บทที่ 7

การทำซ้ำ (Repetitive Statement)

โดยทั่วไปแล้วการทำงานของโปรแกรมคอมพิวเตอร์จะทำงานเรียงตามลำดับ ตั้งแต่สเตตเมนต์แรกถึงสเตตเมนต์สุดท้าย แต่เราสามารถให้คอมพิวเตอร์ทำงานซ้ำ ๆ ที่สเตตเมนต์ชุดหนึ่งได้ โดยใช้คำสั่งควบคุมให้ทำงานซ้ำ บางครั้งจะเรียกว่าคำสั่งลูป ในภาษาปาสคาลมีคำสั่งให้ทำงานซ้ำอยู่ 3 รูปแบบคือ

1. For Statement
2. While Statement
3. Repeat Statement

การทำงานของคำสั่ง For เป็นการทำซ้ำแบบเรียงลำดับ โดยเราสามารถระบุจำนวนครั้งที่ให้โปรแกรมทำงานซ้ำได้ คำสั่งทำงานซ้ำแบบ While โปรแกรมจะตรวจสอบเงื่อนไขก่อน ถ้าเงื่อนไขเป็นจริงจะทำสเตตเมนต์ที่กำหนดซ้ำ ส่วนคำสั่งทำซ้ำแบบ Repeat โปรแกรมจะทำสเตตเมนต์ที่กำหนดหนึ่งครั้งจากนั้นจะตรวจสอบเงื่อนไข ถ้าเงื่อนไขเป็นจริงจะทำสเตตเมนต์ที่กำหนดซ้ำอีก

7.1 ลูป For

การทำซ้ำแบบ for หรือลูป for จะเป็นการให้โปรแกรมทำซ้ำ โดยมีจำนวนครั้งที่จะทำแน่นอน เริ่มแรกโปรแกรมจะกำหนดค่าเริ่มต้นให้กับตัวแปรเริ่มต้น จากนั้นทำสเตตเมนต์และเพิ่มค่าหรือลดค่าในตัวแปรเริ่มต้น จากนั้นเปรียบเทียบเงื่อนไขว่าเท่ากับค่าสุดท้ายหรือยัง ถ้ายังไม่เท่าจะทำซ้ำ ถ้าเท่ากับค่าสุดท้ายจะเลิกทำ โดยรูปแบบของคำสั่งเป็นดังนี้

```
FOR <index> := <initial value> TO <final value> DO
        <statement >;
```

ค่า index บางครั้งจะเรียกว่าตัวแปรควบคุมลูป (loop control variable) ส่วนค่าเริ่มต้น (initial value) จะต้องน้อยกว่าค่าสุดท้าย (final value) สำหรับสเตตเมนต์ที่จะทำซ้ำ อาจเป็นสเตตเมนต์รวม (Compound Statement) ก็ได้แต่ต้องอยู่ภายใน begin กับ End; ผังงานของการทำคำสั่งลูป for เป็นดังนี้

ตัวอย่างเช่นถ้าเขียนคำสั่งดังต่อไปนี้

```
FOR  I:= 1  TO  5  DO
        Statement;
```

เริ่มแรกโปรแกรมจะใส่ค่าเริ่มต้น 1 ลงในตัวแปร I จากนั้นจะทำสเตตเมนต์ และเพิ่มค่าตัวแปร I ขึ้นหนึ่ง และตรวจสอบว่าค่าในตัวแปร I เท่ากับค่าสุดท้ายหรือยัง ถ้ายังให้ทำสเตตเมนต์ซ้ำ ถ้าเท่ากับค่าสุดท้ายแล้วให้หยุดทำ จำนวนรอบของคำสั่งสามารถคำนวณได้จาก ค่าสุดท้ายลบค่าเริ่มต้นและบวกด้วยหนึ่ง

ในการเพิ่มค่าให้กับตัวแปรควบคุมจะเพิ่มขึ้นเป็นลำดับ โดยอาจเป็นตัวเลข 1,2,3….. หรือตัวอักษร ‘A’,’B’,’C’, ดังนั้นการประกาศประเภทของตัวแปรควบคุมจะต้องให้สอดคล้องกับค่าของข้อมูลด้วย

ตัวอย่างที่ 7.1

```
PROGRAM  TEST;
VAR      I   :  Integer;
BEGIN
         FOR  I:= 1  TO  5  DO
         WRITELN( ‘Number  ‘ ,I );
END.
```

เมื่อรันโปรแกรมจะทำให้คอมพิวเตอร์พิมพ์ค่า 1 ถึง 5 ดังต่อไปนี้

```
Number  1
Number  2
Number  3
Number  4
Number  5
```

สำหรับการลดค่าตัวแปรควบคุมหลังทำคำสั่งซ้ำ รูปแบบของคำสั่งจะเป็นดังนี้

```
FOR  <index>  :=  <initial value>  DOWNTO  <final value>  DO
        <statement >;
```

โดยค่าเริ่มต้นจะต้องมากกว่าค่าสุดท้าย ตัวอย่างดังโปรแกรมต่อไปนี้

```
PROGRAM  TEST;
VAR      Num   : Integer;
BEGIN
         For  Num  :=  5  DownTo  1  Do
                  WRITELN (‘Num  =  ‘,Num);
END.
```

ผลลัพธ์จากการรันโปรแกรมจะเป็นดังนี้

Num = 5

Num = 4

Num = 3

Num = 2

Num = 1

โปรแกรมตัวอย่างต่อไปจะให้คอมพิวเตอร์หาค่ายกกำลังสองของเลข 1 ถึง 5 โดยประกาศตัวแปร I เป็น integer และใช้คำสั่ง For ทำซ้ำ 5 ครั้ง ในแต่ละครั้งจะหาค่ายกกำลังสองโดยใช้ฟังก์ชั่นของปาสคาลคือ Sqr(x)

