บทที่ห้า

คำสั่งควบคุม (Control Statement)

โดยปกติการทำงานของคอมพิวเตอร์จะทำงานเรียงลำดับคำสั่งลงมาตั้งแต่ต้นโปรแกรมจนจบโปรแกรม แต่ถ้าเราต้องการเปลี่ยนแปลงขั้นตอนของการทำงานของคำสั่ง เช่น โดดข้ามไปทำคำสั่งใดคำสั่งหนึ่ง หรือให้วนกลับมาทำคำสั่งที่เคยทำไปแล้วอีก ลักษณะการสั่งงานแบบนี้จะต้องใช้คำสั่งควบคุม ดังนั้นคำสั่งควบคุมจึงเป็นคำสั่งที่ใช้เปลี่ยนแปลงลำดับขั้นตอนของการทำงานของโปรแกรม ซึ่งแบ่งออกเป็น 3 ชนิด คือ

1. คำสั่งให้ไปทำงานโดยไม่มีเงื่อนไข (Unconditional branch statement) ซึ่งได้แก่คำสั่ง goto
2. คำสั่งให้ไปทำงานโดยมีเงื่อนไข (Conditional branch statement) ได้แก่คำสั่ง if, switch
3. คำสั่งให้ไปทำงานเป็นวงจร (loop Control statement) ได้แก่ while, do – while, for

5.1 คำสั่ง goto

เป็นคำสั่งให้ไปทำงานยังคำสั่งหนึ่งคำสั่งใดโดยไม่มีเงื่อนไข

รูปแบบ

```
goto statement – label;
```

statement – label หมายถึง ชื่อประจำคำสั่ง (label) ที่ต้องการให้ไปทำงาน

ตัวอย่างที่ 1

```
……
……
goto loop1;
……
……
```

หมายความว่า ให้ไปทำคำสั่งที่มีชื่อประจำคำสั่ง เป็น loop1 ดังนี้

loop1……	……
……	……
……	……
……	loop 1:
……	……
……	……
goto loop1;	goto loop1;
……	……
……	……

ตัวอย่างที่ 2 ให้เขียนโปรแกรมหาพื้นที่วงกลมที่มีรัศมีตั้งแต่ 1,2,3,4,…… ไปเรื่อย ๆ

```
main ( )
{
        float   r = 1;
         float   area;
loop1:   area = 22/7 *r*r;
         printf ("R =  %. 2f area = %.4f \n", r, area);
         r + +;
         goto   loop1;
}
```

5.2 คำสั่ง if

เป็นคำสั่งให้ไปทำงานยังคำสั่งใดคำสั่งหนึ่งโดยมีเงื่อนไขให้ตัดสินใจอย่างหนึ่งอย่างใดก่อน แล้วจึงไปทำงานโดยผลของการตัดสินใจจะมีโอกาสเป็นไปได้ 2 ทางคือ

1. จริง หมายความว่า มีค่าไม่เท่ากับศูนย์
2. ไม่จริง หมายความว่า มีค่าเท่ากับศูนย์

รูปแบบ

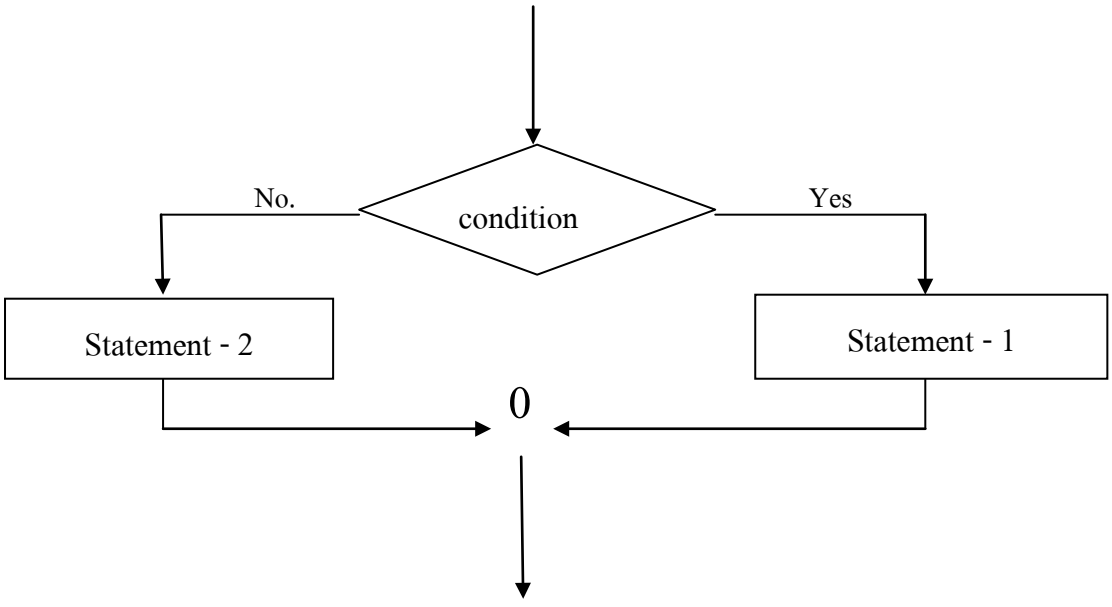
```
if (condition)
        statement – 1;
else
        statement – 2;
```

