรับการกดอักขระผ่านทางแป้นพิมพ์ 1 ตัว

จากตัวอย่างที่ 6.5 แสดงผลการรันโปรแกรมได้ดังนี้



ภาพที่ 6.7 ผลการรันโปรแกรมทำซ้ำแสดงสูตรคูณของเลขที่ป้อนผ่านทางแป้นพิมพ์

ตัวอย่างที่ 6.6 การเขียนโปรแกรมทำซ้ำรับและแสดงเลขที่รับผ่านทางแป้นพิมพ์ไปเรื่อยๆ จนกว่าจะรับเลขน้อยกว่า 0

```
1  #include<stdio.h>
2  #include<conio.h>
3  main()
4  {
5          clrscr();
6          int num;
7          printf(”input integer number : ”);
8          scanf(”%d”,&num);
9          while(num >= 0)
10         {
11                 printf(”number is %d\n”,num);
12                 printf(”input integer number  : ”);
13                 scanf(”%d”,&num);
14         }
15         getch();
16 }
```

จากตัวอย่างที่ 6.6 อธิบายการทำงานของโปรแกรมได้ดังนี้

บรรทัดที่ 6 ประกาศตัวแปร num มีชนิดเป็น int

บรรทัดที่ 7 แสดงคำว่า input integer number : ออกทางจอภาพ

บรรทัดที่ 8 รับค่าตัวแปร num ผ่านทางแป้นพิมพ์ โดยตัวแปร num มีชนิดเป็น int จึงใช้คู่กับ %d

บรรทัดที่ 9 วนรอบทำซ้ำคำสั่งในบรรทัดที่ 11-13 ตราบใดที่ค่าตัวแปร num มากกว่าหรือเท่ากับ 0 แต่ถ้าหากค่าตัวแปร num มีค่ามากกว่า 0 ให้ทำคำสั่งในบรรทัดที่ 15

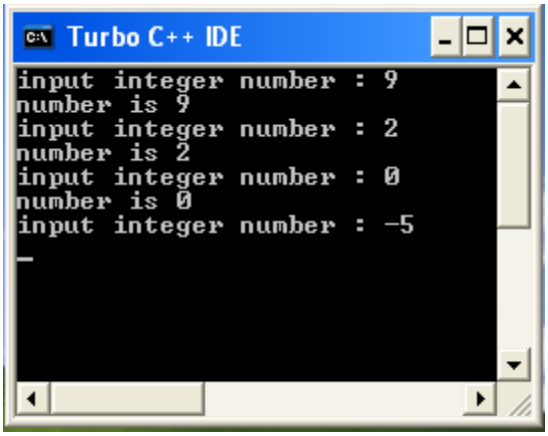
บรรทัดที่ 11 แสดงคำว่า number is ตามด้วยค่าตัวแปร num โดยตัวแปร num มีชนิดเป็น int จึงใช้คู่กับ %d

บรรทัดที่ 12 แสดงคำว่า input integer number : ออกทางจอภาพ

บรรทัดที่ 13 รับค่าตัวแปร num ผ่านทางแป้นพิมพ์ โดยตัวแปร num มีชนิดเป็น int จึงใช้คู่กับ %d

บรรทัดที่ 15 รอรับการกดอักขระผ่านทางแป้นพิมพ์ 1 ตัว

จากตัวอย่างที่ 6.6 อธิบายการทำงานของโปรแกรมได้ดังนี้



ภาพที่ 6.8 ผลการรันโปรแกรมทำซ้ำรับและแสดงเลขที่รับผ่านทางแป้นพิมพ์ไปเรื่อยๆ จนกว่าจะรับเลขน้อยกว่า 0

6.2 คำสั่งทำซ้ำแบบ do-while

เป็นคำสั่งโครงสร้างแบบทำซ้ำที่จะมีการทำคำสั่งในลูปก่อนทดสอบเงื่อนไข โดยหากเงื่อนไขที่อยู่หลัง while เป็นจริง จะทำคำสั่งในลูป และวนรอบตรวจสอบเงื่อนไขไปเรื่อยๆ จนกว่าเงื่อนไขที่อยู่หลัง while จะเป็นเท็จ คำสั่งทำซ้ำแบบ do-while มีรูปแบบการใช้คำสั่งดังนี้

