บทที่ 5

คำสั่งการตัดสินใจและวนรอบ

ในการเขียนโปรแกรมโดยทั่วไป (Alexander, A. ,Online) มักจะต้องมีส่วนที่ทำการตัดสินใจ ด้วยการทดสอบเงื่อนไขหรือเปรียบเทียบข้อมูล และต้องมีการทำงานซ้ำๆ หรือสั่งให้มีการวนรอบ ของข้อมูลเป็นประจำในการทำงานของโปรแกรมคอมพิวเตอร์ (Steven และ Lutfar, 2006) (Brian W. K., Online) ในบทนี้ได้ทำการแบ่งคำสั่งทำงานเป็น 2 ประเภทคือ

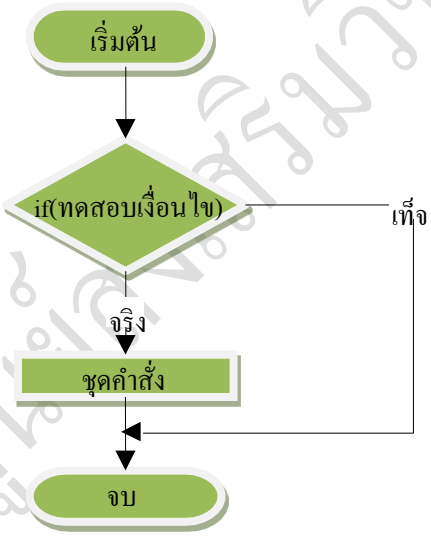
1. คำสั่งการตัดสินใจ (decision statement) เป็นกลุ่มคำสั่งที่มีเงื่อนไขให้ทำการทดสอบและทำการเลือกทำงานตามที่กำหนดเมื่อเงื่อนไขถูกต้อง และถ้าเงื่อนไขไม่ถูกต้องให้ยกเลิกการทำงาน หรือทำงานที่กำหนดอีกเรื่องหนึ่งที่แตกต่างกับงานของเงื่อนไขที่ถูกต้อง กลุ่มคำสั่งได้แก่คำสั่ง if_else คำสั่ง switch_case คำสั่ง break เป็นต้น

2. คำสั่งวนรอบ (iteration statement) เป็นอีกกลุ่มคำสั่งที่สำคัญอย่างมากในการเขียนโปรแกรม แต่ละครั้ง การเขียนโปรแกรมคอมพิวเตอร์นั้น มีวัตถุประสงค์หลักต้องการให้เครื่องคอมพิวเตอร์ทำงานแทนมนุษย์ ดังนั้นนอกจากต้องสามารถตัดสินใจตามเงื่อนไขที่กำหนดแล้ว ยังต้องสามารถทำงานซ้ำๆ ตามที่มนุษย์กำหนดให้ทำงานได้ด้วย

คำสั่งการตัดสินใจ if มีรูปการใช้งานตามลักษณะการนำคำสั่ง if มาทำการแก้ปัญหา ซึ่งในการตัดสินใจในการแก้ปัญหาว่าเมื่อโปรแกรมทำการทดสอบเงื่อนไขว่าถูกต้องแล้วต้องทำโปรแกรมส่วนใด และถ้าตรวจสอบเงื่อนไขแล้วไม่ถูกต้อง มีโปรแกรมให้ทำหรือไม่ หรือทำการตรวจสอบเงื่อนไขอื่นต่อหรือไม่ ดังนั้นคำสั่ง if จึงมีรูปแบบการเขียนสั่งงาน 3 รูปแบบ คือ รูปแบบทางเลือกเดียว รูปแบบสองทางเลือก และรูปแบบหลายทางเลือก

รูปแบบทางเลือกเดียว

รูปแบบของ if() ทางเลือกเดียว เป็นการทำงานที่การทดสอบเงื่อนไขก่อนว่า ถ้าภายใต้เงื่อนไขเป็นจริง (มีค่าทางตรรกะ ไม่เท่ากับศูนย์) จะทำชุดคำสั่งให้ทำงาน แต่ถ้าทดสอบแล้วผลการทดสอบเป็นเท็จ (มีค่าทางตรรกะ เท่ากับศูนย์) จะไม่ชุดคำสั่งให้ทำงาน จะจบการทำงานของคำสั่ง if() และจะทำงานคำสั่งต่อไปหรือจบโปรแกรม รูปที่ 5.1



รูปที่ 5.1 แสดงโฟลว์ชาร์ทการทำงานคำสั่ง if() รูปแบบทางเดียว

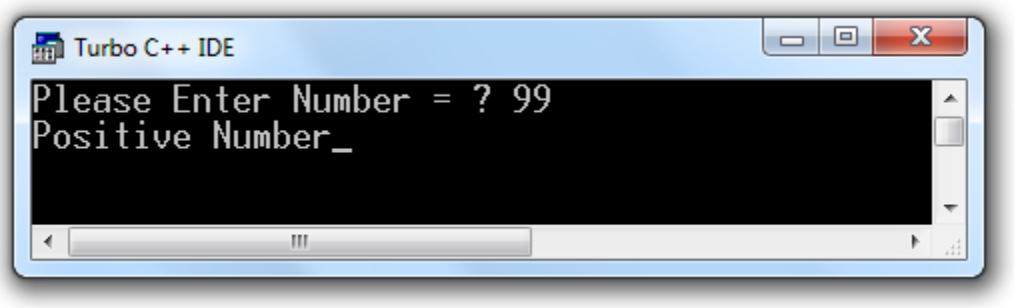
รูปแบบ

```
if(ตัวแปร ตัวกระทำเปรียบเทียบ ค่าคงที่หรือตัวแปร)
        {
                ชุดคำสั่ง
        }
```

หมายเหตุ ชุดคำสั่งถ้ามีเพียงคำสั่งเดียวไม่ต้องใส่เครื่องหมายปีกกาเปิดและปิด แต่ถ้าในชุดคำสั่งมีคำสั่งมากกว่า 1 คำสั่ง จะต้องใส่เครื่องหมายปีกกาเปิดและปิด เพื่อให้ทำงานทั้งชุดคำสั่ง ดังตัวอย่างโปรแกรมที่ 5.1

```
#include<stdio.h>
#include<conio.h>
        int x;
void main()
        {
         clrscr();
         printf("Please Enter Number = ?");
         scanf("%d", &x);
                if(x > 0)
                printf("Positive Number");
         getch();
        }
```