ตัวอย่างที่ 7.2

```
PROGRAM TEST;
USES  CRT;
VAR  i  :  Integer;
BEGIN
         CLRSCR;
         WRITELN('i':10,'Square':10);
         FOR I := 1 to 5  DO
         WRITELN(I:10,Sqr(I):10);
END.
```

จากโปรแกรมจะเห็นว่าให้พิมพ์ตัวแปร I มีความกว้างเท่ากับ 10 และค่ายกกำลังสองมีความกว้างเท่ากับ 10 ผลลัพธ์จากการรันโปรแกรมจะเป็นดังต่อไปนี้

i	Square
1	1
2	4
3	9
4	16
5	25

เนื่องจากตัวแปรควบคุมของคำสั่ง For เป็นตัวแปรแบบลำดับ ดังนั้นค่าเริ่มต้นและค่าสุดท้ายถูกกำหนดให้เป็นตัวอักษรได้ ในโปรแกรมต่อไปจะให้คอมพิวเตอร์พิมพ์ตัวอักษร A ถึง Z

ตัวอย่างที่ 7.3

```
PROGRAM TEST;
USES  CRT;
VAR   i   :   char;
BEGIN
        CLRSCR;
        FOR I := 'A' to 'J'  DO
                WRITE(I:5);
END.
```

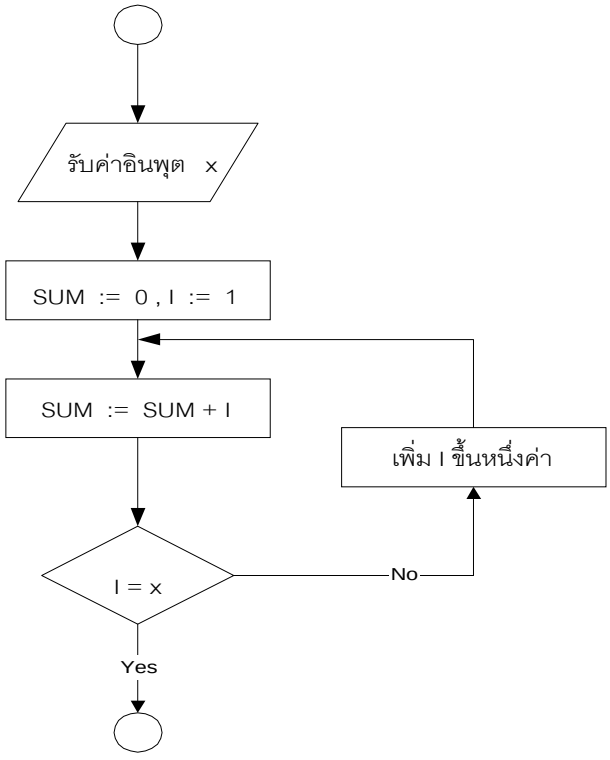
ตัวอย่างที่ 7.4 เป็นตัวอย่างการเขียนโปรแกรมให้แสดงเป็นตารางสูตรคูณ

```
PROGRAM TEST;
USES  CRT;
VAR   i   :   integer;
BEGIN
        CLRSCR;
        FOR i := 1 to 12  DO
        WRITELN(2:5,' x',i:2,' = ',2*i);
END.
```

จากโปรแกรมคอมพิวเตอร์จะแสดงผลเป็นสูตรคูณแม่ 2 การทำงานจะใช้การวนลูป 12 ครั้ง และให้ค่าคงที่ซึ่งเท่ากับ 2 คูณกับตัวแปรควบคุมลูป ผลจากการรันจะเป็นดังนี้

2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
2 x 11 = 22
2 x 12 = 24

ตัวอย่างโปรแกรมต่อไปเป็นโปรแกรมหาค่าผลบวกของตัวเลข เมื่อรันโปรแกรมเครื่องจะให้ใส่ค่าตัวเลข ถ้าใส่เลข 5 คอมพิวเตอร์จะทำการบวกเลขตั้งแต่ 1 ไปจนถึง 5 การทำงานของโปรแกรมจะประการตัวแปรสำหรับเก็บผลบวกให้ชื่อว่า SUM โดยเริ่มต้นให้ตัวแปรนั้นมีค่าเป็น 0 จากนั้นจะให้โปรแกรมทำงานซ้ำ โดยนำค่าตัวแปรควบคุมบวกกับค่า SUM และให้ผลลัพธ์เก็บใน SUM ตามเดิม การทำงานของโปรแกรมสามารถเขียนเป็นผังงานได้ดังนี้



ตัวอย่างที่ 7.5 ทำการบวกเลขตั้งแต่ 1 ถึงค่าที่อินพุตเข้าไป

```
PROGRAM TEST;
USES  CRT;
VAR   SUM,Num,i  :   integer;
BEGIN
        CLRSCR;
        SUM := 0;
        WRITE('Input Number = ');
        READLN(Num);
        FOR i := 1 to Num  DO
                SUM  :=  SUM + I;
        WRITELN('SUM  =  ',SUM);
END.
```

สำหรับโปรแกรมตัวอย่างที่ 7.5 จะใช้สำหรับบวกเลข 5 ค่า และแสดงออกทางจอภาพ ในโปรแกรมจะให้โปรแกรมทำซ้ำ 5 ครั้ง การทำงานแต่ละครั้งเครื่องจะให้ใส่ตัวเลข และนำตัวเลขที่รับเข้าไปบวกกับค่าตัวแปรชื่อ SUM และเก็บผลลัพธ์ไว้ใน SUM จากนั้นทำซ้ำและบวกไปเรื่อย ๆ จากโปรแกรมจะเห็นว่าในลูป For จะเป็นสเตตเมนต์รวม ซึ่งจะมี Begin กับ End คล่อมอยู่

```
PROGRAM TEST;
USES  CRT;
VAR   SUM,Num,i  :   integer;
BEGIN
        CLRSCR;
        SUM := 0;
        FOR i := 1 to 5  DO
        BEGIN
                WRITE('Input Number ',i,' = ');
                READLN(Num);
                SUM  :=  SUM + Num;
        END;
        WRITELN('SUM  =  ',SUM);
END.
```