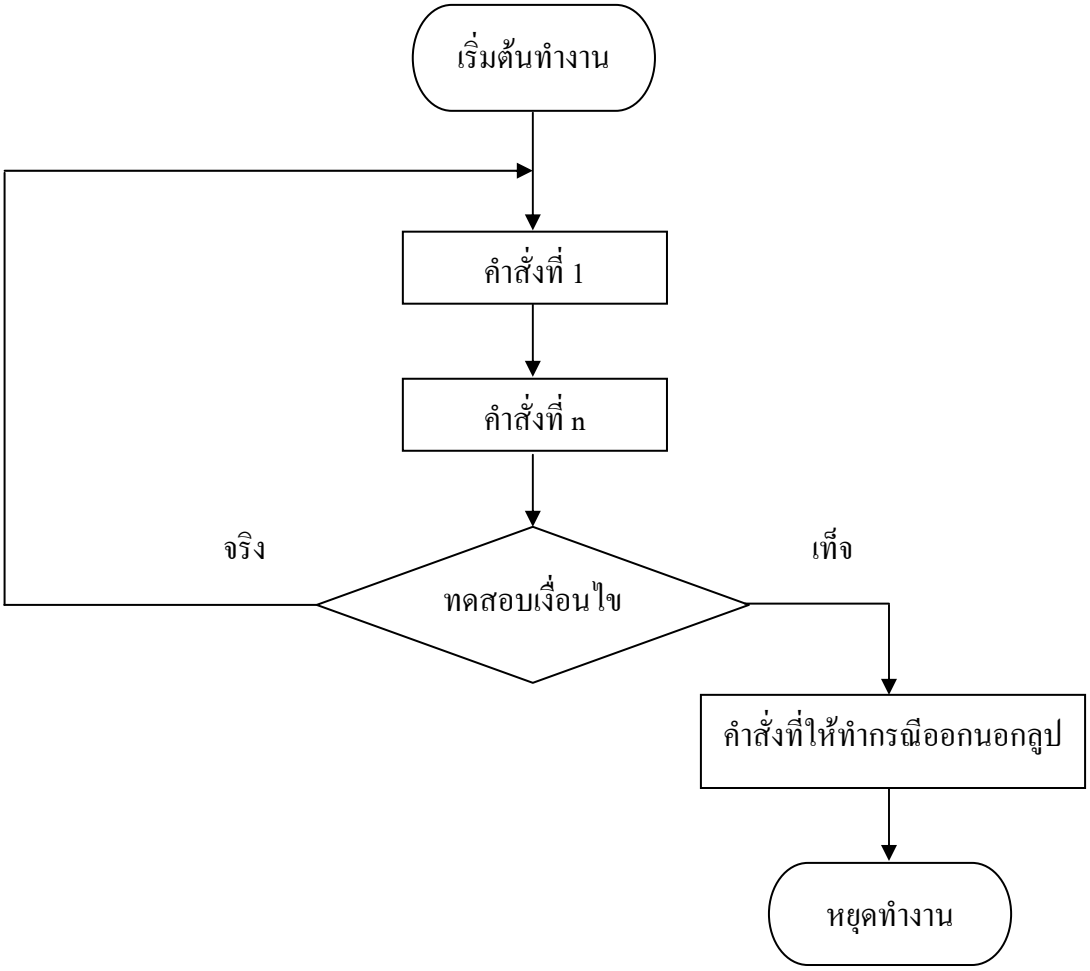
```
do
{
        คำสั่งที่ 1;
        …….
        คำสั่งที่ n;
}while(เงื่อนไข);
คำสั่งให้ทำในลำดับถัดไป;
```

ภาพที่ 6.9 รูปแบบคำสั่งทำซ้ำแบบ do-while

จากภาพที่ 6.9 อธิบายได้ดังนี้

คำสั่งที่ 1..คำสั่งที่ n	หมายถึง คำสั่งที่ให้ทำกรณีผลการทดสอบเงื่อนไขที่อยู่หลัง while มีค่าเป็นจริง
เงื่อนไข	หมายถึง นิพจน์ที่นำมาใช้ในการทดสอบ
คำสั่งให้ทำในลำดับถัดไป	หมายถึง คำสั่งที่ให้ทำกรณีเงื่อนไขที่นำมาทดสอบหลัง while มีค่าเป็นเท็จ หรือเมื่อทำคำสั่งกรณีที่เงื่อนไขเป็นจริงภายในลูปเรียบร้อยแล้ว

รูปแบบคำสั่งทำซ้ำแบบ do-while สามารถเขียนเป็นผังงานได้ดังนี้



ภาพที่ 6.10 ผังงานรูปแบบคำสั่งทำซ้ำแบบ do-while

ต่อไปนี้เป็นตัวอย่างการใช้คำสั่งทำซ้ำแบบ do-while

ตัวอย่างที่ 6.7 การเขียนโปรแกรมทำซ้ำแสดงชื่อนักเรียน จำนวน 10 รอบ

```
1  #include<stdio.h>
2  #include<conio.h>
3  main()
4  {
5          clrscr();
6          int count = 1;
7          do
8          {
9                  printf(”Miss Thanawan Kiriya\n”);
10                 count++;
11         }while(count <= 10);
12         getch();
13 }
```

จากตัวอย่างที่ 6.7 อธิบายการทำงานของโปรแกรมได้ดังนี้

บรรทัดที่ 6 ประกาศตัวแปร count มีชนิดเป็น int มีค่าเท่ากับ 1

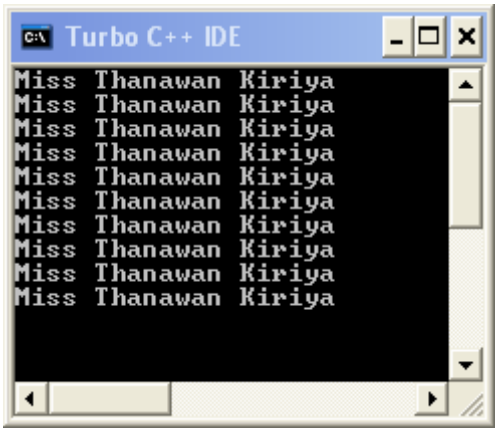
บรรทัดที่ 7,11 วนรอบทำซ้ำคำสั่งในบรรทัดที่ 9-10 ตราบใดที่ค่าตัวแปร count น้อยกว่าหรือเท่ากับ 10 หากค่าตัวแปร count มากกว่า 10 ให้ทำคำสั่งในบรรทัดที่ 12