ผลการทำงานโปรแกรม

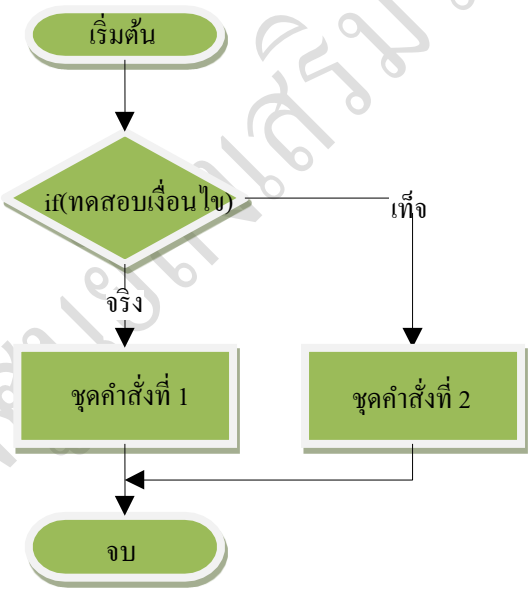


คำอธิบาย

จากตัวอย่างโปรแกรมที่ 5.1 เป็นโปรแกรมตรวจสอบเลขที่รับจากผู้ใช้ทางคีย์บอร์ด ว่าเป็นเลขจำนวนเต็มบวกหรือไม่ โดยโปรแกรมจะรอรับค่าจำนวนตัวเลขมาเก็บไว้ที่ตัวแปร x จากการป้อนทางแป้นพิมพ์ นำมาทำการ เปรียบเทียบค่าตัวเลขที่รับเข้ามาว่ามากกว่าศูนย์หรือไม่ ถ้าเป็นจริงก็จะแสดงข้อความ Positive Number

รูปแบบสองทางเลือก

รูปแบบของ if() ตั้งแต่สองทางเลือกขึ้นจะต้องใช้ else ร่วมทำงาน ซึ่งการทำงานจะทำการทดสอบเงื่อนไขเช่นเดียวกับการทดสอบทางเลือกเดียว โดยถ้าเงื่อนไขเป็นจริง จะทำงานในชุดคำสั่งที่ 1 แต่ถ้าทดสอบแล้วผลการทดสอบเป็นเท็จ จะทำงานในชุดคำสั่งที่ 2 ที่อยู่ภายใต้ else และจบการทำงาน รูปที่ 5.2



รูปที่ 5.2 แสดงโฟลว์ชาร์ทการทำงานคำสั่ง if() สองทางเลือก

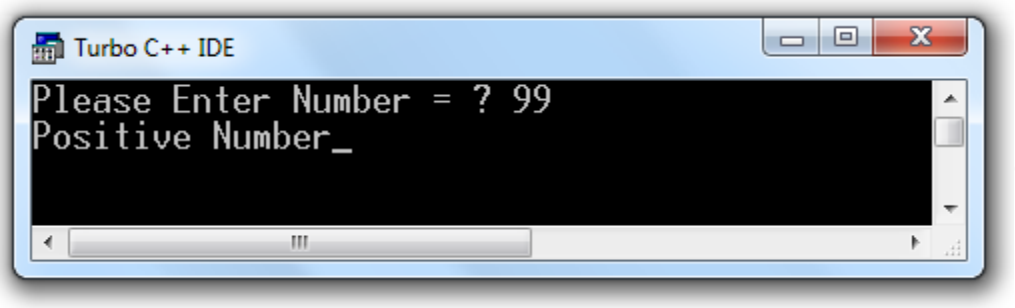
รูปแบบ

```
if (ตัวแปร ตัวกระทำเปรียบเทียบ ค่าคงที่หรือตัวแปร)
  {
        ชุดคำสั่งที่ 1
  }
else
  {
        ชุดคำสั่งที่ 2
  }
```

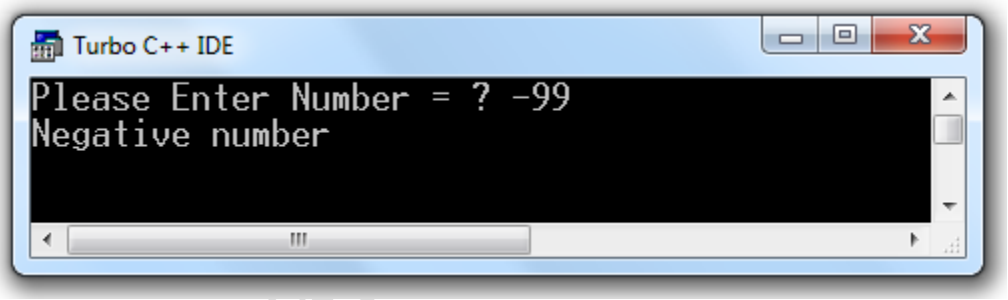
ตัวอย่างโปรแกรมที่ 5.2

```
#include<stdio.h>
#include<conio.h>
int x;
void main()
{       clrscr();
        printf("Please Enter Number = ?");
        scanf("%d", &x);
                if(x > 0)
                printf("Positive Number");
        else
                printf("Negative Number");
         getch();
}
```

ผลการทำงานโปรแกรม



หรือ

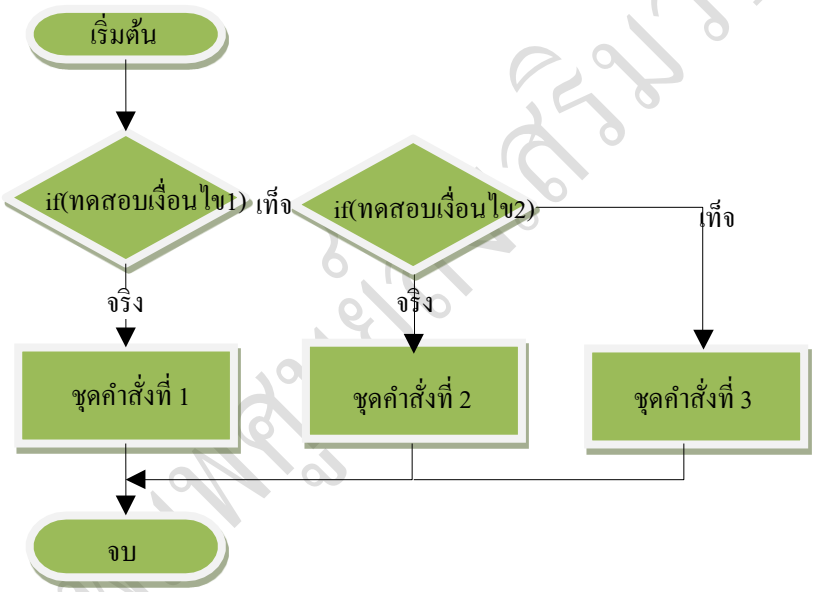


คำอธิบาย

จากตัวอย่างโปรแกรมที่ 5.2 โปรแกรมแสดงตรวจสอบ ในรูปแบบสองทางเลือกของการรับเลขจำนวนเต็มบวกและจำนวนเต็มลบ โปรแกรมจะรอรับค่าจำนวนตัวเลขมาเก็บไว้ที่ตัวแปร x จากการป้อนทางแป้นพิมพ์ นำมาทำการ เปรียบเทียบค่าตัวเลขที่รับเข้ามาว่ามากกว่าศูนย์หรือไม่ ถ้าเป็นจริงก็จะแสดงข้อความ Positive Number แต่ถ้าเป็นเท็จ จะแสดงข้อความ Negative Number