คำถาม จากโปรแกรมต่อไปนี้เมื่อรันโปรแกรมผลลัพธ์ที่ได้จะเป็นอย่างไร

```
PROGRAM  TEST;
VAR   I , Square  :  Integer;
BEGIN
        FOR  I := 6  TO 8  DO
        BEGIN
                Square  :=  I * I;
                WRITELN ('This time I equals  ', I);
                WRITELN ('    its square  is  ', square)
        END;
        WRITELN ('So long ')
END.
```

ข้อควรระวัง

1. ถ้าใส่เครื่องหมาย ; หลัง do โปรแกรมจะทำงานผิดพลาด เมื่อคอมไพล์จะไม่แจ้งข้อผิดพลาดออกมา เพราะว่าคอมไพเลอร์จะมองว่าเครื่องหมาย ; เป็นการสิ้นสุดสเตตเมนต์ ตัวอย่างเช่นถ้าเขียนชุดคำสั่งเป็น

```
for  I := 1  to  4  do;
    Writeln ('John  Doe');
```

ผลลัพธ์จากการรันจะทำให้พิมพ์ John Doe เพียงครั้งเดียว

2. อย่าทำการเปลี่ยนตัวแปรควบคุมภายในลูป ตัวอย่างชุดคำสั่งต่อไปนี้จะทำให้โปรแกรมทำงานไม่หยุด

```
for  I := 1  to  10  do
    BEGIN
        WRITELN (I);
        I  :=  I + 2
    END;
```

3. ค่าเริ่มต้นและค่าสุดท้ายในลูปจะถูกกำหนดได้เพียงครั้งเดียวเท่านั้น แม้ว่าในลูป FOR จะทำการเปลี่ยนก็ไม่มีผลต่อโปรแกรม พิจารณาชุดคำสั่งต่อไปนี้

```
j  := 5;
for I := 1  to  j  do
BEGIN
        WriteIn (I);
        j  :=  j + 1;
END;
```

เมื่อรันชุดคำสั่งนี้ โปรแกรมจะทำงานในลูปเพียง 5 ครั้งตามเดิม

คำถาม จงเติมประโยคลงในที่ว่างในโปรแกรม ถ้าหากต้องการให้โปรแกรมทำงานต่อไปนี้

$1^2 + 2^2 + 3^2 + ... + 100^2$

```
PROGRAM  TEST;
VAR  I      :  Integer;
       Sum  :  longint;
BEGIN
        Sum  :=  0;
        For  I := 1  to 100  do
        ……..……;
        WRITELN ('Sum of 1st  100 squares  is  ', Sum)
END.
```

คำตอบ อาจเขียนได้เป็น

```
Sum  :=  Sum  +  I * I ;
```

หรืออาจใช้ฟังก์ชั่นยกกำลังมาช่วยก็ได้ ในโปรแกรมจะเห็นว่าประกาศตัวแปร Sum เป็นแบบ longint เนื่องจากผลลัพธ์มีค่าเกิน 32,767 ถ้าประกาศเป็น integer จะได้ค่าไม่ถูกต้อง

ตัวอย่างที่ 7.6 ถ้าหากต้องการนำลวดหนามมาล้อมเป็นรั้วสี่เหลี่ยม โดยมีลวดหนามยาว 200 เมตร อยากทราบว่าจะล้อมอย่างไรให้ได้พื้นที่มากที่สุด

วิธีทำ จากโจทย์เราสามารถเขียนสูตรการหาพื้นที่ได้เป็น X(100 – X)

100 – X

ในการเขียนโปรแกรมอาจให้ค่าเริ่มต้นกับ X และให้โปรแกรมวนลูปไปเรื่อย ๆ จนถึงค่าค่าหนึ่ง ในระหว่างวนลูปให้คำนวณพื้นที่ไปด้วย ถ้าพบค่ามากที่สุดให้เก็บค่านั้นเอาไว้ ซึ่งเขียนโปรแกรมได้ดังนี้

```
PROGRAM  MAXAREA;
VAR      x,area,maxsofar  :  Integer;
BEGIN
         WRITELN ('Value  of  x ' : 10,'Area ' : 10);
         Maxsofar  :=  -1;
         FOR  x  := 10  TO  45  DO
         BEGIN
                  Area  := x * (100 – 2 * x);
                  WRITELN (x : 10,area : 10);
                  If area > maxsofar Then maxsofer  :=  area
         END;
         WRITELN ('Maximum area is ',maxsofar)
END.
```

ผลลัพธ์จากการรันโปรแกรมจะเป็นดังนี้

Value of x	Area
10	800
11	858
..	..
..	..
44	528
45	450

Maximum area is 1250

โปรแกรมตัวอย่างที่ 7.7 เป็นการใช้คำสั่งลูป for และคำสั่งเงื่อนไข ให้โปรแกรมพิมพ์เลข 1 ถึง 100 โดยพิมพ์บรรทัดละ 10 ตัว ร่วม 10 บรรทัด การทำงานของโปรแกรมจะใช้คำสั่งลูป for โดยเริ่มนับตั้งแต่ 1 ถึง 100 ในแต่ละลูปจะนำค่าที่นับได้มาหารด้วย 10 จากนั้นใช้คำสั่งเงื่อนไขตรวจสอบว่าเศษที่ได้จากการหารเป็น 0 หรือไม่ ถ้าเป็นให้ขึ้นบรรทัดใหม่โดยใช้คำสั่ง WRITELN

ตัวอย่างที่ 7.7

```
PROGRAM TEST;
USES CRT;
VAR  x  : integer;
BEGIN
        CLRSCR;
        FOR x  := 1 TO 100 DO
        BEGIN
                WRITE(x:6);
                IF (x MOD 10) = 0 Then WRITELN;
        END
END.
```

