วิชา การพัฒนาโปรแกรม (ภาษาซี)

รหัสวิชา ว31293

หน่วยการเรียนรู้ที่ 5 คำสั่งควบคุมการทำงาน

สาระการเรียนรู้

1. คำสั่งควบคุมเงื่อนไข
2. การทำงานเป็นรอบ
3. คำสั่งประกอบการควบคุมทิศทาง

ผลการเรียนรู้

สามารถเลือกใช้ชุดคำสั่งควบคุมเงื่อนไขในการทำงาน การทำงานเป็นวนรอบ และคำสั่งประกอบการควบคุมทิศทางได้อย่างถูกต้องเหมาะสม

จุดประสงค์การเรียนรู้

1. เข้าใจรูปแบบการเขียนคำสั่ง if เพื่อควบคุมเงื่อนไข
2. สามารถเขียนชุดคำสั่ง if เพื่อตรวจสอบเงื่อนไขในรูปแบบต่าง ๆ ได้
3. สามารถนำคำสั่ง if และ swich ไปใช้งานได้อย่างเหมาะสม
4. เปรียบเทียบหลักการทำงานของลูป while , Do-While และ For ได้
5. สามารถนำลูปชนิดต่าง ๆ ไปประยุกต์ใช้งานได้อย่างถูกต้อง และเหมาะสม

ใบความรู้ที่ 6 เรื่อง การควบคุมการทำงาน

จัดทำโดย นางพรพนารัตน์ ชมภูนุช

โครงสร้างการควบคุมการทำงาน จะเป็นการควบคุมทิศทางการกระทำการ (Execution) ในโปรแกรมหรือฟังก์ชัน ซึ่งคำสั่งในภาษา C ที่ใช้ในการควบคุมการทำงานของโปรแกรม จะแบ่งออกเป็น 3 แบบ คือ

1) แบบลำดับ (Sequence)
2) แบบเลือกทำ (Selection)
3) แบบทำซ้ำ (Repetition)

ในการทำงานของคำสั่งหลายคำสั่งที่ต้องอาศัยผลการตรวจสอบเงื่อนไขก่อนว่าจริงหรือเท็จจึงจะปฏิบัติตามคำสั่ง ในการตรวจสอบเงื่อนไขดังกล่าวต้องอาศัยหลักการเปรียบเทียบและการหาค่าบูลีนเป็นสำคัญ ดังนี้

การเปรียบเทียบ เป็นการนำค่าหรือตัวแปรที่เก็บข้อมูลตั้งแต่ 2 ค่าขึ้นไปมาทำการเปรียบเทียบด้วยตัวดำเนินการทางตรรกศาสตร์ และนำไปใช้ในการเขียนโปรแกรมเพื่อควบคุมการทำงานของโปรแกรม ซึ่งผลลัพธ์ที่ได้จากการตรวจสอบเงื่อนไขนั้นจะได้ผลลัพธ์ 2 ค่า คือ จริง (True) กับ เท็จ (False) ซึ่งสัญลักษณ์หรือเครื่องหมายตัวดำเนินการที่ใช้ในการเปรียบเทียบเพื่อการตรวจสอบเงื่อนไขในภาษา C สามารถแสดงตัวอย่างการทำงานได้ดังนี้

ตัวอย่าง กำหนดค่าให้ x = 20 และ y = 10

ตัวดำเนินการ	ความหมาย	ตัวอย่าง	ผลลัพธ์
<	น้อยกว่า	x < y	เท็จ
<=	น้อยกว่าเท่ากับ	x<= y	เท็จ
>	มากกว่า	x> y	จริง
>=	มากกว่าเท่ากับ	x >= y	จริง
= =	เท่ากับ	x == y	เท็จ
!=	ไม่เท่ากับ	x != y	จริง

- **การตัดสินใจ หรือการหาค่าบูลีน** เป็นวิธีการหาค่าจากเครื่องหมายหรือสัญลักษณ์ทางบูลีน เพื่อให้ได้มาซึ่งผลลัพธ์ โดยผลลัพธ์จะมีอยู่ 2 ค่า คือ True กับ False เครื่องหมายทางบูลีนมีหลายตัว แต่ในที่นำมาใช้ในโปรแกรมบ่อย ๆ ก็คือ and , or , not และ xor สามารถแสดงตัวอย่างการทำงานได้ดังนี้

x	y	x and y	x or y	not x	x xor y
True	True	True	True	False	False
True	False	False	True	False	True
False	True	False	True	True	True
False	False	False	False	True	False