รูปแบบหลายทางเลือก

รูปแบบของ if() หลายทางเลือกจะทำการเปรียบเทียบ ถ้าเงื่อนไขที่ 1 เป็นจริงจะทำชุดคำสั่งที่ 1 แต่ถ้าเป็นเท็จจะทำการเปรียบเทียบกับเงื่อนไขที่ 2 ถ้าเงื่อนไขที่ 2 เป็นจริงจะทำชุดคำสั่งที่ 2 แต่ถ้าเป็นเท็จจะทำชุดคำสั่งที่ 3 รูปที่ 5.3



รูปที่ 5.3 แสดงโฟลว์ชาร์ทการทำงานคำสั่ง if() หลายทางเลือก

รูปแบบ

```
if (ตัวแปร ตัวกระทำเปรียบเทียบ ค่าคงที่หรือตัวแปร)
  {
        ชุดคำสั่งที่ 1
  }
else if (ตัวแปร ตัวกระทำเปรียบเทียบ ค่าคงที่หรือตัวแปร)
  {
        ชุดคำสั่งที่ 2
  }
else
  {
        ชุดคำสั่งที่ 3
  }
```

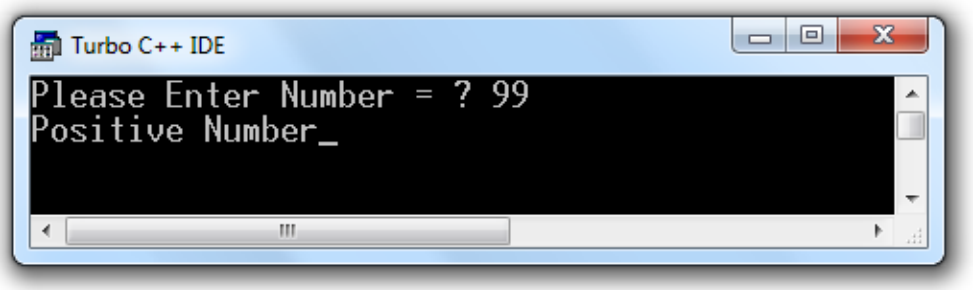
ตัวอย่างโปรแกรมที่ 5.3 โปรแกรมตรวจสอบเลขเงื่อนไขให้ทำงานด้วย 3 ชุดคำสั่ง ชุดที่หนึ่งแสดงเลขจำนวนเต็มบวก ชุดที่สองแสดงเลขจำนวนเต็มลบ และชุดที่สามแสดงศูนย์

```
#include<stdio.h>
#include<conio.h>
int x;
void main()
{
  clrscr();
  printf("Enter Number = ");
```

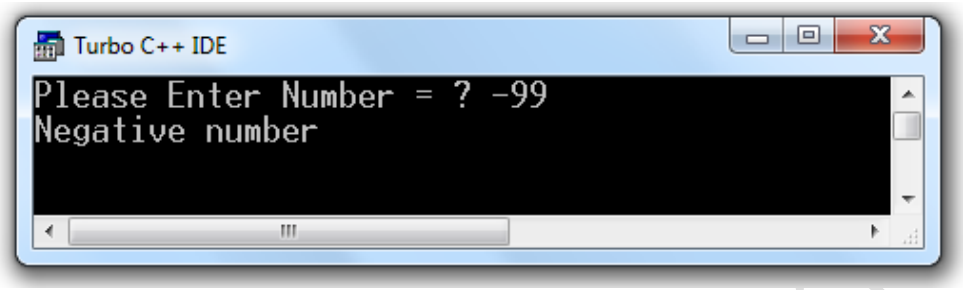
```
ต่อจากด้านบน
        scanf("%d",&x);
        if( x > 0)
                printf("Positive Number");
        else if( x < 0)
                printf("Negative Number");
        else
                printf("Zero Number");
                getch();
}
```