ผลลัพธ์จากการรันโปรแกรมเป็นดังต่อไปนี้

โปรแกรมตัวอย่างที่ 7.8 เป็นโปรแกรมพิมพ์รหัสแอสกี (ASCII Code) รหัส 0 ถึง 255 โดยใช้คำสั่งแบบลูป For นับค่าตัวเลขตั้งแต่ 1 ถึง 255 และใช้คำสั่ง write พิมพ์รหัสแอสกี่ของตัวเลข โดยเปลี่ยนตัวเลขให้เป็นรหัสแอสกี่ด้วยคำสั่ง Chr(x)

ตัวอย่างที่ 7.8

```
PROGRAM TEST;
USES CRT;
VAR  x  : integer;
BEGIN
        CLRSCR;
        FOR x  := 32 TO 255 DO
        BEGIN
                WRITE(x:4,'=',chr(x));
                IF (x MOD 10) = 0 THEN WRITELN;
        END
END.
```

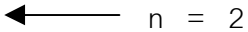
ผลลัพธ์จากการรันโปรแกรมเป็นดังต่อไปนี้

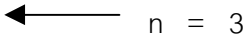
ประโยคคำสั่งลูป For สามารถนำมาซ้อนกันได้ โดยภายในลูป For สามารถเป็นลูป For ได้อีกเช่นกัน บางครั้งจะเรียกว่า Nested FOR ดังตัวอย่างชุดคำสั่งต่อไปนี้

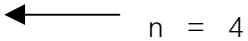
```
For  n := 2  to  4  do
        BEGIN
                For  I := 6  To  7  do
                    WRITELN (n, ‘  ‘,I);
                WRITELN (‘hello’)
        END;
```

ผลลัพธ์จากการรันชุดคำสั่งจะเป็นดังนี้

```
2   6
2   7
hello
3   6
3   7
hello
4   6
5   7
hello
```







ตัวอย่างที่ 7.9 เป็นการนำลูป For มาใส่ในสเตตเมนต์รวมของ For ลูปนอก ให้โปรแกรมพิมพ์เครื่องหมาย * เป็นรูปสามเหลี่ยม

ตัวอย่างที่ 7.9

```
PROGRAM TEST;
USES  CRT;
VAR   i,j   :  integer;
BEGIN
        CLRSCR;
        FOR i := 1  TO  10   do
        BEGIN
                FOR j := 1 TO  i  do
                WRITE('*');
                WRITELN;
        END;
END.
```

ผลจากการรันโปรแกรมเป็นดังต่อไปนี้

โปรแกรมตัวอย่างที่ 7.10 เป็นการใช้คำสั่ง if – then – else ซ้อนกันในลูป For ให้คอมพิวเตอร์พิมพ์อักขระกราฟิกเป็นรูปเส้นทแยงมุม

ตัวอย่างที่ 7.10

```
PROGRAM TEST;
USES CRT;
VAR  X , Y  :  Integer;
BEGIN
        CLRSCR;
        WRITELN;
        FOR Y := 1 TO  23  DO
        BEGIN
                FOR  X := 1 TO 23 DO
                IF (X = Y)
                        THEN  WRITE(CHR(219))
                ELSE
                        IF (X = 24 - Y)
                                THEN  WRITE(CHR(219))
                        ELSE
                        WRITE(CHR(176));
                WRITELN;
        END;
        END.
```

เมื่อรันโปรแกรมผลลัพธ์ที่ได้จะเป็นดังรูปต่อไปนี้

ตัวอย่างที่ 7.11 ตัวอย่างนี้จะแสดงการใช้ฟังก์ชั่นของเทอร์โบปาสคาลคือ Randomize,Random และ Inc โดยจะเขียนโปรแกรมให้วนลูปสุ่มตัวเลขขึ้นมา 500 ครั้ง

```
PROGRAM Rando4;
USES  CRT;
VAR     I  : Integer;
        j, Ones,Twos, Threes,Fours, Fives  :  LongInt;
BEGIN
        CLRSCR;
        RANDOMIZE;
        Ones    :=   0;
        Twos    :=   0;
        Threes  :=   0;
        Fours   :=   0;
        Fives   :=   0;
        FOR  j  :=  1   TO   500  DO
        BEGIN
                i := Random(5) + 1;
                WRITELN(i);
                Case i of
                1       :   Inc(Ones);
                2       :   Inc(Twos);
                3       :   Inc(Threes);
```

```
                4       :   Inc(Fours);
                else    Inc(Fives);
                END;  {end Case}
        END;
        CLRSCR;
        WriteLn('One was returned: ', Ones, ' Times.');
        WriteLn('Two was returned: ', Twos, ' Times.');
        WriteLn('Three was returned: ', Threes, ' Times.');
        WriteLn('Four was returned: ', Fours, ' Times.');
        WriteLn('Five was returned: ', Fives, ' Times.');
        WriteLn('Total = ', Ones + Twos + Threes + Fours + Fives);
        WriteLn;
        WriteLn('Press the Enter key to end this program.');
        ReadLn;
END.
```

ในการเขียนโปรแกรมด้วยเทอร์โบปาสคาลถ้าหากต้องการเพิ่มค่าให้ตัวแปรหนึ่งค่าเราสามารถใช้คำสั่ง INC เพิ่มค่าให้กับตัวแปรที่ตามมาได้ รูปแบบของคำสั่งเป็นดังนี้

```
INC(a,b);
```

จะเป็นการเพิ่มค่าให้ตัวแปร a ด้วยค่าในตัวแปร b โดยตัวแปร b นี้จะมีหรือไม่มีก็ได้ ถ้าไม่มี b จะเป็นการเพิ่มค่าตัวแปร a ด้วยค่า 1