การควบคุมการทำงานแบบลำดับ

จะมีลักษณะการกระทำงาน (Executeion) จากข้อความคำสั่งแรกไปตามลำดับ จากบนลงล่างจนถึงข้อความคำสั่งสุดท้าย โดยปิดล้อมด้วยเครื่องหมาย { } เพื่อกำหนดจุดเริ่มต้นของลำดับการทำงานแบบเรียงลำดับ ดังตัวอย่าง

ตัวอย่าง

```
{
        Statement 1;
        Statement 2;
              :
              :
        Statement n;
}
```

การควบคุมการทำงานแบบเลือกทำ

ภาษา C จะใช้ประโยค if ในการสร้างเงื่อนไข ซึ่งสามารถตรวจสอบเงื่อนไขว่าเป็น จริง หรือ เท็จ ได้ นอกจากประโยค if แล้วในภาษา C ยังมีการกำหนดทางเลือกด้วยประโยค switch ให้เลือกใช้อีกด้วย

1. การควบคุมเงื่อนไขด้วยคำสั่ง if

ในการเขียนโปรแกรมจะมีบางขั้นตอนที่จะต้องเลือกการทำงานอย่างใดอย่างหนึ่ง ดังนั้นจะต้องอาศัยตัวแปรและค่าของตัวแปรในการระบุเงื่อนไขเพื่อกำหนดให้เป็นทางเลือก ถ้าค่าของตัวแปรตรงกับเงื่อนไขที่กำหนด ให้โปรแกรมทำงานอย่างใดอย่างหนึ่ง ขึ้นอยู่กับผู้เขียนโปรแกรม การกำหนดทางเลือกของคำสั่ง if จะมีอยู่ 4 รูปแบบ คือ

1.1 การสร้างเงื่อนไข if แบบกรณีเดียว

1.2 การสร้างเงื่อนไข if แบบ 2 กรณี

1.3 การสร้างเงื่อนไข if แบบหลายกรณี

1.4 การสร้างเงื่อนไข if แบบ Nested if

1.1 การสร้างเงื่อนไขแบบกรณีเดียว

เป็นการกำหนดทางเลือกให้มีการทำงาน ในกรณีที่เงื่อนไขเป็นจริงเท่านั้น โดยโปรแกรมจะทำงานตามคำสั่งที่อยู่ในบรรทัดถัดมา หรือถ้าคำสั่งที่ต้องการให้โปรแกรมทำงานมีมากกว่า 1 คำสั่ง ต้องกำหนดคำสั่งไว้ในเครื่องหมาย { } โดยมีรูปแบบการใช้คำสั่งดังนี้

รูปแบบที่ 1 กรณีที่มีคำสั่งเพียงคำสั่งเดียว

```
If (condition)
    Statement;
```

รูปแบบที่ 2 กรณีที่มีหลายคำสั่ง

```
If (condition)
    {
    statement;
    statement;
        ...
    }
```

ตัวอย่าง

```
If  (score >= 80)
    printf ("Very Good");
```

หรือ

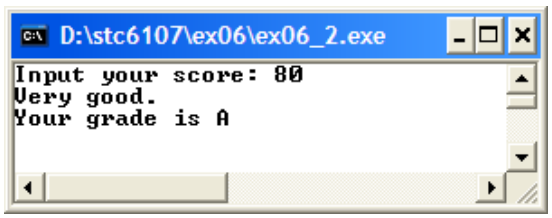
```
If  (score >= 80)
    {
    printf (“Very Good ”);
    printf (“Your grade is A”);
    }
```

กิจกรรมเสริมทักษะ

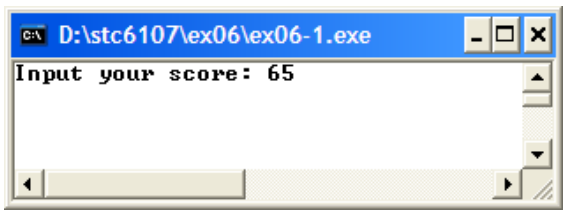
ให้นักเรียนทดลองเขียนโปรแกรมตามตัวอย่างด้านล่าง แล้วบันทึกข้อมูลไว้ในโฟลเดอร์งานที่ชื่อว่า ex06 แล้วทดลองใช้งานโปรแกรมแล้วสังเกตผลลัพธ์ที่ได้

```
1  #include <stdio.h>
2  #include <conio.h>
3  int  score;
4  main( )
5  {
6     printf(“Input your score : ”);
7     scanf(“%d”, &score);
8     if (score>=80)
9     {
10    printf (“Very Good. \n”);
11    printf (“Your grade is A.”);
12    }
13    getch();
14 }
15
```