ผลการทำงานโปรแกรม

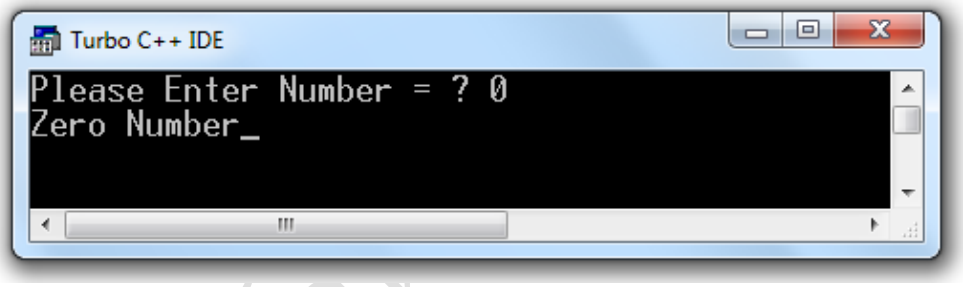
ชุดที่ 1



หรือชุดที่ 2

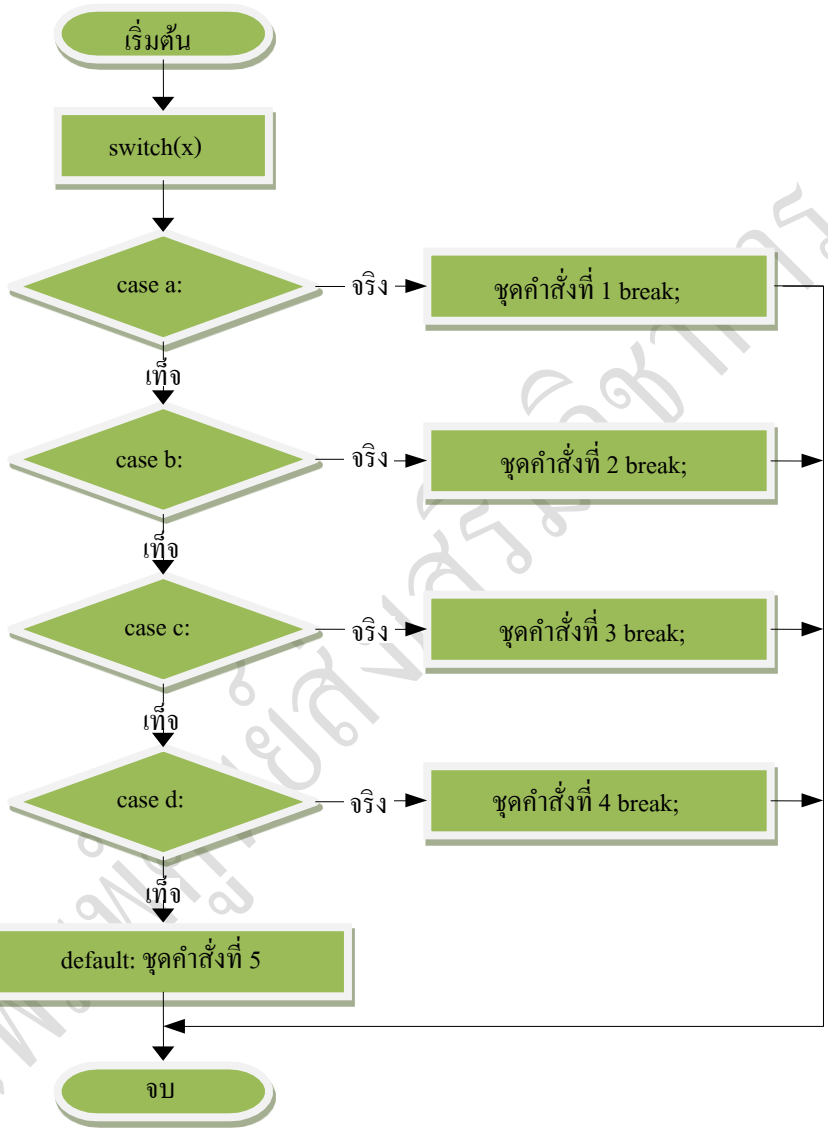


หรือชุดที่ 3



คำสั่ง switch()/case

เป็นคำสั่งที่มีวัตถุประสงค์ใช้ในการเลือกรายการทำงาน หรือกลุ่มคำสั่งของผู้ใช้งานโปรแกรม โดยผ่านทางแป้นพิมพ์ ซึ่งคำสั่งที่เป็นการเปรียบเทียบข้อมูลอีกคำสั่งหนึ่งเหมาะสำหรับการเปรียบเทียบข้อมูลหลาย ๆ ทางเลือก แต่การเปรียบเทียบจะไม่สามารถเปรียบเทียบข้อมูลเชิงค่าตัวเลขชนิดให้เลือกค่ามากกว่าหรือน้อยกว่าเหมือนกับคำสั่ง if() ได้แต่จะเปรียบเทียบข้อมูลชนิดค่าคงที่, ตัวอักษร หรือตัวแปร ว่ามีค่าเหมือนกันหรือเท่ากันหรือไม่ ดังแสดงการทำงานในรูปที่ 5.4



รูปที่ 5.4 แสดงโฟลว์ชาร์ทการทำงานคำสั่ง switch()

การทำงานของคำสั่ง switch(x) จะนำตัวแปร x มาเปรียบเทียบกับค่าในแต่ละ case ว่ามีเป็นค่าคงที่ หรือตัวอักษรเหมือนกับใน case a หรือ case b หรือ case c หรือ case d ซึ่งถ้าตรงกับ case ใดหรือไม่ถ้าตรงกับ case ใดให้ทำงานตามชุดคำสั่ง 1, 2, 3, และ 4 ตามลำดับแล้วจบการทำงาน แต่ถ้าค่าของตัวแปร x ไม่ตรงกับ case ใด โปรแกรมจะทำชุดคำสั่งที่ 5 ที่อยู่ใต้ default ซึ่งคำสั่ง default จะมีหรือไม่ก็ได้ จำนวน case ที่ใช้ในการเปรียบเทียบข้อมูลอาจมีมากกว่าหรือน้อยกว่านี้ได้

<u>รูปแบบ</u>

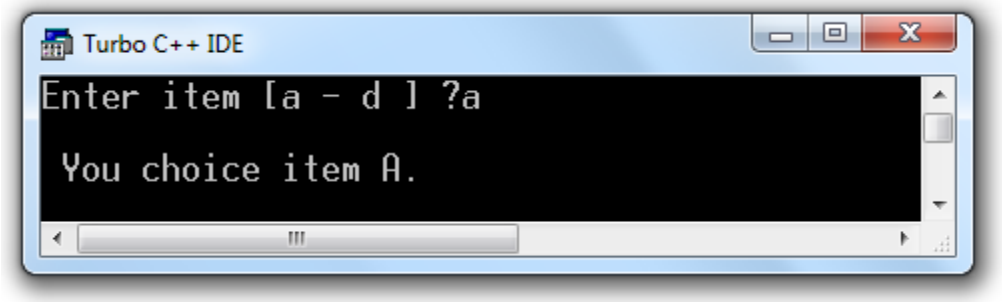
```
switch(x)
    {
        case a:
                ชุดคำสั่งที่ 1
                break;
        case b:
                ชุดคำสั่งที่ 2
                break;
        case c:
                ชุดคำสั่งที่ 3
                break;
        case d:
                ชุดคำสั่งที่ 4
                break;
        default :
                ชุดคำสั่งที่ 5
    }
```

หมายเหตุ x เป็นตัวแปรซึ่งต้องกำหนดค่าหรือมีการป้อนค่าให้กับตัวแปรก่อนที่จะนำมาเปรียบเทียบ a, b, c, d เป็นค่าคงที่, ตัวอักษร หรือตัวแปร คำสั่ง break ที่อยู่ท้ายแต่ละ case เป็นการสั่งให้โปรแกรมกระโดดออกจาก switch

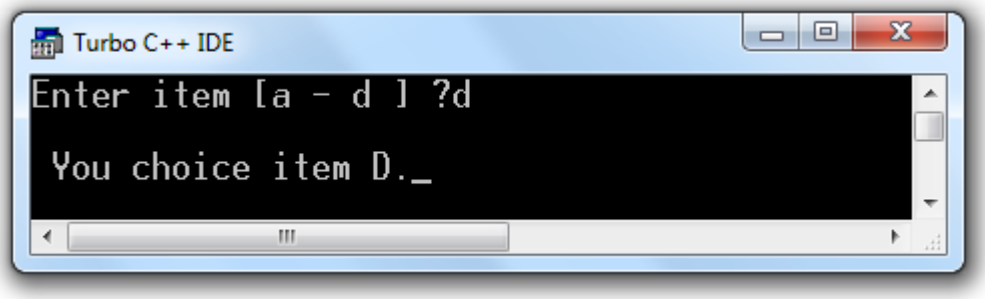
ตัวอย่างโปรแกรมที่ 5.4 โปรแกรมให้ผู้ใช้โปรแกรมเลือกรายการโปรแกรมย่อย

```
#include<stdio.h>
#include<conio.h>
char x;
void main()
{           clrscr();
            printf("Enter item [a - d ] ?");
            scanf("%c",&x);
switch(x)
        {       case 'a': printf("\n You choice item A."); break;
                case 'b': printf("\n You choice item B."); break;
                case 'c': printf("\n You choice item C."); break
                case 'd': printf("\n You choice item D."); break;
                default : printf("\n You choice fail item ");
        }
                getch();
}
```