ถ้าเขียนเป็น INC(I) เมื่อคอมพิวเตอร์ทำงานจะมีค่าเท่ากับการเขียนเป็น I := I + 1; ในโปรแกรมเมื่อคอมพิวเตอร์ทำงานจะสุ่มค่าขึ้นมา 500 ค่า จากนั้นจะนำค่าทุกค่าที่สุ่มได้มารวมกัน การสุ่มข้อมูลจะใช้ฟังก์ชั่น Random ซึ่งจะคืนค่าที่เป็นเลขจำนวนเต็มออกมา ตัวอย่างเช่นถ้าเขียนเป็น Random(10) จะสุ่มค่า 10 ค่า โดยมีค่าตั้งแต่ 0 ถึง 9 การใช้ฟังก์ชั่นนี้จะต้องเรียกใช้ฟังก์ชั่น Randomize ก่อน ถ้าหากไม่ใช้เมื่อโปรแกรมทำการสุ่มค่าจะทำให้ค่าที่ได้มีค่าเท่ากันทุกครั้งที่รันโปรแกรม

ตัวอย่างที่ 7.12 เป็นตัวอย่างการใช้คำสั่ง INC โดยให้โปรแกรมบวกค่าตั้งแต่ 100 ถึง 200

```
PROGRAM ForLoop3;
USES  CRT;
VAR     Total   : Integer;
```

```
        i     : Integer;
BEGIN
        CLRSCR;
        Total   :=   0;                    { ใส่ค่า 0 เป็นค่าเริ่มต้น }
        FOR  i := 100 TO 200 DO
              INC(Total,i);                { บวกค่าใน Total ด้วยค่าในตัวแปร I }
        WRITELN('The Total is ',Total);    { แสดงผลลัพธ์ของการบวก }
        READLN;
END.
```

ตัวอย่างที่ 7.13 เป็นตัวอย่างการใช้คำสั่ง INC เพิ่มค่าให้กับตัวแปร โปรแกรมจะให้รับข้อมูลทางแป้นพิมพ์เข้าไปครั้งละตัว จากนั้นจะแสดงว่าข้อมูลที่พิมพ์เข้าไปนั้นเป็นตัวอักขระภาษาอังกฤษตัวเล็กจำนวนเท่าใด

```
PROGRAM Counter1;
USES  CRT;
VAR    LCCnt    :   Integer;
        i        :   Integer;
        Ch       :  Char;
BEGIN
        CLRSCR;
        LCCnt  :=  0;
        WRITELN('Type any character and press Enter');
        FOR  i := 1 TO 10 DO
        BEGIN
              WRITE('Character  ',i,' = ');
              READLN(Ch);                  { รับอักขระ 1 ตัว เก็บในตัวแปร Ch  }
              IF (Ch >= 'a') AND (Ch <= 'z')   { ตรวจสอบว่า  Ch เป็นอักขระตัวเล็ก ? }
                    THEN  INC(LCCnt);      { เพิ่มค่าตัวแปรขึ้น 1 ค่า }
        END;
        WRITELN('Number of lowercase characters typed = ',LCCnt);
        READLN;
END.
```

7.2 ลูป WHILE (While ..Do Statement)

ประโยคคำสั่งลูปแบบ while จะใช้โปรแกรมทำงานซ้ำโดยตรวจสอบเงื่อนไขก่อน ถ้าเงื่อนไขเป็นจริงจะทำซ้ำ ลูปแบบนี้จะต่างจากลูปแบบ for เพราะจำนวนครั้งที่ทำซ้ำจะไม่แน่นอนขึ้นกับเงื่อนไข รูปแบบของคำสั่งเป็นดังนี้

```
while {test condition}  do
        begin
                {body of loop}
        end;
```

การใช้คำสั่งนี้จะเริ่มต้นด้วยคำว่า while และตรวจสอบเงื่อนไข ส่วนสเตตเมนต์ที่จะทำงานจะอยู่หลังคำว่า do ในการตรวจสอบเงื่อนไขนั้นจะใช้ตัวดำเนินการเปรียบเทียบแบบบูลีน ซึ่งเป็นนิพจน์ทางตรรกที่จะเป็นเงื่อนไขให้โปรแกรมทำงานซ้ำ การใช้งานเป็นดังชุดคำสั่งต่อไปนี้

```
n := 7;
while n >= 0 do
        begin
                writeln (n);
                n := n – 5;
                writeln ('Hi  ', n)
        end.
```

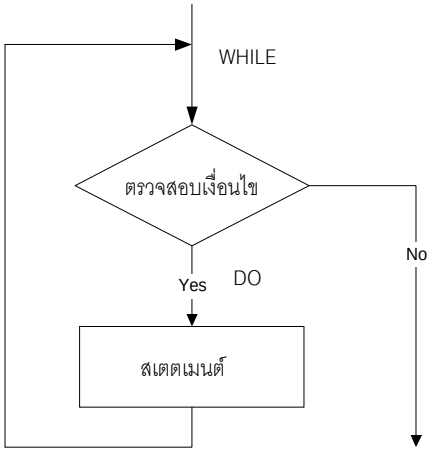
ผลลัพธ์ที่ได้จากการทำคำสั่งจะเป็นดังนี้

```
7
Hi  2
2
Hi  -3
```

ทำงานพิมพ์ลูปแรก

ทำงานพิมพ์ลูปที่สอง

ในลูปแรก n มีค่าเท่ากับ 7 ทำให้เงื่อนไขเป็นจริง โปรแกรมจะทำงานในลูปซึ่งจะทำให้ n มีค่าเป็น 2 ต่อมาโปรแกรมตรวจสอบเงื่อนไขเพื่อทำลูปที่สอง พบว่าเงื่อนไขเป็นจริงเมื่อทำงานลูปที่สองทำให้ n มีค่าเป็น –3 เมื่อโปรแกรมตรวจสอบเงื่อนไขพบว่าเงื่อนไขเป็นเท็จจึงไม่ทำลูปที่สาม การทำงานของคำสั่งลูป while – do เขียนเป็นผังงานได้ดังนี้