กรณีเงื่อนไขเป็นจริง



กรณีเงื่อนไขเป็นเท็จ



รูปภาพ แสดงผลลัพธ์ที่ได้จากการทำงานของโปรแกรม ex06_1.exe

1.2 การสร้างเงื่อนไขแบบ 2 กรณี (if…else)

หากต้องการกำหนดทางเลือกให้มีการทำงานทั้ง 2 กรณี คือ กรณีที่เป็นจริง และเป็นเท็จ คำสั่งที่ใช้คือ if…else นั่นก็คือ กรณีที่เงื่อนไขเป็นจริง โปรแกรมจะทำงานหลังคำสั่ง if แต่ถ้าเงื่อนไขเป็นเท็จโปรแกรมจะทำงานหลังคำสั่ง else ซึ่งมีรูปแบบการใช้คำสั่งดังนี้

รูปแบบที่ 1 กรณีที่มีคำสั่งเพียงคำสั่งเดียว

```
if (condition)
    statement
else
    statement
```

รูปแบบที่ 2 กรณีที่มีหลายคำสั่ง

```
if (condition)
    {
    statement-1
    statement-2
        ...
    statement-n
    }
    else
    {
    statement-1
    statement-2
        ...
    statement-n
    }
```

ตัวอย่าง

```
if  (score >= 80)
    printf ("Very Good");
else
    printf ("Sorry...");
```

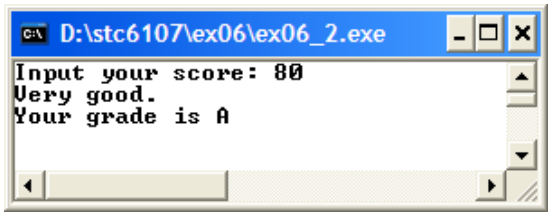
หรือ

```
if  (score >= 80)
    {
    printf ("Very Good");
    printf ("Your grade is A");
    }
else
    printf ("Sorry...");
```

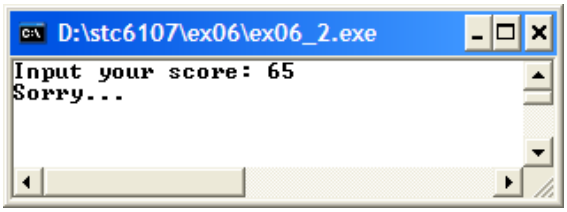
ให้นักเรียนทดลองเขียนโปรแกรมตามตัวอย่างด้านล่าง แล้วบันทึกข้อมูลไว้ในโฟลเดอร์งานที่ชื่อว่า ex06 แล้วทดลองใช้งานโปรแกรมแล้วสังเกตผลลัพธ์ที่ได้

```
1  #include <stdio.h>
2  #include <conio.h>
3  int  score;
4  main( )
5  {
6     printf("Input your score : ");
7     scanf("%d", &score);
8     if (score>=80)
9       {
10       printf ("Very Good. \n");
11       printf ("Your grade is A");
12       }
13    else
14      printf ("Sorry...");
15    getch( );
   }
```

กรณีเงื่อนไขเป็นจริง



กรณีเงื่อนไขเป็นเท็จ



รูปภาพ แสดงผลลัพธ์ที่ได้จากการทำงานของโปรแกรม ex06_1.exe

1.3 การสร้างเงื่อนไข if แบบหลายกรณี

จากรูปแบบข้างต้น เป็นรูปแบบเงื่อนไขแค่ 2 กรณีเท่านั้น ดังนั้น หากรูปแบบการสร้างเงื่อนไขที่ต้องตรวจสอบหลาย ๆ กรณี สามารถทำได้โดยใช้คำสั่ง else if ซึ่งมีรูปแบบการใช้คำสั่งดังนี้