ผลของการทำงานโปรแกรม



หรือ



หรือ

คำอธิบาย

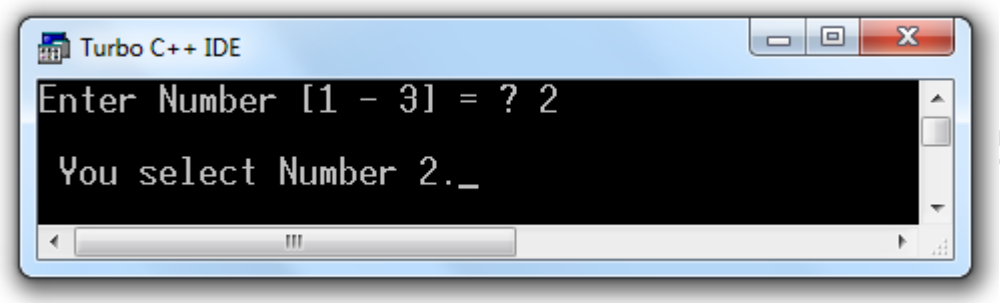
โปรแกรมจะแสดงข้อความ Enter item [a – d] ? โปรแกรมรอการป้อนค่าจากการกดแป้นพิมพ์ 1 ตัวอักษรมาเก็บไว้ที่ตัวแปร x คำสั่ง switch() ทำการเปรียบเทียบค่าของตัวแปร x กับ case ‘a’ ถึง case ‘d’ ถ้าค่าของตัวแปร x ตรงกับ case ใดจะทำคำสั่งใน case นั้น แต่ถ้าไม่ตรงกับ case ใดจะทำคำสั่งหลัง default

ข้อสังเกต

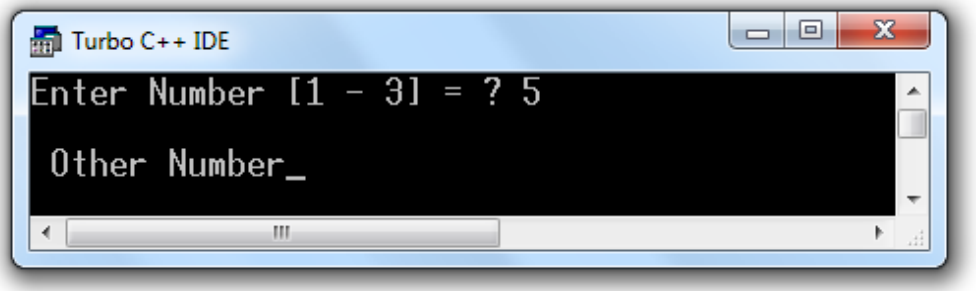
การกำหนดค่าที่รับจากแป้นพิมพ์ มาเก็บไว้ที่ตัวแปร x เป็นตัวแปรชนิดข้อมูลเป็นชนิด char ดังนั้นค่าที่อยู่ในแต่ละ case ต้องมีการเขียนในรูปของตรวจสอบที่เป็นตัวอักษรได้ด้วยใส่เครื่องหมาย ‘ ’ ลงในตัวอักษร เช่น case ‘a’: จากตัวอย่างโปรแกรมที่ 5.5

```
#include<stdio.h>
#include<conio.h>
int x;
void main()
{           clrscr();
            printf("Enter Number [1 -3] = ? ");
            scanf("%d",&x);
switch(x)
    {   case 1: printf("\n You select Number 1."); break;
        case 2: printf("\n You select Number 2."); break;
        case 3: printf("\n You select Number 3."); break;
        default: printf("\n Other Number");
    }
```

ผลการทำงานโปรแกรม



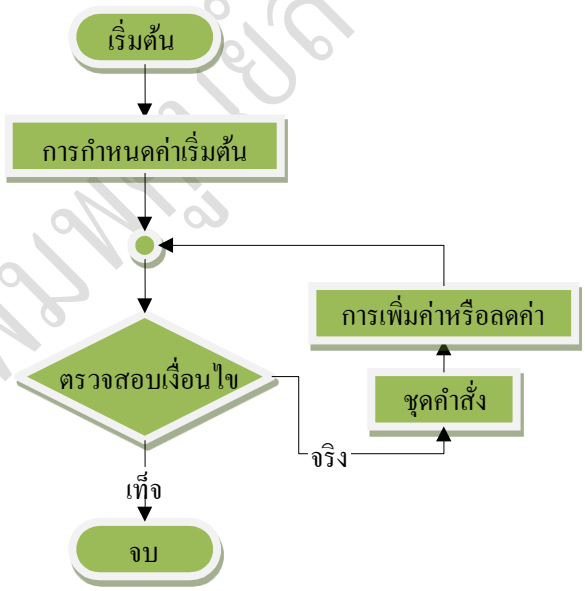
หรือ



การเขียนโปรแกรมที่มีลักษณะการทำงานวนรอบที่ซ้ำ ๆ กัน ในโปรแกรมต่างๆ รวมทั้งโปรแกรมภาษาซี สามารถใช้คำสั่งหรือฟังก์ชันวนลูป (Loop) เพื่อช่วยลดขนาดของโปรแกรมให้เล็กลงและมีการทำงานที่รวดเร็วขึ้น ใช้เวลาในการเขียนโปรแกรมน้อยลง นั้น มีกลุ่มคำสั่งที่นำมาใช้ในงานได้หลายคำสั่ง ได้แก่ คำสั่ง for(), do() และ do() / while() ซึ่งเป็นคำสั่งวนรอบเหมือนกัน แต่มีรูปแบบของคำสั่งและลักษณะการทำงานที่ต่างกัน

คำสั่ง for()