บรรทัดที่ 9 แสดงคำว่า Miss Thanawan Kiriya ออกทางจอภาพ

บรรทัดที่ 10 เพิ่มค่าตัวแปร count อีก 1

บรรทัดที่ 12 รอรับการกดอักขระผ่านทางแป้นพิมพ์ 1 ตัว

จากตัวอย่างที่ 6.7 แสดงผลการรันโปรแกรมได้ดังนี้



ภาพที่ 6.11 ผลการรันโปรแกรมทำซ้ำแสดงชื่อนักเรียน จำนวน 10 รอบ

ตัวอย่างที่ 6.8 การเขียนโปรแกรมทำซ้ำบวกเลขคี่ระหว่าง 1-10

```
1  #include<stdio.h>
2  #include<conio.h>
3  main()
4  {
5          clrscr();
6          int num = 1, sum = 0;
7          do
8          {
9                  sum += num;
10                 num += 2;
11         }while(num <= 10);
12         printf("sum of odd number 1-10 = %d\n",sum);
13         getch();
14 }
```

จากตัวอย่างที่ 6.8 อธิบายการทำงานของโปรแกรมได้ดังนี้

บรรทัดที่ 6 ประกาศตัวแปร num และตัวแปร sum มีชนิดเป็น int มีค่าเท่ากับ 1 และ 0 ตามลำดับ

บรรทัดที่ 7,11 วนรอบทำซ้ำคำสั่งในบรรทัดที่ 9-10 ตราบใดที่ค่าตัวแปร num น้อยกว่าหรือเท่ากับ 10 หากค่าตัวแปร num มากกว่า 10 ให้ทำคำสั่งในบรรทัดที่ 12-13

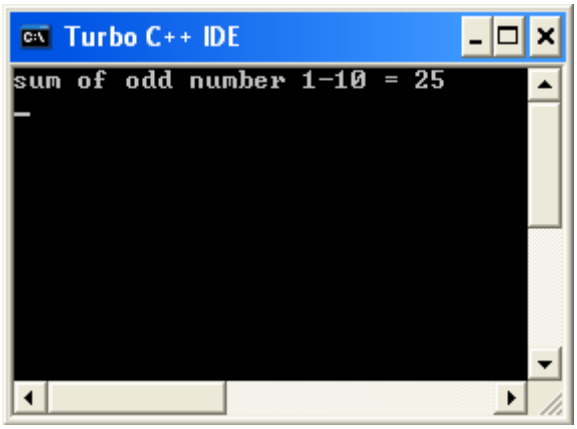
บรรทัดที่ 9 คำนวณค่าตัวแปร sum โดยนำค่าตัวแปร sum บวกด้วยค่าตัวแปร num นำผลลัพธ์เก็บไว้ในตัวแปร sum

บรรทัดที่ 10 เพิ่มค่าตัวแปร num อีก 2

บรรทัดที่ 12 แสดงคำว่า sum of odd number 1-10 = ตามด้วยค่าตัวแปร sum โดยตัวแปร sum มีชนิดเป็น int จึงใช้คู่กับ %d

บรรทัดที่ 13 รอรับการกดอักขระผ่านทางแป้นพิมพ์ 1 ตัว

จากตัวอย่างที่ 6.8 แสดงผลการรันโปรแกรมได้ดังนี้



ภาพที่ 6.12 ผลการรันโปรแกรมทำซ้ำบวกเลขคี่ระหว่าง 1-10

ตัวอย่างที่ 6.9 การเขียนโปรแกรมทำซ้ำหาค่าตัวแปร x,y,z และแสดงค่าตัวแปร x,y,z ออกทางจอภาพ

```
1  #include<stdio.h>
2  #include<conio.h>
3  main()
4  {
5          clrscr();
6          int x = 0, y = 0, z = 0;
7          do
8          {
9                  if(x > 0)
10                 {
11                         y = x + 2;
12                         z = x+y;
13                         x = z+3;
14                 }else{
15                         z = y+x;
16                         x += 2;
17                 }
18         }while(x <= 10);
19         printf("x = %d, y = %d, z = %d \n",x,y,z) ;
20         getch();
21 }
```

จากตัวอย่างที่ 6.9 อธิบายการทำงานของโปรแกรมได้ดังนี้

บรรทัดที่ 6 ประกาศตัวแปร x, y และ z มีชนิดเป็น int มีค่าเท่ากับ 0

บรรทัดที่ 7,18 วนรอบทำซ้ำคำสั่งในบรรทัดที่ 9-16 ตราบใดที่ค่าตัวแปร x น้อยกว่าหรือเท่ากับ 10 หากค่าตัวแปร x มากกว่า 10 ให้ทำคำสั่งในบรรทัดที่ 19-20

บรรทัดที่ 9 ถ้าค่าตัวแปร x มากกว่า 0 ให้ทำคำสั่งในบรรทัดที่ 11-13 มิฉะนั้นแล้ว ให้ทำคำสั่งในบรรทัดที่ 15-16

บรรทัดที่ 11 คำนวณค่าตัวแปร y โดยนำค่าตัวแปร x บวกด้วย 2 นำผลลัพธ์เก็บไว้ในตัวแปร y

บรรทัดที่ 12 คำนวณค่าตัวแปร z โดยนำค่าตัวแปร x บวกด้วยค่าตัวแปร y นำผลลัพธ์เก็บไว้ในตัวแปร z

บรรทัดที่ 13 คำนวณค่าตัวแปร x โดยนำค่าตัวแปร z บวกด้วย 3 นำผลลัพธ์เก็บไว้ในตัวแปร x

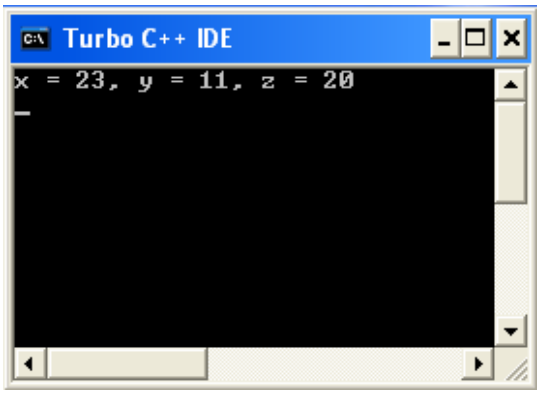
บรรทัดที่ 15 คำนวณค่าตัวแปร z โดยนำค่าตัวแปร y บวกด้วยค่าตัวแปร x นำผลลัพธ์เก็บไว้ในตัวแปร z