รูปแบบ

```
If (condition)
    statement
else if (condition)
    statement
else if (condition)
    statement
else
    statement
```

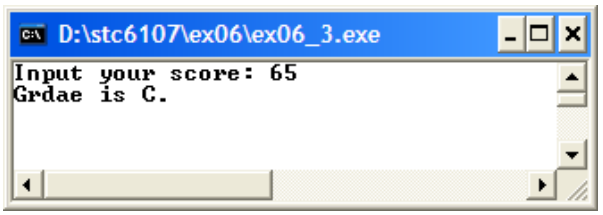
ตัวอย่าง

```
printf("Input your score : ");
scanf("%d", &score);
if  (score >= 80)
    printf ("Grdae is A.");
else  if  (score >= 70)
    printf ("Grdae is B.");
else  if  (score >= 60)
    printf ("Grdae is C.");
else  if  (score >= 50)
    printf ("Grdae is D.");
else
    printf ("Grdae is F.");
```

กิจกรรมเสริมทักษะ

ให้นักเรียนทดลองเขียนโปรแกรมตามตัวอย่างด้านล่าง แล้วบันทึกข้อมูลไว้ในโฟลเดอร์งานที่ชื่อว่า ex06 แล้วทดลองใช้งานโปรแกรมแล้วสังเกตผลลัพธ์ที่ได้

```
1  #include <stdio.h>
2  #include <conio.h>
3  int  score;
4  main( )
5  {
6     printf("Input your score: ");
7     scanf("%d", &score);
8     if   (score >= 80)
9        printf ("Grdae is A.");
10    else  if   (score >= 70)
11       printf ("Grdae is B.");
12    else  if   (score >= 60)
13       printf ("Grdae is C.");
14    else  if   (score >= 50)
15       printf ("Grdae is D.");
16    else
17       printf ("Grdae is F.");
18    getch();
19 }
```



รูปภาพ แสดงผลลัพธ์ที่ได้จากการทำงาน ของโปรแกรม ex06_3.exe

1.4 การสร้างเงื่อนไขแบบซ้อน (Nested if)

เป็นรูปแบบการสร้างเงื่อนไขที่ซับซ้อนยิ่งขึ้น โดยจะมีการตรวจสอบเงื่อนไขซ้อยย่อยลงไปอีก ซึ่งการสร้างประโยคซ้อนเงื่อนไขดังกล่าว จำเป็นต้องตรวจสอบให้รอบคอบ มิฉะนั้นอาจเกิดผลผิดพลาดได้

รูปแบบ

```
if (condition)
    if (condition)
        statement
    else
        statement
else
    if (condition)
        statement
    else
        statement
```

ตัวอย่าง

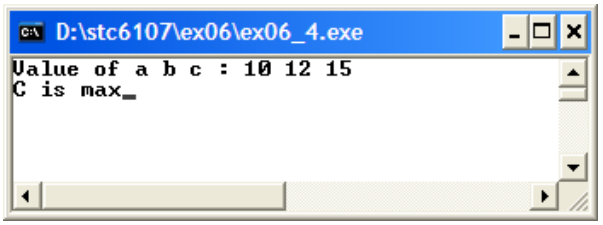
การตรวจสอบค่า a , b และ c ว่าค่าใดมีค่ามากที่สุด (เป็นการตรวจสอบทีละค่าแบบ Nested if)

```
if (a>b)
    if (a>c)
        printf ("A is max");
    else
        printf ("C is max");
else
    if (b>c)
        printf ("B is max");
    else
        printf ("C is max");
```

กิจกรรมเสริมทักษะ

ให้นักเรียนทดลองเขียนโปรแกรมตามตัวอย่างด้านล่าง แล้วบันทึกข้อมูลไว้ในโฟลเดอร์งานที่ชื่อว่า ex06 แล้วทดลองใช้งานโปรแกรมแล้วสังเกตผลลัพธ์ที่ได้

```
1  #include <stdio.h>
2  #include <conio.h>
3  int  a,b,c;
4  main( )
5  {
6     printf("Value of  a  b  c : ");
7     scanf("%d %d %d", &a,&b,&c);
8        if  (a>c)
9              printf ("A is max");
10       else
11             printf ("C is max");
12    else
13       if  (b>c)
14             printf ("B is max");
15       else
16             printf ("C is max");
17 getch();
18 }
```