เป็นคำสั่งวนรอบที่เป็นพื้นฐานสำคัญในการทำความเข้าใจการวนลูปของโปรแกรม การใช้คำสั่งในการเขียนโปรแกรมวนลอมนิยามนำมาสั่งให้ทำงานวนรอบที่มีลักษณะจำนวนรอบที่แน่นอน สามารถกำหนดจำนวนรอบจากค่าเริ่มต้นจากน้อยไปหามาก หรือจากมากไปหาน้อย ได้อย่างง่ายซึ่งค่าเริ่มต้นที่กำหนดให้กับคำสั่ง for() จะถูกทำงานเพียงครั้งแรกครั้งเดียว จากนั้นจะนำค่าที่กำหนดเป็นค่าเริ่มต้น มาทำการตรวจรอบในการทำงาน ถ้าถูกต้องตามเงื่อนไขก็จะทำงานตามที่กำหนด แล้วทำการเพิ่มค่าหรือลดค่าที่ละหนึ่งค่าแล้วนำค่าที่ได้ทำการลดหรือเพิ่มค่ามาทำการตรวจสอบเงื่อนไขถูกต้องหรือไม่ถ้าถูกต้องทำงานตามที่กำหนด จนผลการตรวจสอบเงื่อนไขไม่ถูกต้อง โปรแกรมหยุดทำงาน แสดงการทำงานด้วยโฟลว์ชาร์ท รูป 5.5



รูปที่ 5.5 โฟลว์ชาร์ตการทำงานของคำสั่ง for

รูปแบบ

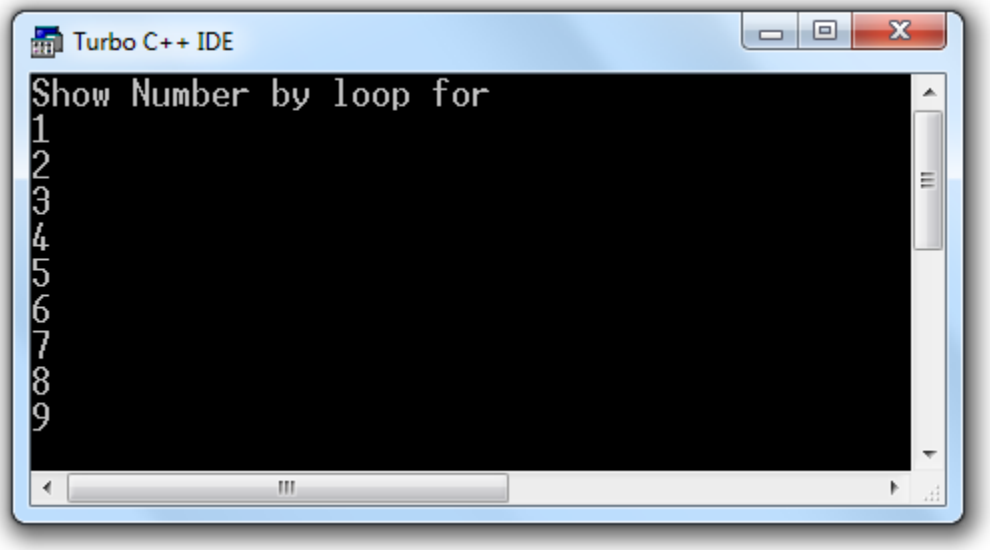
```
for(ค่าเริ่มต้น; เงื่อนไข; การเพิ่มค่าหรือลดค่า)
    {
      ชุดคำสั่ง
    }
```

ข้อสังเกต ห้ามใส่เครื่องหมายเซมิคอลลอน (;) หลังคำสั่ง for() เพราะการทำงานยังไม่จบคำสั่ง ด้วยเพราะเครื่องหมายเซมิคอลลอน มีความหมายสั่งให้ตัวคอมไพเลอร์นำคำสั่งด้านหน้า ; ไปประมวลผล

ตัวอย่างโปรแกรมที่ 5.6 โปรแกรมแสดงจำนวนเลขตามจำนวนรอบด้วยคำสั่ง for()

```
#include<stdio.h>
#include<conio.h>
int count;
void main()
{
        clrscr();
        printf("Show Number by loop for\n");
        for (count=1; count<10; count++)
                printf("%d \n",count);
        getch();
}
```

ผลการทำงานโปรแกรม

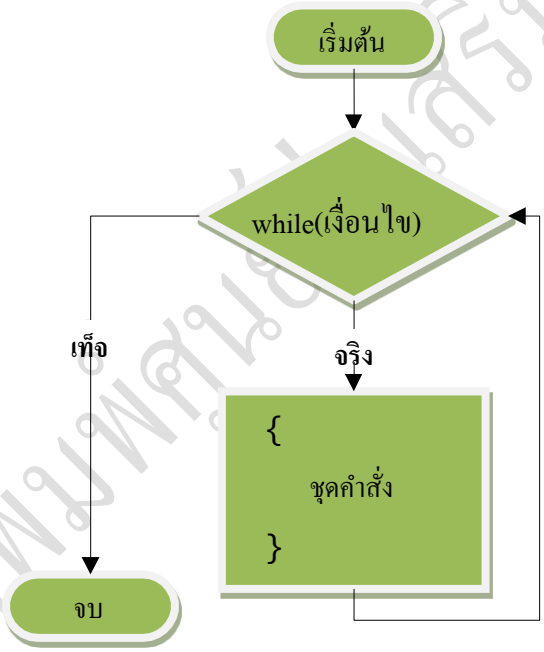


คำอธิบาย

1. กำหนดค่าเริ่มต้นให้กับตัวแปร count = 1
2. กำหนดเงื่อนไขในการวนรอบ count < 10 หรือวนรอบ 9 รอบ
3. ทำการตรวจสอบเงื่อนไขว่าเป็นจริงหรือไม่
4. ถ้าเงื่อนไขเป็นจริงพิมพ์ค่าในตัวแปร count และขึ้นบรรทัดใหม่ด้วยคำสั่ง \n
5. เพิ่มค่า count ขึ้นทีละ 1 ค่า ด้วยคำสั่ง count++
6. กลับไปทดสอบเงื่อนไขจนเป็นเท็จ
7. จบการทำงาน

คำสั่ง while()

เป็นคำสั่งวนรอบเช่นเดียวกับคำสั่ง for() เหมือนกัน มีการกำหนดค่าเริ่ม กำหนดเงื่อนไข การเพิ่มค่าหรือลดค่า แต่รูปแบบการกำหนดค่าต่าง ๆ แตกต่างกัน โดยการทำงานของคำสั่ง while()จะทำการตรวจสอบเงื่อนไขก่อนการกระทำคำสั่งในลูป ถ้าเงื่อนไขเป็นจริงจะทำชุดคำสั่งภายในปีกกาเปิดและปิด ({ }) แต่ถ้าเงื่อนไขเป็นเท็จจะออกจากการวนลูป ดังแสดงโฟลว์ชาร์ท รูปที่ 5.6



รูปที่ 5.6 โฟล์วชาร์ตการทำงานของคำสั่ง while()