Condition หมายถึง เงื่อนไขที่สร้างให้คอมพิวเตอร์ตัดสินใจอย่างหนึ่งอย่างใดก่อนที่จะไปทำงานโดยถ้าเงื่อนไขเป็นจริง จะไปทำงานยัง statement – 1 ถ้าเงื่อนไขไม่เป็นจริงจะไปทำงานยัง statement – 2

Statement – 1 หมายถึงคำสั่งใด ๆ ในภาษา C 1 คำสั่ง

Statement – 2 หมายถึงคำสั่งใด ๆ ในภาษา C 1 คำสั่ง

ค่าที่จะนำไปเปรียบเทียบกันเพื่อให้เป็นเงื่อนไขจะต้องเป็นค่าชนิดเดียวกัน ดังนั้นลักษณะการทำงานของคำสั่ง if สามารถเขียนอยู่ในรูปของผังงาน ได้ดังนี้

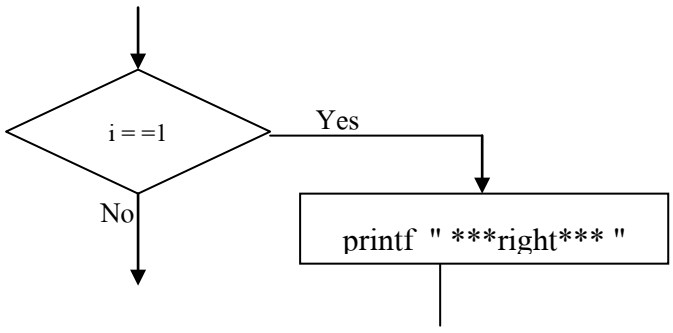


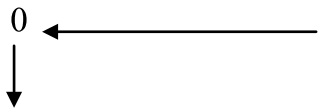
ตัวอย่างที่ 3 แสดงการรับค่า a และ b จากแป้นพิมพ์แล้วเอาค่า b มาทำการการตรวจสอบ ถ้าค่า b ไม่เท่ากับศูนย์ให้คำนวณหาค่า a หาร b แต่ถ้าค่า b เท่ากับศูนย์ให้พิมพ์ข้อความ "Cannot divide by Zero" ออกมาทางจอภาพ1

```
/* Divide the first number by the second.*/
#include <stdio.h>
main ( )
{
    int a, b;
     printf ("enter two number:");
     scanf ("%d %d", &a, &b);
   if (b != 0)

    printf ("%d \n", a/b);
   else printf ("cannot divide by zero \n");
}
```

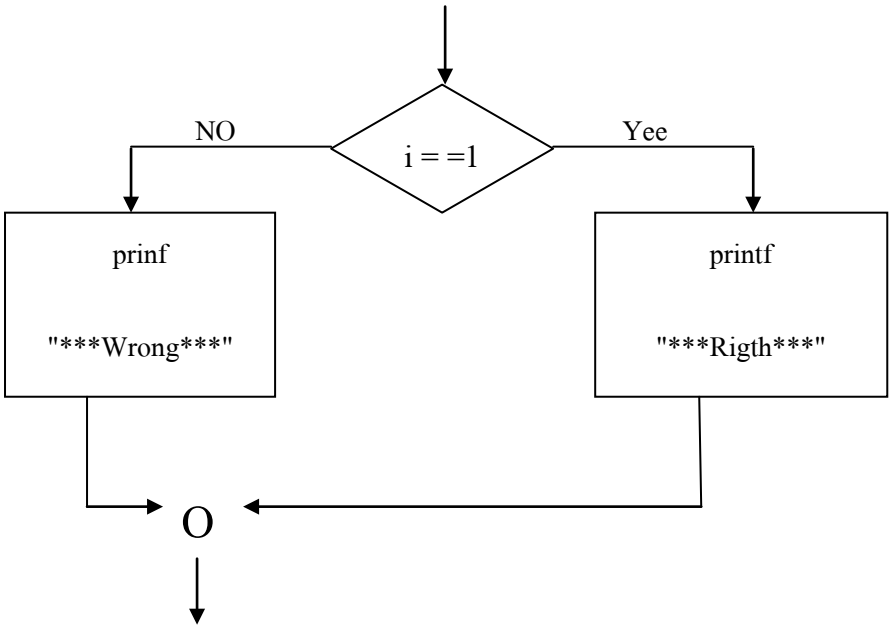
ตัวอย่างที่ 4 แสดงการใช้คำสั่ง if โดยละ else ทำการรับค่าจากแป้นพิมพ์มาตรวจสอบกับค่าที่กำหนดไว้คือ 123 ว่าค่าที่รับเข้ามานั้นเท่ากับค่าที่กำหนดหรือไม่ถ้าเท่าให้พิมพ์ " ***Right*** " ออกมาทางจอภาพ






```
main ( )
{
    int   i = 123;   int 1;
    scanf ("%d", &1);
    if (i==1)
    printf (" ***Right*** ");
}
```