บรรทัดที่ 16 คำนวณค่าตัวแปร x โดยเพิ่มค่าตัวแปร x อีก 2 นำผลลัพธ์เก็บไว้ในตัวแปร x

บรรทัดที่ 19 แสดงคำว่า x = , y = , z = ตามด้วยค่าตัวแปร x, y และ z ตามลำดับ โดยค่าตัวแปร x, y และ z มีชนิดเป็น int จึงใช้คู่กับ %d

บรรทัดที่ 20 รอรับการกดอักขระผ่านทางแป้นพิมพ์ 1 ตัว

จากตัวอย่างที่ 6.9 แสดงผลการรันโปรแกรมได้ดังนี้



ภาพที่ 6.13 ผลการรันโปรแกรมทำซ้ำหาค่าตัวแปร x,y,z และแสดงค่าตัวแปร x,y,z ออกทางจอภาพ

ตัวอย่างที่ 6.10 การเขียนโปรแกรมทำซ้ำแสดงเลขจาก 10 จนถึง 1 โดยลดลงครั้งละ 1

```
1  #include<stdio.h>
2  #include<conio.h>
3  main()
4  {
5          clrscr();
6          int i = 10;
7          do
8          {
9                  printf("i = %d\n",i);
10                 i--;
11         }while(i  > 0);
12         getch();
13 }
```

จากตัวอย่างที่ 6.10 อธิบายการทำงานของโปรแกรมได้ดังนี้

บรรทัดที่ 6 ประกาศตัวแปร i มีชนิดเป็น int มีค่าเท่ากับ 10

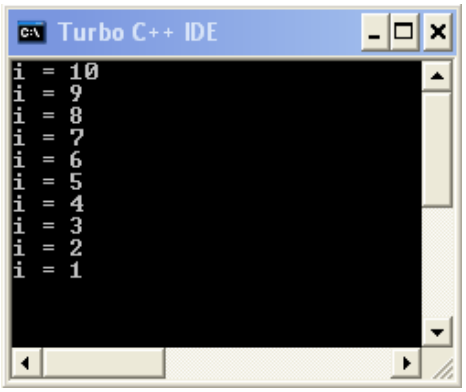
บรรทัดที่ 7,11 วนรอบทำซ้ำคำสั่งในบรรทัดที่ 9-10 ตราบใดที่ค่าตัวแปร i มากกว่า 0 หากค่าตัวแปร i เท่ากับ 0 ให้ทำคำสั่งในบรรทัดที่ 12

บรรทัดที่ 9 แสดงคำว่า i = ตามด้วยค่าตัวแปร i โดยตัวแปร i มีชนิดเป็น int จึงใช้คู่กับ %d

บรรทัดที่ 10 ลดค่าตัวแปร i ลงอีก 1

บรรทัดที่ 12 รอรับการกดอักขระผ่านทางแป้นพิมพ์ 1 ตัว

จากตัวอย่างที่ 6.10 แสดงผลการรันโปรแกรมได้ดังนี้



ภาพที่ 6.14 ผลการรันโปรแกรมทำซ้ำแสดงเลขจาก 10 จนถึง 1 โดยลดลงครั้งละ 1

6.3 คำสั่งทำซ้ำแบบ for

เป็นคำสั่งโครงสร้างแบบทำซ้ำที่จะมีการทดสอบเงื่อนไขก่อนทำคำสั่งในลูป โดยหากเงื่อนไข เป็นจริง จะทำคำสั่งในลูปไปเรื่อยๆ จนกว่าเงื่อนไขเป็นเท็จ จึงจะออกนอกลูป

โครงสร้างคำสั่งทำซ้ำแบบ for เหมาะสำหรับการทำซ้ำที่ทราบจำนวนรอบแน่นอน หากไม่ทราบจำนวนรอบที่แน่นอน ควรใช้คำสั่ง while หรือ do-while แทน คำสั่งทำซ้ำแบบ for มีรูปแบบการใช้คำสั่งดังนี้

```
for(กำหนดค่าตัวแปร; เงื่อนไข; การปรับเปลี่ยนค่าตัวแปร)
{
        คำสั่งที่ 1;
        .......
        คำสั่งที่ n;
}
คำสั่งให้ทำในลำดับถัดไป;
```

ภาพที่ 6.15 รูปแบบคำสั่งทำซ้ำแบบ for

จากภาพที่ 6.15 อธิบายได้ดังนี้

กำหนดค่าตัวแปร	หมายถึง การกำหนดค่าเริ่มต้นให้กับตัวแปรวนรอบ
เงื่อนไข	หมายถึง นิพจน์ที่นำมาใช้ในการทดสอบ

การปรับเปลี่ยนค่าตัวแปร	หมายถึง การเปลี่ยนแปลงค่าตัวแปรวนรอบ ซึ่งอาจจะเป็นการเพิ่มหรือลดค่าตัวแปรวนรอบ
คำสั่งที่ 1..คำสั่งที่ n	หมายถึง คำสั่งที่ให้ทำกรณีผลการทดสอบเงื่อนไขมีค่าเป็นจริง
คำสั่งให้ทำในลำดับถัดไป	หมายถึง คำสั่งที่ให้ทำกรณีผลการทดสอบเงื่อนไขมีค่าเป็นเท็จ หรือเมื่อทำคำสั่งกรณีเงื่อนไขเป็นจริงภายในลูปเรียบร้อยแล้ว