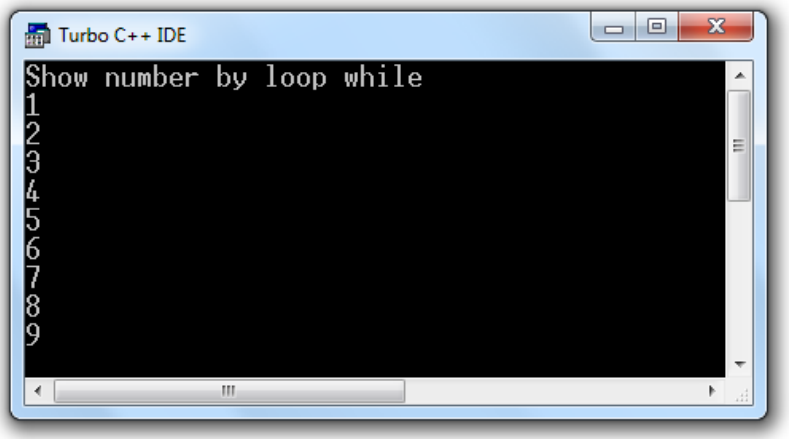
รูปแบบ

```
while(ตรวจสอบเงื่อนไข)
        {
            ชุดคำสั่ง
        }
```

ตัวอย่างโปแกรมที่ 5.7 โปรแกรมแสดงค่าตัวเลขด้วยคำสั่ง while()

```
#include<stdio.h>
#include<conio.h>
int count;
void main()
    {
        clrscr();
        count = 1;
        printf("Show number by loop while \n");
        while( count <10)
          {
            printf("%d \n",count);
            count++;
          }
        getch();
    }
```

ผลการทำงานโปรแกรม



ข้อสังเกต

ผลการทำงานของโปรแกรมได้ไม่แตกต่างกัน แต่แตกต่างกันของรูปแบบการสั่งให้โปรแกรมทำงาน โดยค่าเริ่มต้นของลูปของคำสั่ง while() อยู่นอกลูป และการเพิ่มค่าที่ละหนึ่งค่าจะอยู่ในส่วนของชุดคำสั่งให้ทำงาน ซึ่งแตกต่างกับคำสั่ง for() ที่อยู่ในโครงสร้างของคำสั่ง for(count = 1; count < 10; **count++**)

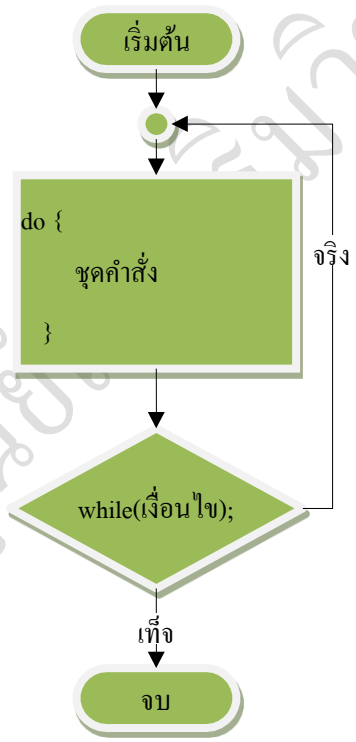
คำสั่ง do_while();

เป็นคำสั่งวนรอบที่มีโครงสร้างคล้ายกับคำสั่ง while() แตกต่างที่การทำงานของ do_while() จะทำชุดคำสั่งของลูปก่อน แล้วจึงทำการตรวจสอบเงื่อนไขของลูป ถ้าเงื่อนไขเป็นจริงจะทำชุดคำสั่งในลูปต่อ แต่ถ้าเงื่อนไขเป็นเท็จจะออกจากลูป ดังแสดงโฟลว์ชาร์ด รูปที่ 5.7

<u>รูปแบบ</u>

```
do
{
        ชุดคำสั่ง
}

while(เงื่อนไข);
```



รูปที่ 5.7 โฟล์วชาร์ตการทำงานของคำสั่ง do_while();

ตัวอย่างโปรแกรมที่ 5.8 โปรแกรมแสดงจำนวนเลขตามจำนวนรอบด้วยคำสั่ง do_while()

```
#include<stdio.h>
#include<conio.h>
count = 1;
void main()
        {
                clrscr();
                printf("show number by loop do_while() \n");
                do{
                        printf("%d \n",count);
                        count++;
                    }
                while(count < 10);
        getch();
        }
```

ผลการทำงานของโปรแกรมเหมือนกับตัวอย่างโปรแกรมที่ 5.6 และ 5.7 แต่ผู้อ่านสามารถเขียนโปรแกรมได้หลากหลายรูปแบบ

สรุป

ในการส่งให้โปรแกรมคอมพิวเตอร์ประมวลผลในการทำงาน สามารถแบ่งกลุ่มได้ 3 กลุ่ม คือ กลุ่มที่ 1 ทำตามขั้นตอนที่ละขั้นตอน ตั้งแต่คำสั่งแรกจนคำสั่งสุดท้าย กลุ่มที่ 2 กลุ่มคำสั่งให้ทำการทดสอบเงื่อนไขก่อน ถ้าเงื่อนไขถูกต้องให้ทำงานที่กำหนด ซึ่งกลุ่มคำสั่งนี้ได้แก่ ฟังก์ชัน if(), if()/else และ switch/case ส่วนกลุ่มคำสั่งที่ 3 เป็นการสั่งกำหนดให้ทำงานเป็นวนรอบ ได้แก่ ฟังก์ชัน for(), while() และ do_while โดยรูปแบบในการสั่งการทำงาน ลำดับแรกกำหนดด้วยค่าเริ่มต้น(initial) ลำดับถัดมาทำงานตามที่กำหนด และทำการเพิ่มหรือลดค่าที่หนึ่ง(ในกรณีนับค่าที่หนึ่ง) ลำดับถัดมาทำการตรวจสอบว่าทำงานครบตามจำนวนหรือไม่ ถ้ายังไม่ครบตามเงื่อนไข โปรแกรมจะทำงานซ้ำ จนครบจำนวนที่กำหนด ตัวประมวลผลสั่งให้ออกจากวนรอบ

ข้อสังเกต ในการกำหนด จำนวนรอบกับการลดหรือเพิ่มค่า ผู้เขียนโปรแกรมต้องคำนวณรบให้ถูกต้อง เพราะถ้ากำหนดผิดพลาด โปรแกรมอาจจะวนอยู่ลูปตลอดเวลาได้

แบบฝึกหัดท้ายบทที่ 5

ตอนที่ 1 จงเติมคำหรือข้อความในช่องว่างต่อไปนี้ให้ถูกต้อง

1. ถ้าต้องการให้คอมพิวเตอร์ช่วยในการตัดสินใจสามารถใช้คำสั่งใดทำงาน

   .....

2. if(เงื่อนไข) กับ if(เงื่อนไข)…else… มีความแตกต่างกันอย่างไร

   .....

3. จงเขียนความแตกต่างระหว่างคำสั่ง if()_else และ switch()

   .....