ตัวอย่างที่ 5 แสดงการใช้คำสั่ง if โดยมี else ทำงานเดียวกับ ตัวอย่างที่ 4 แต่เพิ่มการเปรียบเทียบว่าถ้าไม่เท่าให้พิมพ์ "***Wrong***" ออกมาทางจอภาพด้วย



```
main ( )
{
    int   i = 123;   int   l;
    scanf ("%d", &l);
    i (i = = l)
    printf ("***Right***");
else
    printf ("***Wrong***");
}
```

ในกรณีที่ statement – 1 และ statement – 2 มีมากกว่า 1 คำสั่งจะต้องกั้นบล็อก (block) ให้กับกลุ่มคำสั่งนั้น โดยใช้เครื่องหมายแสดงการเริ่มต้นบล็อก และเครื่องหมาย } แสดงการสิ้นสุดบล็อก

ตัวอย่างที่ 6 แสดงการทำงานเหมือนตัวอย่างที่ 5 แต่ให้ statement – 1 และ statement – 2 มีมากกว่า 1 คำสั่ง เพื่อแสดงการใช้บล็อก

```
main ( )
{
int i = 123, int l;
 scanf (" %d ", &1)
 if (i = = 1)
{
    printf ("***Right*** \n");
```

```
            printf ("_ _ _ _ _ _ _ _ _ _ _ ");
            }
            else
        {
            printf ("***Wrong*** \n");
            printf ("_ _ _ _ _ _ _ _ _ _ _ ");
        }
}
```

นอกจากนั้นใน statement – 1 และ statement – 2 อาจจะเป็น if ซ้อนกอีกได้ซึ่งเรียกว่า Nested If

รูปแบบ

```
if (Condition – 1)
    statement – 1;
else   if (Condition – 2)
    statement – 2;
else   if (Condition – 3)
    statement – 3;
        else
    statement – 4;
```

หมายความว่า ถ้า Condition – 1 จริงจะทำ statement – 1; แล้วออกไปทำคำสั่งที่อยู่ต่อจากคำสั่ง if เลย

ถ้า Condition – 1 ไม่จริงทำ Condition – 2

ถ้า Condition – 2 จริงจะทำ statement – 2 แล้วออกไปทำคำสั่งที่อยู่ต่อจากคำสั่ง if เลย

ถ้า Condition – 2 ไม่จริงทำ Condition – 3

ถ้า Condition – 3 จริงจะทำ statement – 3 แล้วออกไปทำคำสั่งที่อยู่ต่อจาก คำสั่ง if เลย

ถ้า Condition – 3 ไม่จริงจะทำ stamen – 4

ตัวอย่างที่ 7 แสดงการรับค่าของคะแนนมาจากแป้นพิมพ์ แล้วนำคะแนนนั้นมาหาค่าเกรดของนักเรียน โดยนำคะแนนมาพิจารณาเพื่อให้เกรดดังนี้

ถ้าคะแนน	≥ 85	จะได้เกรด	4
"	75-84	"	3
"	50-74	"	2
"	< 50	"	1

```
main ( )
{
  int   score, grade;
  printf ("Enter   Score :");
  scanf ("%d", & score);
  if (score > = 85)
        grade = 3;
  else if (score > - 50;)
       grade = 2;
  else
        grade = 1;
printf ("Grade %d", grade);
}
```

ตัวอย่างที่ *8* แสดงการเปลี่ยนค่าเลขฐานต่าง ๆ จากฐานสิบเป็นฐานสิบหก ฐานสิบหกเป็นฐานสิบ ฐานสิบเป็นฐานแปด และฐานแปดเป็นฐานสิบ โดยมีการให้เลือกว่าจะเปลี่ยนในกรณีไหนแล้วรับค่ามาจากแป้นพิมพ์มาทำการเปลี่ยนตามที่ต้องการ

```
/*Convert   Number
                  decimal -->   hexadecimal
```

```
                hexadecimal -->    decimal

                    decimal  -->   octal

                    octal  -->   decimal

*/

#include   <stdio.h>

main ( )

{

     int   choice ;

    int   value;

    printf ("Convert : \n");

    printf ("          1:decimal   to   hexadecimal \n");

    printf ("          2: hexadecimal    to   decimal \n");

    printf ("          3:decimal   to   octal \n");

    printf ("          4:octal   to   decimal   \n");

    printf ("enter   your   choice:");

    printf ("%d", &choice);

    if  (choice = = 1) {

          printf ("enter   decimal   value:");

          scanf ("%d", &value);

          printf ("%d   in   hexadecimal   is: %x", value, value);
```

```
        }
        else  if  (choice = = 2) {
                printf ("enter  hexadecimal  value : ");
                scanf ("%x ", & value);
                printf ("%x in decimal is : %d", value, value);
           }
        else  if  (choice = = 3)  {
                printf ("enter  decimal  value:");
                scanf ("%d", &value);
                printf ("%d in octal is : %o", value, value);
        }
        else  if  (choice = = 4)  {
           printf ("enter  octal  value:");
           scanf ("%o", &value);
           printf ("%o  in  decimal  is  %d",  value;  value);
        }
}
```