จากรูปแบบของคำสั่ง for มีลักษณะสำคัญดังนี้

1. ตัวแปรที่ใช้สำหรับวนรอบจะต้องเป็นตัวแปรแบบลำดับที่ เช่น int char long เป็นต้น

2. เงื่อนไข เป็นการทดสอบนิพจน์ ซึ่งจะส่งผลให้มีการวนรอบ หากผลการทดสอบเงื่อนไขเป็นจริง จะทำคำสั่งภายในลูป นั่นคือทำคำสั่งภายใต้ {} ซึ่งได้แก่ คำสั่งที่ 1 ถึงคำสั่งที่ n หากผลการทดสอบเงื่อนไขเป็นเท็จ จะออกนอกลูป นั่นคือทำ คำสั่งที่ให้ทำในลำดับถัดไป

3. ในการวนรอบต้องมีการปรับเปลี่ยนค่าตัวแปร ซึ่งอาจจะเป็นการเพิ่มหรือลดค่าตัวแปร ซึ่งจะต้องมีผลทำให้การทดสอบเงื่อนไขมีค่าเป็นจริงหรือเป็นเท็จ

ต่อไปนี้เป็นตัวอย่างการใช้คำสั่งทำซ้ำแบบ for

ตัวอย่างที่ 6.11 การเขียนโปรแกรมทำซ้ำแสดงชื่อนักเรียน จำนวน 10 รอบ

```
1  #include<stdio.h>
2  #include<conio.h>
3  main(){
4          clrscr();
5          int i
6          for(i = 1; i <= 10; i++)
7          {
8                  printf(”i = %d ”,i);
9                  printf(”Miss Thanawan  Kiriya\n”);
10         }
11         getch();
12 }
```

จากตัวอย่างที่ 6.11 อธิบายการทำงานของโปรแกรมได้ดังนี้

บรรทัดที่ 5 ประกาศตัวแปร i มีชนิดเป็น int

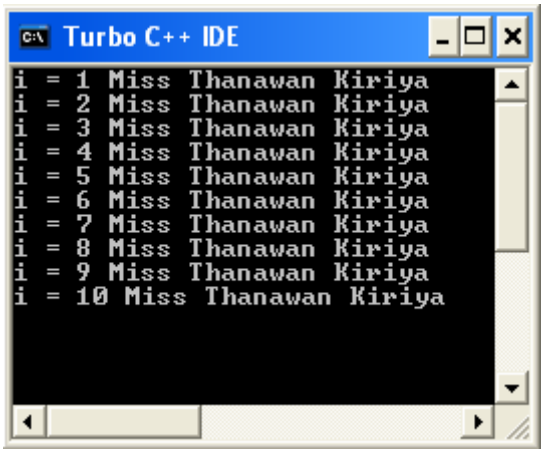
บรรทัดที่ 6 วนรอบทำซ้ำคำสั่งในบรรทัดที่ 8-9 ตราบใดที่ค่าตัวแปร i น้อยกว่าหรือเท่ากับ 10 ในแต่ละรอบเพิ่มค่าตัวแปร i อีก 1 หากค่าตัวแปร i มากกว่า 10 ให้ทำคำสั่งในบรรทัดที่ 11

บรรทัดที่ 8 แสดงคำว่า i = ตามด้วยค่าตัวแปร i โดยตัวแปร i มีชนิดเป็น int จึงใช้คู่กับ %d

บรรทัดที่ 9 แสดงคำว่า Miss Thanawan Kiriya ออกทางจอภาพ

บรรทัดที่ 11 รอรับการกดอักขระผ่านทางแป้นพิมพ์ 1 ตัว

จากตัวอย่างที่ 6.11 แสดงผลการรันโปรแกรมได้ดังนี้



ภาพที่ 6.16 ผลการรันโปรแกรมทำซ้ำแสดงชื่อนักเรียน จำนวน 10 รอบ

ตัวอย่างที่ 6.12 การเขียนโปรแกรมทำซ้ำแสดงตัวอักษรจาก A ถึง Z

```
1  #include<stdio.h>
2  #include<conio.h>
3  main()
4  {
5          clrscr();
6          for(char i = 'A'; i <= 'Z'; i++)
7          {
8                  printf("i = %c ",i);
9                  printf("\n");
10         }
11         getch();
12 }
```

จากตัวอย่างที่ 6.12 อธิบายการทำงานของโปรแกรมได้ดังนี้

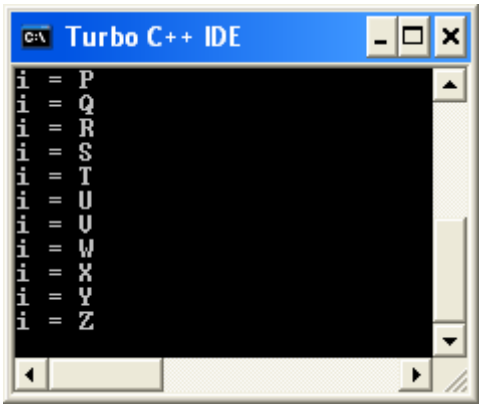
บรรทัดที่ 6 ประกาศตัวแปร i มีชนิดเป็น char มีค่าเท่ากับตัวอักษร A วนรอบทำซ้ำคำสั่งในบรรทัดที่ 8-9 ตราบใดที่ค่าตัวแปร i น้อยกว่าหรือเท่ากับตัวอักษร Z จากนั้นเลื่อนลำดับไปยังตัวอักษรถัดไปอีก 1 ลำดับ หากค่าตัวแปร i มีค่ามากกว่าตัวอักษร Z ให้ทำคำสั่งในบรรทัดที่ 11