รูปภาพ แสดงผลลัพธ์ที่ได้จากการทำงานของโปรแกรม ex06_4.exe

2. การควบคุมเงื่อนไขด้วยคำสั่ง switch

นอกจาก if-else แล้ว ภาษา C ยังมีคำสั่งควบคุมเงื่อนไขอีกตัวหนึ่งคือ switch ซึ่งสามารถนำมาใช้งานได้ดีกับโปรแกรมที่มีรายการเมนูให้เลือก โดยมีรูปแบบการใช้งานดังนี้

รูปแบบ

switch *(variable/expression)*
case (*constant 1) :*
statement;
break;
case (*constant 2) :*
statement;
break;
...

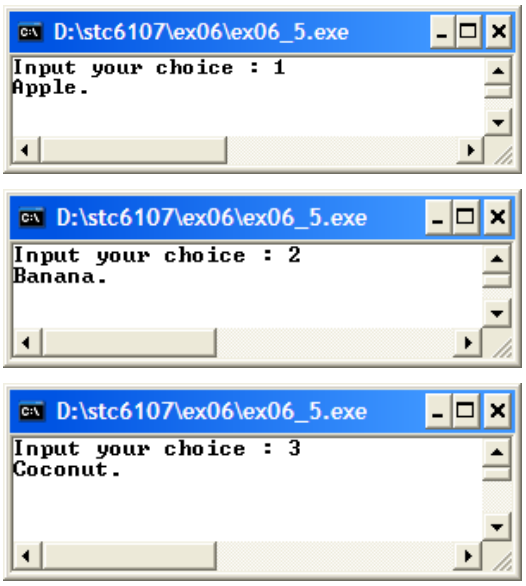
ตัวอย่าง

```
switch (choice)
case '1' : printf ("\nApple.");
      break;
case '2' : printf ("\nBanana.");
      break;
case '3' : printf ("\nCoconut.");
      break;
defalt :
      printf ("\n Please type (1-3) only ..");
```

กิจกรรมเสริมทักษะ

ให้นักเรียนทดลองเขียนโปรแกรมตามตัวอย่างด้านล่าง แล้วบันทึกข้อมูลไว้ในโฟลเดอร์งานที่ชื่อว่า ex06 แล้วทดลองใช้งานโปรแกรมแล้วสังเกตผลลัพธ์ที่ได้

```
1  #include <stdio.h>
2  #include <conio.h>
3  char choice;
4  main( )
5  {
6     printf("Input your choice : ");
7     choice=getche();
8     switch (choice)
9     {
10       case '1' : printf ("Apple.");
11       break;
12       case '2' :printf ("Banana.");
13       break;
14       case '3c' :printf ("Coconut.");
15       break;
16       default :
17       printf ("\n Please type (1-3) only ..");
18       {
      getch();
   }
```



รูปภาพ แสดงผลลัพธ์ที่ได้จากการทำงานของโปรแกรม ex06_5.exe

การทำงานเป็นรอบ

เราสามารถสั่งให้คอมพิวเตอร์ประมวลผลชุดคำสั่งซ้ำ ๆได้ เรียกว่า **กระบวนการทำซ้ำ** หรือ **ลูป (Loop)** เช่น สร้างลูปเพื่อประมวลผลอ่านไฟล์ข้อมูลจนกระทั่งจบไฟล์ , สร้างลูปเพื่อการคำนวณจนครบรอบ หรือสร้างลูปของรายการข้อมูลเพื่อให้ผู้ใช้สามารถใช้งานโปรแกรมไปได้เรื่อย ๆ จนกว่าจะเลือกรายการจบการทำงาน ดังนั้น จะพบว่าจำนวนรอบการทำงานของลูปที่สร้างขึ้นนั้น ขึ้นอยู่กับเงื่อนไขที่กำหนดขึ้นเป็นสำคัญ ในภาษา C มีชุดคำสั่งทำงานเป็นรอบอยู่ 3 ประเภท ด้วยกันดังนี้

1. การทำงานเป็นรอบด้วยลูป while
2. การทำงานเป็นรอบด้วยลูป do – while
3. การทำงานเป็นรอบด้วยลูป for