ในรูปคำสั่ง if เงื่อนไข สามารถมีมากกว่า 1 เงื่อนไขได้ซึ่งเรียกว่า Compound Condition โดยใช้เครื่องหมายตรรก && (AND), || (OR) , ! (NOT) เป็นตัวเชื่อม

ตัวอย่างที่ 9 แสดงการตรวจสอบค่าของตัวแปร sum ซึ่งรับเข้ามาทางแป้นพิมพ์ว่ามีค่าอยู่ระหว่าง 10 ถึง 20 หรือไม่ถ้าอยู่ระหว่าง 10 ถึง 20 จะพิมพ์คำว่า Right ถ้าไม่ใช่จะพิมพ์คำว่า Wrong

```
main ( )
{
   int  sum;
   scanf ("%d", & sum );
   if (sum > = 10 & & sum <= 20)
         printf ("Right");
   else
         printf ("Wrong");
}
```

ตัวอย่างที่ 10 แสดงการจรวจสอบค่าของตัวแปร sum ที่รับเข้ามาทางแป้นพิมพ์ว่ามีค่าเป็น 10 หรือ 20 ใช่หรือไม่ ถ้าใช่จะพิมพ์คำว่า Right ถ้าไม่ใช่จะพิมพ์คำว่า Wrong

```
main ( )
{
   int  sum;
   scanf ("%d", & sum);
   if (sum = = 10 || sum = = 20);
         printf ("Right");
```

```
        else
            printf ("Wrong");
    }
```

5.3 คำสั่ง switch

เป็นคำสั่งให้ไปทำคำสั่งใดคำสั่งหนึ่งตามที่ต้องการ โดยมีทางเลือกให้ไปทำคำสั่งได้หลาย ๆ ทางเลือกและจะเลือกไปทำคำสั่งใดจะขึ้นอยู่กับค่าของตัวแปร ที่ทำหน้าทั้งควบคุมการทำงานของคำสั่ง Switch นั้น

รูปแบบ

```
Switch (Variable) {
  case constant – 1:
      statement sequence;
  break;

 case constant – 2:
      statement sequence;
  break;
  .
  .
  .
 case constant – n:
  statement sequence;
 break;

default;
   statement sequence;
}
```

Variable หมายถึง ตัวแปรที่ทำหน้าที่ควบคุมการทำงานของคำสั่ง switch ซึ่งจะเป็นตัวแปรชนิด int char หรือ นิพจน์ก็ได้โดยค่าของตัวแปรนี้ถ้าตรงกับ Constant ที่อยู่ใน case ใดก็จะทำคำสั่งที่อยู่ใน case นั้นนั่นเอง

Constant – 1, Constant – 2,…. Constant – n เป็นเงื่อนไขที่ให้เลือกไปทำงานตามที่ต้องการ โดยจะต้องมีค่าเป็นค่าคงที่ ชนิด int, char เท่านั้น

Statement sequence หมายถึง กลุ่มของคำสั่งที่ต้องการให้ทำงานตามเงื่อนไขแต่ละ case

break เป็นคำสั่งให้ออกจากบล็อกของ case แต่ละ case แล้วไปทำคำสั่งที่อยู่ต่อจากคำสั่ง switch

default เป็นกรณีที่เมื่อมีค่า Variable ตรงกับเงื่อนไขใดใน case เลยก็จะลงมาทำคำสั่ง switch ไปทำคำสั่งอยู่ต่อจากคำสั่ง switch นั่นเอง

ตัวอย่างที่ 11 จงเขียนโปรแกรมแสดงการรับค่าของประเภทสินค้าและ ราคาสินค้าเข้ามาทางแป้นพิมพ์ แล้วนำมาคำนวณหาภาษีโดยคิดภาษีแยกตามประเภทสินค้า ดังนี้

ประเภทที่	1	เสียภาษี	10%	ของราคาสินค้า
"	2	"	15%	" "
"	3	"	17%	" "
"	4	"	40%	" "