บรรทัดที่ 8 แสดงคำว่า i = ตามด้วยค่าตัวแปร i โดยตัวแปร i มีชนิดเป็น char จึงใช้คู่กับ %c

บรรทัดที่ 9 เลื่อนเคอร์เซอร์ลงไปยังตำแหน่งเริ่มต้นของบรรทัดใหม่

บรรทัดที่ 11 รอรับการกดอักขระผ่านทางแป้นพิมพ์ 1 ตัว

จากตัวอย่างที่ 6.12 แสดงผลการรันโปรแกรมได้ดังนี้



ภาพที่ 6.17 ผลการรันโปรแกรมทำซ้ำแสดงตัวอักษรจาก A ถึง Z

ตัวอย่างที่ 6.13 การเขียนโปรแกรมทำซ้ำแสดงค่าอักขระของรหัสแอสกี้ตั้งแต่ 60 ถึง 100

```
1  #include<stdio.h>
2  #include<conio.h>
3  main()
4  {
5          clrscr();
6          for(int i = 60; i <= 100; i++)
7          {
8                  printf(”ascii code %d = %c ”,i,char(i));
9                  printf(”\n”);
10         }
11         getch();
12 }
```

จากตัวอย่างที่ 6.13 อธิบายการทำงานของโปรแกรมได้ดังนี้

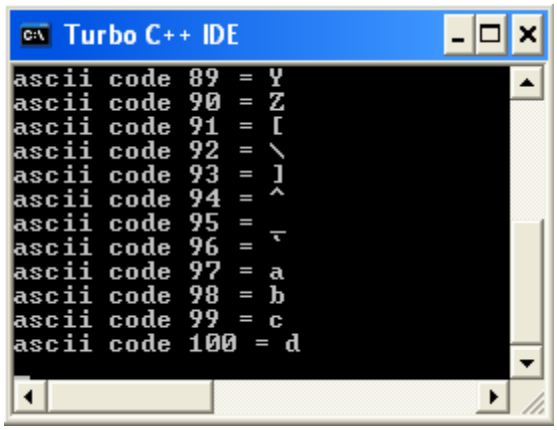
บรรทัดที่ 6 ประกาศตัวแปร i มีชนิดเป็น int มีค่าเท่ากับ 60 วนรอบทำซ้ำคำสั่งในบรรทัดที่ 8-9 ตราบใดที่ค่าตัวแปร i น้อยกว่าหรือเท่ากับ 100 ในแต่ละรอบเพิ่มค่าตัวแปร i อีก 1 หากค่าตัวแปร i มีค่ามากกว่า 100 ให้ทำคำสั่งในบรรทัดที่ 11

บรรทัดที่ 8 แสดงคำว่า asscii code ตามด้วยค่าตัวแปร i = และค่ารหัสแอสกี้ตัวแปร i โดยตัวแปร i มีชนิดเป็น int ส่วนรหัสแอสกี้ของตัวแปร i มีชนิดเป็น char จึงใช้คู่กับ %d และ %c ตามลำดับ

บรรทัดที่ 9 เลื่อนเคอร์เซอร์ลงไปยังตำแหน่งเริ่มต้นของบรรทัดใหม่

บรรทัดที่ 11 รอรับการกดอักขระผ่านทางแป้นพิมพ์ 1 ตัว

จากตัวอย่างที่ 6.13 แสดงผลการรันโปรแกรมได้ดังนี้



ภาพที่ 6.18 ผลการรันโปรแกรมทำซ้ำแสดงค่าอักขระของรหัสแอสกี้ตั้งแต่ 60 ถึง 100

6.4 คำสั่งทำซ้ำแบบ for ซ้อน

เป็นคำสั่งโครงสร้างแบบทำซ้ำแบบลูปซ้อนลูป โดยจะมีการทำงานเริ่มจากลูป for ที่อยู่ภายนอกก่อน จากนั้นจึงจะเข้าทำงานลูป for ที่อยู่ภายใน โดยทุกๆ ค่าของลูป for นอก จะทำซ้ำคำสั่งภายในลูป for ใน ให้ครบทุกรอบก่อน จึงจะออกมาทำลูป for นอก จำนวนรอบทั้งหมดที่ทำซ้ำ จะเท่ากับจำนวนรอบของลูป for นอกคูณกับจำนวนรอบของลูป for ใน เช่น หากจำนวนรอบของลูป for นอก โปรแกรมต้องการให้ทำซ้ำ จำนวน 5 รอบ ส่วนลูป for ใน โปรแกรมต้องการให้ทำซ้ำ จำนวน 3 รอบ จำนวนรอบที่โปรแกรมทำซ้ำมีทั้งสิ้น 15 รอบ คำสั่งทำซ้ำแบบ for ซ้อน มีรูปแบบการใช้คำสั่งดังนี้

```
for(กำหนดค่าตัวแปร; เงื่อนไข; การปรับเปลี่ยนค่าตัวแปร)
{
        คำสั่งที่ 1;
        for(กำหนดค่าตัวแปร; เงื่อนไข; การปรับเปลี่ยนค่าตัวแปร)
        {
                คำสั่งที่ 2;
                …….
                คำสั่งที่ n;
        }
}
คำสั่งให้ทำในลำดับถัดไป;
```