1. การทำงานเป็นรอบด้วยลูป while

คำสั่งวนลูปแบบ while จะเริ่มต้นทำงานจากการตรวจสอบเงื่อนไข เงื่อนไขเป็นจริงจึงจะทำงานตามคำสั่งของ while เมื่อทำงานเสร็จแล้วก็จะวนกลับไปตรวจสอบเงื่อนไขใหม่เป็นเช่นนี้ไปเรื่อย ๆ จนกว่าเงื่อนไขจะเป็นเท็จจึงจะออกจากการทำงานของลูป while รูปแบบการเขียนคำสั่ง while มีดังนี้

รูปแบบ

```
while (condition)
{
    statement-1;
    statement-2;
      ...
    statement-n;
}
```

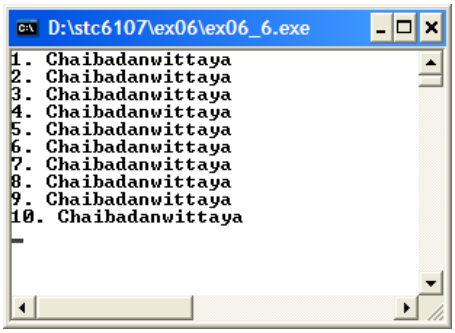
ตัวอย่าง

```
while ( count <= 10)
{
printf("Chaibadanwittaya \n");
count++;
}
```

กิจกรรมเสริมทักษะ

ให้นักเรียนทดลองเขียนโปรแกรมตามตัวอย่างด้านล่าง แล้วบันทึกข้อมูลไว้ในโฟลเดอร์งานที่ชื่อว่า ex06 แล้วทดลองใช้งานโปรแกรมแล้วสังเกตผลลัพธ์ที่ได้

```
1  #include <stdio.h>
2  #include <conio.h>
3  main()
4  {
5  int count=1;
6  while (count<=10)
7    {
8    printf("%d. Chaibadanwittaya \n",count);
9    count++;
10   }
11   getch();}
```



รูปภาพ แสดงผลลัพธ์ที่ได้จากการทำงานของโปรแกรม ex06_6.exe

2. การทำงานเป็นรอบด้วยลูป Do - while

คำสั่งวนลูปแบบ do-while มีลักษณะการทำงานที่แตกต่างจากลูป while ตรงที่ว่าจะทำงานตามคำสั่งของ do ก่อนหนึ่งรอบแล้วจึงตรวจสอบเงื่อนไขที่คำสั่ง while ถ้าเงื่อนไขเป็นจริงแล้วจึงจะวนไปทำงานตามคำสั่งของ do อีกครั้ง แล้วกลับมาตรวจสอบเงื่อนไขใหม่ เป็นอย่างนี้ไปจนกว่าผลการตรวจสอบเงื่อนไขจะเป็นเท็จ จึงจะเลิกทำงาน ออกจากลูป do-while รูปแบบการเขียนคำสั่งลูป do-while มีดังนี้

รูปแบบ

```
do
{
    statement-1;
    statement-2;
      ...
    statement-n;
}
while (condition)
```

ตัวอย่าง

```
do
{
printf("Chaibadanwittaya School \n");
count++;
}
while ( count >= 10)
```

หมายความว่า จะวนลูปซ้ำในขณะที่ count มีค่ามากกว่าเท่ากับ 10 ซึ่งความจริงแล้ว count มีค่าเริ่มต้น น้อยกว่า 10

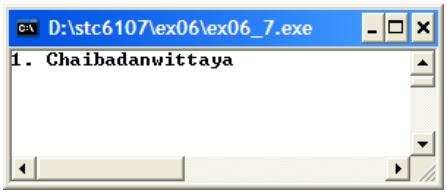
กิจกรรมเสริมทักษะ

ให้นักเรียนทดลองเขียนโปรแกรมตามตัวอย่างด้านล่าง แล้วบันทึกข้อมูลไว้ในโฟลเดอร์งานที่ชื่อว่า ex06_7 แล้วทดลองใช้งานโปรแกรมแล้วสังเกตผลลัพธ์ที่ได้

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

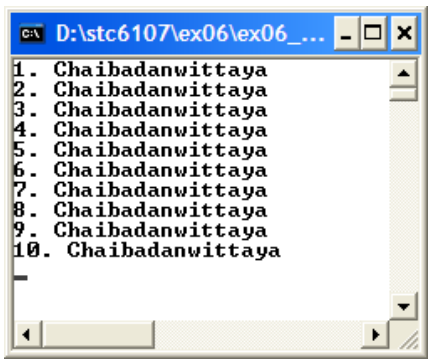
กรณี เงื่อนไข while (*count>=10*);