ให้แสดงผลทางจอภาพเป็นประเภทสินค้า, ราคาสินค้า และภาษี

```
main ( )
   int  c;
   float v, tex;
      clrscr ( );
      printf ("Enter  class:");
```

```
scanf ("%d", &c);
printf ("Enter   value:");
scanf ("%f", &v);
   switch ( c )   {
   case 1:
      tex = v * 0.1;
      break;
   case 2:
      tex = v * 0.15;
      break;
   case 3:
      tex = v * 0.17;
      break;
   case 4:
      tex = v * 0.40;
      break;
   default:
       printf ("Unknown   class\n");
       exit ( 0 );
   }
```

```
        printf ("CLASS = &d \n VALUE = &.2f \n COSTOMS DUTY = %.2f",c,v,tex);
    }
```

ผลลัพธ์

```
    Enter   class  2:
    Enter   value : 3500
        CLASS = 2
        VALUE = 3500.00
        COSTOME = 525.00
    Enter   class  1:
    Enter   value : 1500
        CLASS = 1
        VALUE = 1500.00
        COSTOME = 150.00
Enter Class : 7
Enter value : 3000
Unknown class
```

5.4 คำสั่ง while

เป็นคำสั่งที่เขียนขึ้นเมื่อต้องการให้คอมพิวเตอร์ทำงานเป็นวงจร (loop)

รูปแบบที่ 1 รูปแบบที่ 2

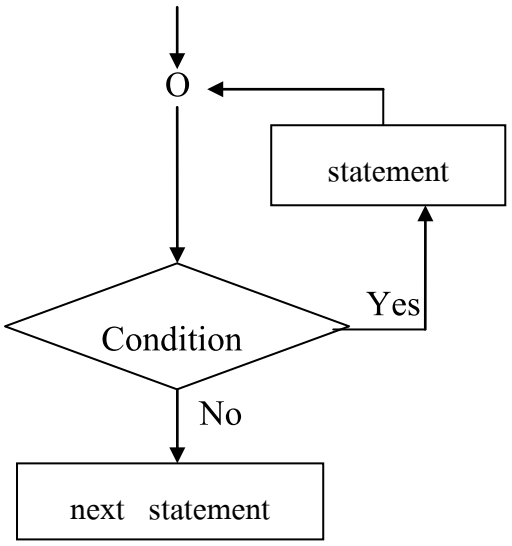
```
while (condition)
    statement;
next statement;
```

```
while (condition) {
    statement;
    statement;
}
next statement;
```

Condition หมายถึง เงื่อนไขที่ต้องให้คอมพิวเตอร์ตัดสินใจ ถ้าเงื่อนไขจริงจะทำคำสั่งที่อยู่ภายใน while ถ้าเงื่อนไขไม่จริงจะออกไปทำคำสั่งที่อยู่ต่อจากคำสั่ง while

Statement หมายถึง คำสั่งใด ๆ ในภาษา C ถ้ามีมากกว่า 1 คำสั่งต้องเขียนอยู่ในลักษณะของบล็อกดังรูปแบบที่ 2

Next statement หมายถึงคำสั่งใด ๆ ที่อยู่ต่อจากคำสั่ง while ลักษณะการทำงานของ while สามารถอธิบายในลักษณะผังงานได้ดังนี้



ตัวอย่างที่ 12 จงเขียนโปรแกรมแสดงค่า 1 ถึง 10 ออกมาบนหน้าจอภาพ

```
main ( )
{
```

```
    int   i = 0;
        while (++i < 10)
            printf ("I = %d \n", i);
}
```

ตัวอย่างที่ 13 จงเขียนโปรแกรมแสดงการหาค่าผลบวกของเลข 1 ถึง 100 ออกมาบนจอภาพ

```
main ( )
{
    int   i = 1;
    int   sum = 0;
        while (i < = 100) {
            sum = sum + i;
            i + +;
}
printf ("SUM = %d", sum);
}
```

ตัวอย่างที่ 14 ให้เขียนโปรแกรมเปลี่ยนค่าองศาเซลเซียส ให้เป็นองศาฟาเรนไฮท์ และโรเมอร์จาก 0 องศา ถึง 20 องศา โดยให้เพิ่มค่าองศาเซลเซียสขึ้นครั้งละ 1 ให้แสดงผลเป็นค่าองศาเซลเซียสมองศาฟาเรนไฮท์ และองศาโรเมอร์ทุกครั้ง

```
/*convert   celsius –   fahrenheit – romer */
main ( )
```

```
{
int   c = 0;
float f,r;
    printf ("Celsius : Fahrenheit : Romer: /n");
  while   (c , = 20 )
{
f = c / 5.*9.+32.;
r = c / 4.* 5.;
printf ("%6d %6.2f  %6.2f \n", c, f, r);
c + +;
    }
}
```

B :> radious

Celsius	:	Fahrenheit	:	Romer	:
0		32.00		0.00	
1		33.00		1.25	
2		35.00		2.50	
3		37.40		3.75	
4		39.20		5.00	
5		41.00		6.25	