ภาพที่ 6.19 รูปแบบคำสั่งทำซ้ำแบบ for ซ้อน

จากภาพที่ 6.19 อธิบายได้ดังนี้

กำหนดค่าตัวแปร	หมายถึง การกำหนดค่าเริ่มต้นให้กับตัวแปรวนรอบ
เงื่อนไข	หมายถึง นิพจน์ที่นำมาใช้ในการทดสอบ
การปรับเปลี่ยนค่าตัวแปร	หมายถึง การเปลี่ยนแปลงค่าตัวแปรวนรอบ ซึ่งอาจจะเป็นการเพิ่มหรือลดค่าตัวแปรวนรอบ
คำสั่งที่ 1	หมายถึง คำสั่งที่ให้ทำกรณีเงื่อนไขของลูป for นอก มีค่าเป็นจริง

คำสั่งที่ 2..คำสั่งที่ n	หมายถึง คำสั่งที่ให้ทำกรณีผลการทดสอบเงื่อนไขของลูป for ใน มีค่าเป็นจริง
คำสั่งให้ทำในลำดับถัดไป	หมายถึง คำสั่งที่ให้ทำกรณีผลการทดสอบเงื่อนไขของลูป for นอก มีค่าเป็นเท็จ หรือเมื่อทำคำสั่งกรณีเงื่อนไขเป็นจริงภายในลูปเรียบร้อยแล้ว

ตัวอย่างที่ 6.14 การเขียนโปรแกรมทำซ้ำแสดงสตาร์ (*) โดยจำนวนสตาร์ขึ้นอยู่กับเลขที่รับผ่านทางแป้นพิมพ์ เช่น หากรับเลข 5 จะแสดงสตาร์ ดังนี้

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

```
1  #include<stdio.h>
2  #include<conio.h>
3  main()
4  {
5          clrscr();
6          int num,i,j;
7          printf(”input number : ”);
8          scanf(”%d”,&num);
9          for(i = 1; i <= num; i++)
10         {
11                 for(j = 1; j <= i; j++)
12                         printf(”*”);
13                 printf(”\n”);
14         }
15         getch();
16 }
```

จากตัวอย่างที่ 6.14 อธิบายการทำงานของโปรแกรมได้ดังนี้

บรรทัดที่ 6 ประกาศตัวแปร num, i และ j มีชนิดเป็น int

บรรทัดที่ 7 แสดงคำว่า input number : ออกทางจอภาพ

บรรทัดที่ 8 รับค่าตัวแปร num ผ่านทางแป้นพิมพ์ โดยตัวแปร num มีชนิดเป็น int จึงใช้คู่กับ %d

บรรทัดที่ 9 กำหนดให้ตัวแปร i มีค่าเท่ากับ 1 วนรอบทำซ้ำคำสั่งในบรรทัดที่ 11-13 ตราบใดที่ค่าตัวแปร i น้อยกว่าหรือเท่ากับค่าตัวแปร num ในแต่ละรอบเพิ่มค่าตัวแปร i อีก 1 หากค่าตัวแปร i มีค่ามากกว่าค่าตัวแปร num ให้ทำคำสั่งในบรรทัดที่ 15

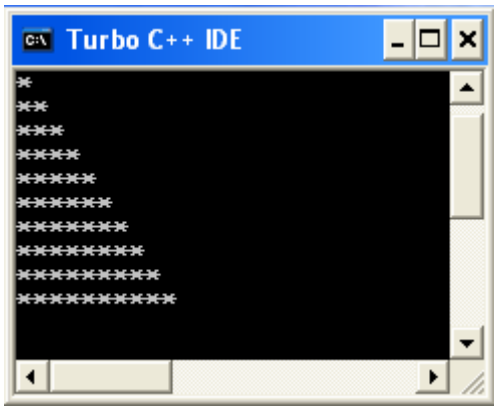
บรรทัดที่ 11 กำหนดให้ตัวแปร j มีค่าเท่ากับ 1 วนรอบทำซ้ำคำสั่งในบรรทัดที่ 12 ตราบใดที่ค่าตัวแปร j น้อยกว่าหรือเท่ากับค่าตัวแปร i (ค่าตัวแปร j จะปรับเปลี่ยนตามค่าตัวแปร i) ในแต่ละรอบเพิ่มค่าตัวแปร j อีก 1 หากค่าตัวแปร j มีค่ามากกว่าค่าตัวแปร i ให้ทำคำสั่งในบรรทัดที่ 13

บรรทัดที่ 12 แสดงเครื่องหมาย * (star)

บรรทัดที่ 13 เลื่อนเคอร์เซอร์ลงไปยังตำแหน่งเริ่มต้นของบรรทัดใหม่

บรรทัดที่ 15 รอรับการกดอักขระผ่านทางแป้นพิมพ์ 1 ตัว

จากตัวอย่างที่ 6.14 แสดงผลการรันโปรแกรมได้ดังนี้



ภาพ 6.20 ผลการรันโปรแกรมทำซ้ำแสดงสตาร์ (*) โดยจำนวนสตาร์ขึ้นอยู่กับเลขที่ป้อนผ่านทางแป้นพิมพ์

ตัวอย่างที่ 6.15 การเขียนโปรแกรมรับค่าตัวเลขจำนวนเต็ม และแสดงตัวเลขโดยเริ่มจากเลข 1 และเพิ่มขึ้นครั้งละ 1 จนถึงเลขที่ป้อนผ่านทางแป้นพิมพ์ เช่น หากป้อนเลข 5 จะแสดงผลดังนี้

```
1  #include<stdio.h>
2  #include<conio.h>
3  main()
4  {
5          clrscr();
6          int number,i,j;
7          printf(”input number : ”);
8          scanf(”%d”,&number);
9          for(i = 1; i <= number; i++)
10         {
11                 for(j = 1; j <= i; j++)
12                         printf(”%d  ”,j);
13                 printf(”\n”);
14         }
15         getch();
16 }
```