ในกรณีที่ต้องการให้โปรแกรมทำซ้ำที่มีจำนวนครั้งที่แน่นอน แต่ต้องการให้ตัวแปรควบคุมมีค่าเปลี่ยนไปที่ไม่ใช่ครั้งหนึ่งแบบลูป for เราสามารถนำลูป while มาแทนลูปแบบ for ได้ ตัวอย่างชุดคำสั่งต่อไปจะให้โปรแกรมทำงานโดยเพิ่มค่าตัวแปรควบคุมครั้งละสอง

```
I := 1;
while I <= 9 do
    Begin
        writeln (I);
        I := I + 2;
    End;
Writeln ('so long ');
```

เอาต์พุต

```
1
3
5
7
9
so long
```

ถ้าเปรียบเทียบลูปแบบ while กับลูป for จะเห็นว่าในการกำหนดค่าเริ่มต้นให้ตัวแปรควบคุมจะกำหนดก่อนคำสั่งทำลูป

จากที่ทราบมาแล้วว่าลูปแบบ for เราจะทราบจำนวนครั้งของการทำลูปที่แน่นอน ส่วนลูปแบบ while การทำลูปจะขึ้นกับเงื่อนไข แต่บางครั้งการทำงานลูปแบบ while อาจไม่มีที่สิ้นสุดได้ ดังชุดคำสั่งได้ดังต่อไปนี้

```
n := 1;
Sum := 0;
While Sum <> 10 do
      Begin
          n := n + 1;
          Sum := Sum + n;
      End;
Writeln (n, ' terms used. ');
```

จากชุดคำสั่งจะเห็นว่าจะเริ่มต้นด้วยการให้ตัวแปร n มีค่าเป็น 1 และ Sum มีค่าเป็น 0 โปรแกรมจะตรวจสอบว่า ถ้า Sum ไม่เท่ากับ 10 จะทำงานในลูป การทำงานในลูปแต่ละครั้งค่าของ Sum จะมีค่าดังนี้ 2,5,9,14 และไปเรื่อย ๆ ซึ่งจะทำให้การทำงานไม่มีที่สิ้นสุด

คำถาม จงเติมคำประโยคต่าง ๆ ลงในที่ว่างของโปรแกรม ถ้าหากต้องการให้คอมพิวเตอร์ทำงานต่อไปนี้ $12^2 + 15^2 + 18^2 + 21^2 + 24^2 + 27^2 + 30^2$

```
PROGRAM TEST;
VAR  I , Sum  : integer;
Begin
        Sum := 0;
        ……..
        While I ……. do
            Begin
                Sum := Sum + ( I * I );
                ……..
            End;
        Writeln ('Sum is ', Sum)
End.
```

คำตอบ จะเห็นว่าตัวแปร I จะต้องเริ่มจาก 12 ไปจนถึง 30 โดยเพิ่มค่าครั้งละ 3 ดังนั้นประโยคในที่ว่างจะเป็นดังนี้

```
I := 12;
While I <= 30 do
        I := I + 3
```

ตัวอย่างที่ 7.14 จงเขียนโปรแกรมคำนวณค่า $1^2 + 2^2 + 3^2 + \ldots\ldots$ โดยคำนวณไปเรื่อย ๆ ถ้าค่าเกิน 1000 ให้หยุดคำนวณ พร้อมทั้งบอกว่าค่าแรกที่ผลรวมเกิน 1000 เป็นค่าใด และเป็นเลขยกกำลังของค่าใด โดยแสดงผลดังนี้

```
Sum  first  goes  over  1000 when you add …… squared
Sum  is  ……
```

วิธีทำ จากโจทย์ เราสามารถเขียน Pseudocode ได้ดังนี้

```
initialize  n and Sum
while  sum  <= 1000  do
        Begin
                Increase  n  by  1
                Add  n^2 to Sum
        End
Print the results
```

การคำนวณสามารถเขียนเป็นโปรแกรมได้ดังต่อไปนี้

```
PROGRAM  Over1000;
USES CRT;
VAR  n , sum  :  integer;
BEGIN
        CLRSCR;
        n  :=  0;
        Sum  :=  0;
        While  Sum  <=  1000 do
                Begin
                        n := n + 1;
                        Sum  :=  Sum + (n * n)
                End;
        Writeln ('Sum first goes over 1000 when you add ',n, ' squared');
        Writeln ('Sum is  ',sum)
END.
```

ตัวอย่างที่ 7.15 ถ้าต้องการเขียนโปรแกรมให้คอมพิวเตอร์แสดงค่ายกกำลังสองของข้อมูล โดยแสดงข้อความดังต่อไปนี้

```
1.0     squared  is     1.00
1.2     squared  is     1.44
1.4     squared  is     1.96
1.6     squared  is     2.56
1.8     squared  is     3.24
2.0     squared  is     4.00
```

วิธีทำ จะเห็นว่าโปรแกรมจะทำการคำนวณซ้ำไปเรื่อย ๆ จนค่าข้อมูลมีค่าเป็น 2.0 ดังนั้นจะเขียนลูปได้ดังนี้

```
while  x <= 2  do
```

แต่ค่าของข้อมูลค่าสุดท้ายเป็น 2 ดังนั้นเขียนเงื่อนไขของ while ได้ดังนี้

```
while  x  <=  2  +  0.001  do
```

ค่า x จะเริ่มต้นที่ค่า x = 1 และการทำลูปแต่ละครั้งจะเพิ่มค่า x ขึ้นเท่ากับ 0.2 ดังนั้นจะเขียนโปรแกรมได้ดังต่อไปนี้

```
PROGRAM  SafeLoop;
VAR      x  :  real;
BEGIN
         x  :=  1;
         While  x  <=  2 + 0.001  do
                  Begin
                           Writeln (x : 0 : 1,  '   squared  is  ', x*x : 0 : 2);
                           x  :=  x + 0.2
                  End;
         END.
```