ผลการตรวจสอบเป็น **เท็จ**



กรณี เงื่อนไข while (*count<=10*);

ผลการตรวจสอบเป็น **จริง**



รูปภาพ แสดงผลลัพธ์ที่ได้จากการทำงานของโปรแกรม ex06_7.exe

3. การทำงานเป็นรอบด้วยลูป for

คำสั่งวนลูปแบบ for เป็นคำสั่งใช้สำหรับการควบคุมทิศทางของโปรแกรมให้ทำงานแบบวนซ้ำ เช่นเดียวกับ while และ do-while แต่คำสั่ง for มีลักษณะพิเศษกว่าคำสั่งลูปแบบอื่น ๆ ตรงที่คำสั่ง for เหมาะกับกรณีที่รู้จำนวนแน่นอนแล้วว่า ต้องการให้วนลูปทำงานกี่รอบ รูปแบบการเรียกใช้คำสั่ง for มีดังนี้

รูปแบบ

```
for (initiailzat;  condition;  increment )
{
      statement-1;
      statement-2;
         ...
      statement-n;
}
```

initialzat หมายถึง เงื่อนไขหรือตัวแปรที่ให้ค่าเริ่มต้นของการทำงานแบบวนรอบ
condition หมายถึง เงื่อนไขที่ใช้ในการวนซ้ำโดยถ้าเงื่อนไขเป็นจริง จะได้ทำงานวนซ้ำ
increment หมายถึง การเพิ่มค่าขึ้นของตัวแปร (หรือเป็นแบบลดลงก็ได้)
statement หมายถึง คำสั่งใด ๆ ที่จะถูกวนซ้ำทำงานในแต่ละรอบ ถ้ามีมากกว่า 1 คำสั่ง ให้คำสั่งเหล่านั้นอยู่ในเครื่องหมาย { }

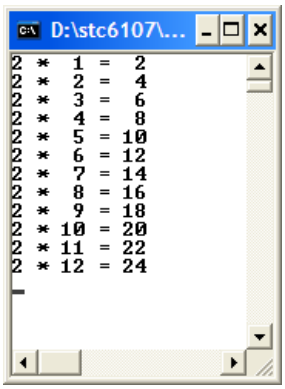
ตัวอย่าง

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

กิจกรรมเสริมทักษะ

ให้นักเรียนทดลองเขียนโปรแกรมตามตัวอย่างด้านล่าง แล้วบันทึกข้อมูลไว้ในโฟลเดอร์งานที่ชื่อว่า ex06 แล้วทดลองใช้งานโปรแกรมแล้วสังเกตผลลัพธ์ที่ได้

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



รูปภาพ แสดงผลลัพธ์ที่ได้จากการทำงานของโปรแกรม ex06_8.exe

คำสั่งประกอบการควบคุมทิศทาง

นอกจากคำสั่งควบคุมการทำงานของโปรแกรมแล้ว ภาษา C ยังมีคำสั่งอีกกลุ่มหนึ่ง ซึ่งอาจจะไม่ได้ใช้สำหรับควบคุมทิศทางการทำงานของโปรแกรมโดยตรง แต่คำสั่งเหล่านี้มักจะถูกนำไปใช้ร่วมกับคำสั่งอื่น ๆ เช่น นำไปใช้เพื่อหยุดการเลือกทำ หรือออกจากการทำงานของระบบ ซึ่งคำสั่งเหล่านี้ ได้แก่ break , continue และ exit( )

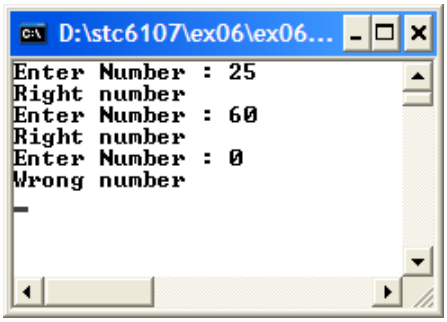
1. คำสั่ง break

คำสั่ง break ถูกนำไปใช้ร่วมกับคำสั่งแบบเลือกทำหรือวนซ้ำ เพื่อให้โปรแกรมหยุดการทำงานของคำสั่งแบบเลือกทำหรือวนซ้ำที่กำลังทำงานอยู่