จากตัวอย่างที่ 6.15 อธิบายการทำงานของโปรแกรมได้ดังนี้

บรรทัดที่ 6 ประกาศตัวแปร number, i และ j มีชนิดเป็น int

บรรทัดที่ 7 แสดงคำว่า input number : ออกทางจอภาพ

บรรทัดที่ 8 รับค่าตัวแปร number ผ่านทางแป้นพิมพ์ โดยตัวแปร number มีชนิดเป็น int จึงใช้คู่กับ %d

บรรทัดที่ 9 กำหนดให้ตัวแปร i มีค่าเท่ากับ 1 วนรอบทำซ้ำคำสั่งในบรรทัดที่ 11-13 ตราบใดที่ค่าตัวแปร i น้อยกว่าหรือเท่ากับค่าตัวแปร number ในแต่ละรอบเพิ่มค่าตัวแปร i อีก 1 หากค่าตัวแปร i มีค่ามากกว่าค่าตัวแปร number ให้ทำคำสั่งในบรรทัดที่ 15

บรรทัดที่ 11 กำหนดให้ตัวแปร j มีค่าเท่ากับ 1 วนรอบทำซ้ำคำสั่งในบรรทัดที่ 12

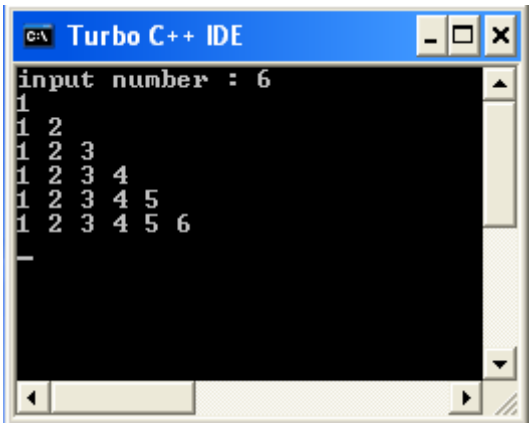
ตราบใดที่ค่าตัวแปร j น้อยกว่าหรือเท่ากับค่าตัวแปร i (ค่าตัวแปร j จะปรับเปลี่ยนตามค่าตัวแปร i) ในแต่ละรอบเพิ่มค่าตัวแปร j อีก 1 หากค่าตัวแปร j มีค่ามากกว่าค่าตัวแปร i ให้ทำคำสั่งในบรรทัดที่ 13

บรรทัดที่ 12 แสดงค่าตัวแปร j โดยตัวแปร j มีชนิดเป็น int จึงใช้คู่กับ %d

บรรทัดที่ 13 เลื่อนเคอร์เซอร์ลงไปยังตำแหน่งเริ่มต้นของบรรทัดใหม่

บรรทัดที่ 15 รอรับการกดอักขระผ่านทางแป้นพิมพ์ 1 ตัว

จากตัวอย่างที่ 6.15 แสดงผลการรันโปรแกรมได้ดังนี้



ภาพ 6.21 ผลการรันโปรแกรมรับค่าตัวเลขจำนวนเต็ม และแสดงตัวเลขโดยเริ่มจากเลข 1 และเพิ่มขึ้นครั้งละ 1 จนถึงเลขที่ป้อนผ่านทางแป้นพิมพ์

สรุปท้ายหน่วยการเรียน

การเขียนโปรแกรมโครงสร้างแบบทำซ้ำ เป็นการเขียนโปรแกรม ที่ให้ทำซ้ำหลายๆ ครั้ง ตามจำนวนรอบที่ผู้เขียนโปรแกรมต้องการ แทนการเขียนโปรแกรมโครงสร้างแบบลำดับ การเขียนโปรแกรมทำซ้ำ จะทำให้การเขียนโปรแกรมสั้นลง และประหยัดตัวแปร

ในการเขียนโปรแกรมโครงสร้างแบบทำซ้ำ มีคำสั่งที่สำคัญ 4 คำสั่ง ได้แก่ คำสั่ง while คำสั่ง do-while คำสั่ง for และคำสั่ง for ซ้อน โดยแต่ละคำสั่ง มีรูปแบบโครงสร้างแตกต่างกัน แต่ทุกคำสั่งจะให้ทำซ้ำคำสั่งในลูป เมื่อผลการทดสอบเงื่อนไขมีค่าเป็นจริง และจะออกนอกลูป เมื่อผลการทดสอบเงื่อนไขมีค่าเป็นเท็จ หากต้องการทำซ้ำโดยทราบจำนวนรอบที่แน่นอน ควรใช้คำสั่ง for แต่ถ้าไม่ทราบจำนวนรอบที่แน่นอน เช่น ให้ทำซ้ำจนกว่าจะกดเลข 0 หรือจนกว่าจะกดเลข -9 การทำซ้ำลักษณะนี้ไม่ทราบจำนวนรอบที่นอน ควรใช้คำสั่ง while หรือ do-while และหากต้องการให้โปรแกรมทำซ้ำแบบลูปซ้อน ควรใช้คำสั่ง for ซ้อน ในโปรแกรมหนึ่งๆ สามารถมีการใช้คำสั่งโครงสร้างแบบทำซ้ำแบบเดียวหรือหลายแบบก็ได้ ขึ้นอยู่กับความเหมาะสม

แหล่งข้อมูลเพิ่มเติม

http://www.rayongwit.ac.th/audchara/lesson.htm

http://itd.htc.ac.th/st_it51/it5144/npt/work/c/bc7.html

http://mistertun.exteen.com/20090520/entry