ตัวอย่างที่ 7.16 ตัวอย่างนี้จะใช้ฟังก์ชั่นลูปแบบ WHILE ในการพิมพ์เครื่องหมาย * เป็นกราฟค่าของ sin(y)

```
PROGRAM Sin1;
USES    Crt;
VAR
        x, y      : Integer;
        r         : Real;
BEGIN
        ClrScr;                         { ลบจอภาพ }
        y  :=  1;
        WHILE  y < 25 do                { ทำซ้ำถ้า y น้อยกว่า 25}
        BEGIN
                r   :=   Sin(y);
                x   :=   Round(r);      { เปลี่ยนค่าทศนิยมให้เป็นจำนวนเต็ม }
                GotoXY(x + 10, y);      { ย้ายเคอร์เซอร์ }
                Write('*');             { พิมพ์เครื่องหมาย * }
                y  :=  y + 1;
        END;
        ReadLn;
END.
```

จากโปรแกรมจะใช้ฟังก์ชั่น Gotoxy เพื่อย้ายตำแหน่งเคอร์เซอร์ไปยังตำแหน่ง x,y ที่กำหนดบนจอภาพ และโปรแกรมจะทำซ้ำจนกว่า y < 25 จะเป็นเท็จ ในลูปของการทำซ้ำค่า r ที่ได้จะเป็นเลขทศนิยมเนื่องจากค่าที่ได้จาก sin(y) เป็นทศนิยม แต่การใช้ฟังก์ชั่น Gotoxy จะต้องรับค่า x,y ที่เป็นเลขจำนวนเต็ม จึงใช้ฟังก์ชั่น Round แปลงเลขจำนวนจริงเป็นเลขจำนวนเต็ม

ตัวอย่างที่ 7.17 ตัวอย่างนี้จะแก้ไขโปรแกรมที่ผ่านมา โดยไม่ใช้ฟังก์ชั่น Gotoxy แต่จะใช้ลูปแบบ While มาซ้อนกัน โดยลูปในจะทำหน้าที่ย้ายตำแหน่งเคอร์เซอร์ในแนวแกน x

```
PROGRAM Sin2;
USES  Crt;
VAR     x,y,l     :  Integer;
        r         :  Real;
BEGIN
        ClrScr;
        y  :=  1;
```

```
        WHILE y < 100 DO                        { ทำซ้ำ 99 ครั้ง }
        BEGIN
                i   :=   1;
                r   :=    sin(y);
                x   := Round(r) + 12;
                WHILE  i < x   DO               { ย้ายตำแหน่งเคอร์เซอร์ }
                 BEGIN
                        Write(' ');             { พิมพ์ที่ว่าง }
                        INC(i);
                END;
                WriteLn('*');
                y   :=   y + 1;
                Delay(100);  { สามารถเปลี่ยนแปลงค่าเพื่อเปลี่ยนความเร็วได้ }
        END;
END.
```

7.3 ลูป Repeat-until

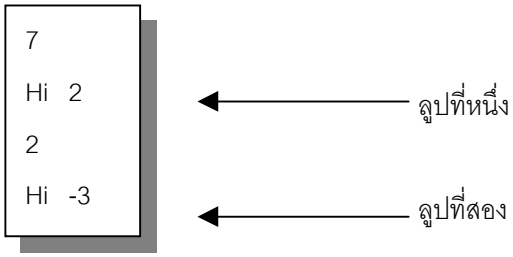
คำสั่งลูปแบบนี้จะทำการตรวจสอบเงื่อนไขภายหลังการทำงานในลูป โดยโปรแกรมจะทำลูปซ้ำไปเรื่อย ๆ ถ้าเงื่อนไขเป็นเท็จจะทำโปรแกรมซ้ำต่อไป จนกระทั่งเงื่อนไขที่เปรียบเทียบอยู่นั้นเป็นจริงจึงหยุดทำ เนื่องจากลูปแบบนี้จะตรวจสอบเงื่อนไขหลังจากทำลูป จึงทำให้ประโยคในลูปถูกทำหนึ่งครั้งเสมอ ซึ่งต่างจากกรณีของลูปแบบ While..do รูปแบบคำสั่งเป็นดังนี้

```
Repeat
        { body  of  Loop }
Until { test condition }
```

ตัวอย่างที่ 7.18 ถ้าคอมพิวเตอร์ทำชุดคำสั่งต่อไปนี้ เอาต์พุตที่ได้จะเป็นอย่างไร

```
n  :=  7;
repeat
        WriteIn (n);
        n  :=  n  -  5;
        WriteIn (‘Hi  ‘,n)
Until  n < 0;
```

เอาต์พุตที่ได้จะเป็นดังนี้



ตัวอย่างที่ 7.19 ถ้าคอมพิวเตอร์ทำชุดคำสั่งต่อไปนี้

```
n := 1;
Sum := 0;
Repeat
        Sum := Sum + (n * n);
        n := n + 2
Until n > 7;
Writeln ('Sum = ', sum);
```

เอาต์พุตที่ได้จะเป็น Sum = 84 การทำลูปแต่ละครั้งและค่า n จะเป็นดังนี้

Sum	การคำนวณ	n
0		1
1	$(0 + 1^2)$	3
10	$(1 + 3^2)$	5
35	$(10 + 5^2)$	7
84	$(35 + 7^2)$	9

จะเห็นว่าโปรแกรมจะทำไปจน n มีค่าเท่ากับ 9

ถ้าหากเปรียบเทียบการทำงานของคำสั่ง While กับ Repeat จะเป็นดังนี้

1. การใช้คำสั่ง Repeat จะมีการทำงานในลูปเป็นอย่างน้อยหนึ่งครั้งเสมอ แต่ถ้าเป็น while อาจจะไม่ทำเลยก็ได้
2. คำสั่ง while ถ้าเงื่อนไขเป็นจริงจะทำงานในลูป แต่ Repeat ถ้าเงื่อนไขเป็นจริงจะออกจากการทำลูป
3. ในกรณีที่สเตตเมนต์ของลูปเป็นสเตตเมนต์รวม ถ้าใช้คำสั่ง while คำสั่งต่าง ๆ จะอยู่ใน Begin กับ End แต่ถ้าเป็น Repeat จะไม่มี เนื่องจากมี Repeat และ until คลุมอยู่