6	42.80	7.50
7	44.60	8.75
8	46.40	10.00
9	48.20	11.25
10	50.00	12.50
11	51.80	13.75
12	53.60	15.00
13	55.40	16.25
14	57.20	17.50
15	59.00	18.75
16	60.00	20.00
17	62.00	21.25
18	64.40	22.50
19	66.20	23.75
20	68.00	25.00

ตัวอย่างที่ 15 จงเขียนโปรแกรมนับค่าที่รับเข้ามาทางแป้นพิมพ์ว่าเป็นค่าบวกกี่จำนวน ค่าลบกี่จำนวน และศูนย์กี่จำนวนโดยแสดงค่าที่รับเข้ามาไว้ตอนต้นของงานก่อนจึงพิมพ์ยอดรวมของค่าบวก, ลบ และศูนย์ไว้ตอนท้ายของรายงาน

```
main ( )
{
```

```
    float   in_key;
    int   p = 0, z = 0, n = 0, c =1;
        while (c < = 10) {
           printf ("Enter a value %d :", c);
              scanf ("%f", &in_key);
              if(in_key < 0);
              n++;
              else if (in_key = = 0.);
              z++;
             else
              p++;
    c++;
}
printf ("NEGATIVE = %d \n", n);
printf ("POSITIVE = %d \n", p)
printf("ZERO = %d \n", z)
}
Enter   a   value   1   :   1
Enter   a   value   2   :   2
Enter   a   value   3   :   0
```

```
Enter a value 4 : 0

Enter a value 5 : -6

Enter a value 6 : -7

Enter a value 7 : 23

Enter a value 8 : 2

Enter a value 9 : 0

Enter a value 10 : 7

NEGATIVE = 2

POSITIVE = 5

ZERO = 3
```

ตัวอย่างที่ 16 แสดงการใช้คำสั่ง while ทำให้เกิดการทำงานเป็นวงจรรวม 10 ครั้ง และมีการใช้คำสั่ง switch ที่ไม่มี break และ default ให้เห็นผลของการทำงาน

```
main ( )

{

  int t = 0;

    while (t < 10) {

     switch ( t ) {

     case 1:

          printf ("A");

          break;
```

```
            case 2:
                printf ("AB");
            case 3:
                printf ("C");
                printf ("DE");
                printf ("F\n");
                break;
            case 5:
            case 6:
                printf ("G");
                break;
            case 7:
            case 8:
            case 9:
                printf (".");
                }
              t++;
            }
        }
```

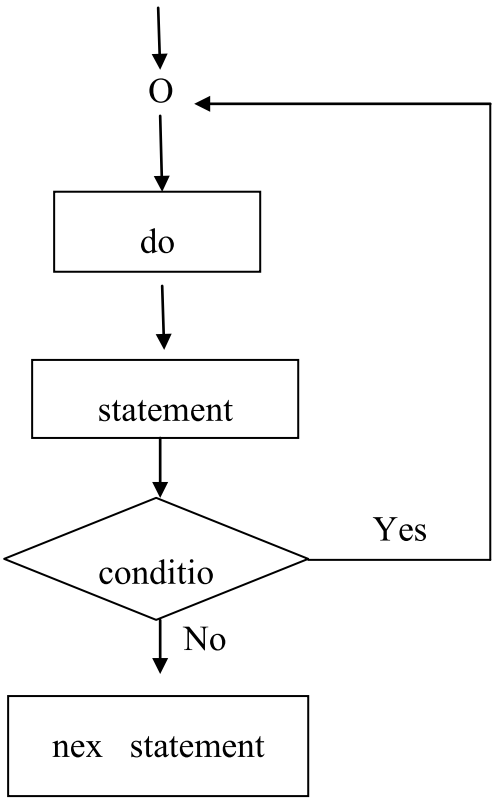
ผลลัพธ์

AABCDFF

CDEFGG…

5.5 คำสั่ง do-while

เป็นคำสั่งเขียนขึ้นเมื่อต้องการให้คอมพิวเตอร์ทำงานเป็นวงจรเหมือนกับ while แต่คำสั่ง do-while จะทำคำสั่งที่ต้องการก่อนแล้วจึงตรวจสอบเงื่อนไขถ้าเป็นจริงก็จะกลับขึ้นมาทำคำสั่งที่ต้องการใหม่จนกว่าเงื่อนไขจะไม่เป็นจริงก็จะออกจาก loop ของ do-while ลักษณะการทำงานจะขึยนเป็นผังงานดังนี้