กิจกรรมเสริมทักษะ

ให้นักเรียนทดลองเขียนโปรแกรมการใช้คำสั่ง break ซ้อนอยู่ในคำสั่ง if ตามตัวอย่างด้านล่าง แล้วบันทึกข้อมูลไว้ในโฟลเดอร์งานที่ชื่อว่า ex06 แล้วทดลองใช้งานโปรแกรมแล้วสังเกตผลลัพธ์ที่ได้

```
1  #include <stdio.h>
2  #include <conio.h>
3  int  num , count=0 ;
4  main()
5  {
6  while (count < 10)
7        {
8        printf("Enter Number : ");
9        scanf("%d",&num);
10       if (num ==0)
11           {
12           printf("Wrong number \n");
13           break;
14           printf("Enter new number \n");
15           }
16       else
17           printf("Right number \n");
18       }
19 getch();
20 }
```



รูปภาพ แสดงผลลัพธ์ที่ได้จากการทำงานของโปรแกรม ex06_9.exe

เมื่อสั่งรันโปรแกรมให้ทดลองป้อนตัวเลขเข้าไปหลาย ๆ จำนวน จากนั้นให้ป้อน 0 ซึ่งจะทำให้เงื่อนไข if เป็นจริง โปรแกรมจะแสดงข้อความ Wrong number แล้วออกจากการทำงานของคำสั่ง if และลูป while ทันที เนื่องจากการทำงานของคำสั่ง break จึงไม่มีการแสดงข้อความ Enter new number

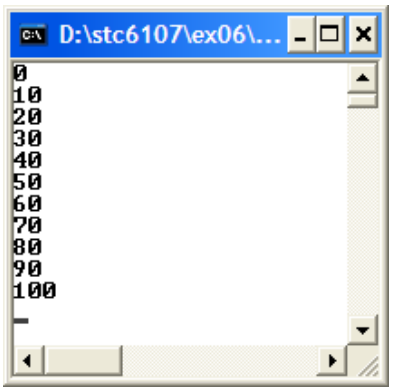
2. คำสั่ง continue

คำสั่ง continue จะใช้งานร่วมกับคำสั่งวนลูป while , do-while และ for เพื่อสั่งให้โปรแกรมหยุดการทำงานในรอบปัจจุบันแล้ววนกลับไปเริ่มทำงานในรอบใหม่

กิจกรรมเสริมทักษะ

ตัวอย่าง ให้นักเรียนทดลองเขียนโปรแกรมการใช้คำสั่ง continue ซ้อนอยู่ในคำสั่ง for ตามตัวอย่างด้านล่าง แล้วบันทึกข้อมูลไว้ในโฟลเดอร์งานที่ชื่อว่า ex06 แล้วทดลองใช้งานโปรแกรมแล้วสังเกตผลลัพธ์ที่ได้

```
1  #include <stdio.h>
2  #include <conio.h>
3  int  count ;
4  main()
5  {
6  for (count=0; count<=100; count++)
7     {
8     if (count%10 = = 0)
9        printf("%d \n",count);
10    else
11       {
12       continue;
13       printf("The number can't div by 10. \n");
14       }
15    }
16 getch();
17 }
```



รูปภาพ แสดงผลลัพธ์ที่ได้จากการทำงานของโปรแกรม ex06_10.exe

เมื่อสั่งรันโปรแกรม เงื่อนไขคำสั่ง if เป็นเท็จ โปรแกรมจะทำงานตามคำสั่ง continue นั่นคือ หยุดการทำงานในรอบปัจจุบันกลับไปเริ่มทำงานในรอบงานใหม่ จึงทำให้ไม่มีการแสดงข้อความ The number can't div by 10.

3. คำสั่ง exit( )

คำสั่ง exit( ) จะใช้สำหรับออกจากการทำงานของโปรแกรม และต้องประการ Header file ที่ชื่อว่า stdlib.h ก่อน ซึ่งรูปแบบการเรียกคำสั่ง exit( ) มีดังนี้

รูปแบบ

```
#include <stdlib.h>
    …
    …
  exit( 0);
```

การออกจากการทำงานของโปรแกรม โดยการใส่เลขศูนย์ไว้ในวงเล็บหมายถึง การออกจากโปรแกรมแบบปกติ