4. จงเขียนความแตกต่างระหว่างคำสั่ง for() และ while()

   .....

5. จงเขียนความแตกต่างระหว่างคำสั่ง while() และdo_while()

   .....

6. จงทำการเขียนโค๊ดโดยทำการแปลงคำสั่ง for (a = 1; a < 10; ++a) ให้มาใช้คำสั่ง while() แทน

   .....

7. จงทำการเขียนโค๊ดโดยทำการแปลง for (x = 0, y = 10; x > 10; x+=5) ให้มาใช้คำสั่ง while() แทน

   .....

8. จงทำการเขียนโค๊ดโดยทำการแปลง for (j=-1; j < 0 || j+2 <1; j+=10) ให้มาใช้คำสั่ง while() แทน

   .....

9. จงทำการเขียนโค๊ดโดยทำการแปลง for (j=-1; j < 0 || j+2 <1; j+=10) ให้มาใช้คำสั่ง while() แทน

   .....

10. จงทำการเขียนโค๊ดโดยทำการแปลง for (i = 0; i < 5 && i*5 > 60; i += 10)

    .....

ตอนที่ 2 จงทำเครื่องหมายกากบาท (x) ทับหน้าข้อที่ถูกต้องที่สุด

1. คำสั่งใดเหมาะกับการให้มีทางเลือกจำนวนมากๆ

   ก. if/else ข. for

   ค. while ง. do_while

2. คำสั่งใดเหมาะกับการให้คอมพิวเตอร์ทำงานวนรอบแบบกำหนดจำนวนรอบที่แน่นอนได้ดี

   ก. if/else ข. for

   ค. while ง. do_while

3. คำสั่งใดเหมาะกับการทำงานวนรอบโดยให้หยุดการทำงานด้วยเงื่อนไขพิเศษหนึ่งเงื่อนไข

   ก. if/else ข. for

   ค. while ง. do_while

4. คำสั่งใดทำหน้าที่สั่งให้หยุดทำงานเมื่องานที่กำหนดสำเร็จ

   ก. case ข. switch()

   ค. default ง. break

5. คำสั่งใดทำหน้าที่สั่งให้ทำงานเมื่องานเมื่อข้อมูลที่ป้อนเข้ามาไม่เป็นไปตามเงื่อนไข

   ก. case ข. switch()

   ค. default ง. break

6. คำสั่งใดทำหน้าที่สั่งให้ทำการเพิ่มค่าที่หนึ่งของกลุ่มคำสั่งวนรอบ

ก. **i ข. i++

ค. –I ง. i+=1

7. คำสั่งใดทำหน้าที่หรืองานก่อน 1 งานแล้วจึงทำการตรวจสอบเงื่อนไขการทำงาน

ก. if/else ข. for

ค. while ง. do_while

8. ข้อใดเขียนเงื่อนไขให้ทำการวนรอบจำนวน 10 ถูกต้อง

ก. for(count=1; (count<10);count++)

ข. for(count=0; (count<10);count++)

ค. for(count=1; (count<=10);count++)

ง. for(count=0; (count<=10);count++)

9. ข้อใดเป็นการสั่งให้ทำงานวนรอบไม่มีโอกาสออกจากลูป เมื่อ count = 1

ก. while (count<10); count++

ข. while (count>10);count++

ค. for(count=1; (count<10);count++)

ง. for(count=10; (count<0);count--)

10. คำสั่งวนรอบใดที่นิยมทำการเขียนเป็นแบบลัด

ก. if/else ข. for

ค. while ง. do_while

ตอนที่ 3 จงทำการวิเคราะห์คำถามและทำการเขียนอภิปรายคำตอบตามที่ผู้อ่านเข้าใจโดยยึดความถูกต้องของเนื้อหาประกอบการบรรยาย

1. จากการโปรแกรมภาษาซี

```
sum = 0.0;
scanf("%f", &x);
do {
    sum += x;
   scanf("%f", &x);
    }
while (x > 0.0);
```

ให้ผู้อ่านเขียนโปรแกรมด้านบนให้สมบูรณ์และเขียนผลการทำงานโปรแกรม

2. จากโปแกรมภาษาซีด้านล่าง

```
#include<stdio.h>
void main()
    {
    for (int k = 10 ; k < 5 ; k++);
    printf ("\t %d", k);
     }
```

ให้ผู้อ่านปรับแก้ให้ถูกต้องและให้ผลการทำงานโปรแกรม ดังนี้

10 8 6 4 2

3. จากโปแกรมภาษาด้านล่าง

```
#include <stdio.h>
void main()
{
        double num;
        num = 0;
        printf ("Enter any characters and press CRTL+Z to quit\n");
        while (getchar() != EOF)
        {
                num++;
        }
        printf("%.0f\n", num);
}
```

ให้ผู้อ่านเขียนความหมายระหว่างคำว่า CTRL+Z กับ EOF

4. ให้ผู้อ่านทำการปรับปรุงโปรแกรมที่ใช้คำสั่ง switch() ให้ใช้คำสั่ง if-else แทน

```
switch (a) {
        case 5 : a = 3;          break;
        case 10: a = 5;          break;
        case 15: b = a;          break;
        case 20: a = 10;         break;
        default: a = b;
          }
```

5. ให้ผู้อ่านทำการเขียนโปรแกรมโดยทำการแปลงคำสั่ง if()-else ให้มาใช้คำสั่ง switch() แทน

```
if (a == 0)
        a = 10;
else if (a == 5)
      x = a + 10;
else if (b == 10)
        a = b;
else if (a == 10 || a == 11 || a == 12)
       a = a * 2;
else
        a = 0;
```

เอกสารอ้างอิง

ศรัณย์ อินทโกสุม (2539). ทฤษฎีและตัวอย่างโจทย์การเขียนโปรแกรมด้วยภาษาซี กรุงเทพฯ : แมคกรอฮิล อินเตอร์เนชั่นแนล เอ็นเตอร์ไพรส์, อิงค์.

ธันวา ศรีประโมง (2539). การเขียนโปรแกรมภาษาซีสำหรับวิศวกรรม. พิมพ์ครั้งที่ 4. กรุงเทพฯ : มหาวิทยาลัยเทคโนโลยีมหานคร.

วิจักษณ์ ศรีสัจจะเลิศวาจา และดุษฎี ประเสริฐธิติพงษ์ ออนไลน์ : www.satit.su.ac.th/soottin.

Alexander, A. **Tutorial** Online : **http://www.cprogramming.com**.

Brian, W. K. Programming in C: A Tutorial Online : http://www.lysator.liu.se/c/bwktutor.html.

Steven, H. & Lutfar, R. (2006). Art of Programming Contest: C Programming | Data Structure | Algorithms (ACM supported), 2nd Edition.