รูปแบบที่ 1

```
do
  statement;
while (condition);
```

รูปแบบที่ 2

```
do {
  statement;
  statement;
:
}
```

statement หมายถึงคำสั่งใด ๆ ในภาษา C ถ้ามีมากกว่า 1 คำสั่งจะต้องกั้นบล็อกให้กับชุดของคำสั่งเหล่านั้นโดยใช้เครื่องหมาย {และ} ดังรูปแบบที่ 2

condition หมายถึงเงื่อนไขที่ต้องการให้คอมพิวเตอร์ตัดสินใจ ถ้าเงื่อนไขจริงก็จะกลับขึ้นไปทำคำสั่ง do ให้ถ้าเงื่อนไขไม่จริงก็จะออกนอกวงจร คือลงมาทำคำสั่งที่อยู่ต่อจาก while เลย

ตัวอย่างที่ 17 จงเขียนโปรแกรมหาค่าผลบวกของเลข 1 ถึง 100 โดยใช้คำสั่ง do……..while

```
main ( )
   int  i = 1;
   int  sum = 0;
        do  {
                sum = sum + i;
                i + +;
        }
        while ( i < = 100);
        printf ("SUM = %d", sum);
}
```

5.6 คำสั่ง for

เป็นคำสั่งให้มีการทำงานเป็นวงจรซ้ำ เหมือนกับ do-while โดยมีรูปคำสั่งดังนี้

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

expression หมายถึง คำสั่งที่ใช้สำหรับกำหนดค่าเริ่มต้นให้กับตัวแปรที่จะทำหน้าที่ควบคุมวงจรของคำสั่ง for

condition หมายถึงเงื่อนไขที่ใช้ในการควบคุมการทำงานของตัวแปรที่ทำหน้าที่ควบคุมวงจร โดถ้าเงื่อนไขจริงจะทำในวงจรแต่ถ้าเงื่อนไขไม่จริงจะออกจากวงจรทันที

increment หมายถึงค่าที่กำหนดให้เพิ่มขึ้นของตัวแปร ที่จะทำหน้าที่ควบคุมวงจรในแต่ละรอบ

statement หมายถึงคำสั่งใด ๆ ในภาษา C ถ้ามีมากกว่า 1 คำสั่งต้องกั้นบล็อกให้กับกลุ่มคำสั่งเหล่านั้นด้วย

ตัวอย่างที่ 18 แสดงการพิมพ์ค่า 1 ถึง 100 ออกมาทางจอภาพโดยใช้คำสั่ง for

```
main ( )
   int  x;
   for (x = 1; x < = 100; x + +)
   printf ("%d \n", x);
}
```

ตัวอย่างที่ 19 แสดงการเปลี่ยนค่าองศาเซลเซียสเป็นองศาฟาเรนไฮท์จาก 0 องศาถึง 100 องศาจากสูตร $F = \frac{9c}{5} + 32$

```
main ( ) {
      int  c, f;
```

```
        printf ("c       f");
            for (c = 0; c < = 100; c + +) {
                f = 9 *c/5+32
                printf ("%d %d", c, f);
        }
}
```

กฎการใช้คำสั่ง for

1. ค่าที่เพิ่มขึ้นในแต่ละรอบของตัวแปรควบคุมนั้นจะเป็นเท่าไรก็ได้ เช่น

```
for (x = 0;; x < = 100; x = x + 5)
    printf ("%d \n", x);
```

หมายความว่า จะพิมพ์ค่า x จาก 5, 10, 15 ไปเรื่อยจนถึง 100 นั่นคือ ค่าของ x เพิ่มขึ้นครั้งละ 5

2. ค่าของตัวแปรควบคุมอาจถูกกำหนดให้ลดลงก็ได้เช่น

```
for (x = 100; x > 0; x - -)
    printf (" % d  n  " ,x);
```

หมายควายว่า จะพิมพ์ค่า x จาก 100 , 99, 98, ลงไปเรื่อยจนถึง 0 นั่นคือค่าของ x จะถูกกำหนดให้ลดลงรอบละ 1 นั่นเอง

3. ตัวแปรควบคุมอาจเป็นชนิด character ได้เช่น

```
for  (ch = '; ch< = 'z'; ch + +)
    printf ("ASCII = %C DECIMAL = %d\n", ch, ch);
```

หมายความว่า ให้พิมพ์รหัส ASCII กับค่า DECIMAL ของตัวอักษร a ถึง z ออกมาให้บนหน้าจอ

4. ตัวแปรควบคุมสามารถมีได้มากกว่า 1 ตัวแปร เช่น

```
for (x = 0, y =0; x + y < 100; + + x, + +y)
```

```
printf ("%d \n", x + y);
```

หมายความว่า ตัวแปร x, y จะทะหน้าที่เป็นตัวควบคุมการทำงานของวงจรทั้ง 2 ตัวโดยจะหยุดทำงานเมื่อ x + y มีค่ามากกว่า 100 และค่า x, y จะเพิ่มขึ้นครั้งละ 1 เสมอ

5. ถ้ามีการละส่วนของ increment ในรูปแบบจะทำให้เกิดการทำงานเป็นวงจรโดยค่าของตัวแปรควบคุมจะไม่เปลี่ยนแปลงและจะออกการทำงานเป็นวงจรเมื่อเงื่อนไขเป็นเท็จเหมือนเดิม

```
for (x = 0; x! = 10;)
        {
        scanf ("%d", &x);
            printf ("PRINT &d", x );
        }
```

6. ถ้ามีการละทั้ง expression, condition, increment เลยจะเป็นการสั่งให้ทำงานในวงจร for โดยไม่รู้จบซึ่งถ้าต้องการให้จบก็ทำได้โดยใช้คำสั่ง break เข้ามาช่วย เช่น

```
for (;;)
        printf ("This loop will run forever \n");
```

หมายความว่า คอมพิวเตอร์จะพิมพ์ This loop will run forever ออกมาไม่รู้จบ

```
for (; ;)  {
        ch = getch ( );
            if(ch = = 'A')  break;
}
    printf ("you typed is 'A' " );
```

หมายความว่า จะมีการรับค่ามาเก็บไว้ในตัวแปร ch ไม่รู้จบจนกว่าค่าใน ch เป็น A จะออกจากวงจร ลงมาพิมพ์คำว่า you typed is 'A' ให้

7. ในคำสั่ง for สมารถมีคำสั่ง for ซ้อนอยู่ภายในได้อีก เช่น

```
for ( x = 1; x < = 3; x + +)  {
    printf ("x = %d \n", x);
      for (y = 1; y < = 5; y + +)
          printf ("y = %d \n", y)
    }
```

คำสั่ง break

เป็นคำสั่งให้ออกจากการทำงานของคำสั่ง case หรือออกการทำงานเป็นวงจร เพื่อให้ไปทำงานยังคำสั่งที่อยู่ต่อจากคำสั่งที่ให้ทำงานเป็นวงจรนั้นคือจะเป็น คำสั่งที่ใช้ภายในคำสั่ง switch, for, while, do while นั่นเอง

ตัวอย่างที่ 20

```
main ( )
{
  int  t;
  for (t = 0; t < 100, t + +)  {
        printf ("%d \n", t);
```

```
        if (t = = 10) break;

    printf ("END OF JOB");

}
```

จากตัวอย่างนี้จะเห็นว่าในวงจรของ for จริง ๆ จะต้อง loop อยู่ถึง 100 ครั้งแต่เนื่องจากมีคำสั่ง if ดังนั้น การทำงานจึงวนไม่ถึง 100 ครั้ง จะวนจน t มีค่าเท่ากับ 10 ก็จะออกจาก loop ด้วย คำสั่ง break มาทำคำสั่ง printf โดยพิมพ์ค่า END OF JOB มาให้

คำสั่ง Continue

เป็นคำสั่งที่ให้กลับไปทำยังคำสั่งแรกของคำสั่งควบคุม for, while, do while, ใหม่

ตัวอย่างที่ 21 จงพิมพ์ค่าเลขคู่จาก 0 ถึง 100 ออกมาบนหน้าจอภาพ

```
main ( )

{

int x;

for (x = 0; x < = 10; x + +)  {

    if (x%2)  continue

    printf ("%d \n", x);

  }

}
```

ผลลัพธ์

0

2

4 8

6 10

แบบฝึกหัด

1. จงหาที่ผิดของส่วนของโปรแกรมต่อไปนี้

   a) for (n = 10; n < = 9; n + +)

      printf ("Happy Day");

   b) int n;

      n = 5;

      while (n < 7)

         printf ("%d =", n);

      n + +;

2. เขียนส่วนโปรแกรมข้างล่างนี้ใหม่ให้ถูกต้อง

   a) …..

      switch (a)

      {

         case 1;

            printf ("Branch value is 1 \n");

```
        case 0;
                printf ("Branch value is 0 \n");
        case -1;
                printf ("Branch value is -1 \n");
    }
```

b) …..

```
    if (a < b)
        goto a1;
    else
        goto a2;
        x = 1;
    a1:
        x = 0;
        goto a3;
    a2:
        x = 2;
    a3:
        x = 3;
```

c) …..

```
    if (x < 1)
    {
```

```
    if (y > 2)
      xx = 0;
  }
    else
    {
      if (x < 10)
            {
              if (y > 10)
              xx = 1;
            }
      }
  …
```

3. จงเขียนโปรแกรมใช้ Newton – Raphson ประมาณค่าของรากที่สองโดย N เป็นจำนวนเลขที่ต้องการหาค่ารากที่สอง และ X_{n-1} เป็นค่าประมาณที่ n – 1 และ X_n เป็นค่าประมาณที่ n

สูตร $X_n = 0.5 (N / X_{n-1} + X_{n-1})$

4. จงเขียนโปรแกรมรับค่าจำนวนเต็มเข้ามาแล้วคำนวณหาค่าเฉลี่ย