ตัวอย่างที่ 7.20 ตัวอย่างนี้จะนำโปรแกรมที่ มาปรับปรุง โดยใช้ลูปแบบ Repeat โดยโปรแกรมจะวาดกราฟไปเรื่อย ๆจนกว่าจะมีการกดคีย์ใด ๆ

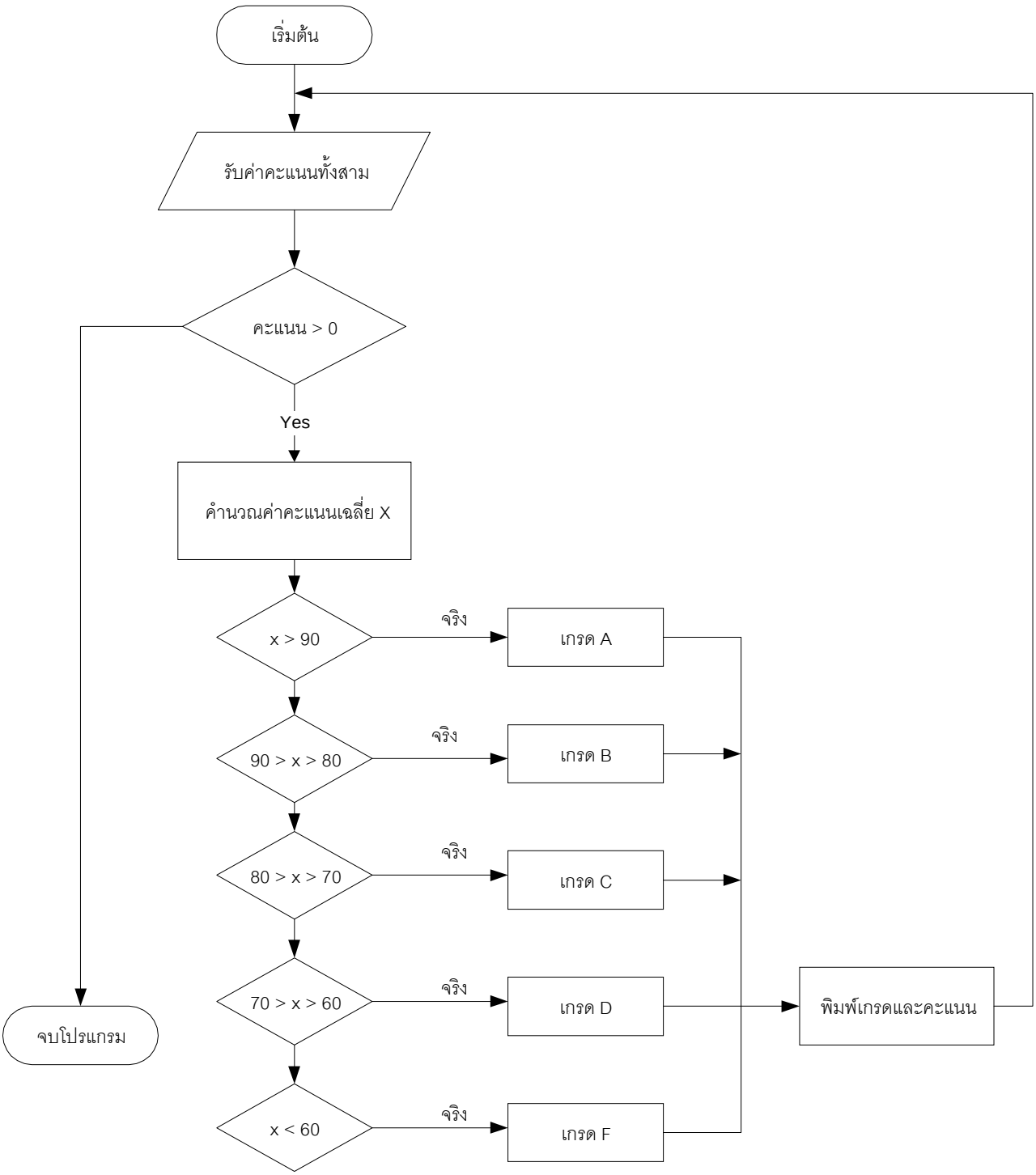
```
PROGRAM Sin3;
USES    Crt;
VAR     x,y,I   : Integer;
        r       : Real;
BEGIN
        ClrScr;
        y  :=  1;
        REPEAT
                i  :=  1;
                r  :=   sin(y);
                x  := Round(r) + 12;
                WHILE i < x  DO
                BEGIN
                        Write(' ');
                        INC(i);
                END;
                WriteLn('*');
                y  :=  y + 1;
                Delay(100);  { Adjust the parameter to change speed }
        UNTIL KeyPressed;
END.
```

ในโปรแกรมจะเห็นว่ามีการใช้ฟังก์ชั่น KeyPressed โดยฟังก์ชั่นนี้จะทดสอบการกดคีย์ ถ้าหากไม่มีการกดคีย์ใด ๆ จะทำให้ค่าที่ได้จากฟังก์ชั่นนี้เป็นเท็จ

ตัวอย่างที่ 7.21 ต่อไปเป็นโปรแกรมสำหรับคำนวณค่าเกรด โดยเมื่อรันโปรแกรมจะให้ใส่ค่าคะแนนสอบ คะแนนการบ้าน และคะแนนแบบฝึกหัด โดยการตัดเกรดจะเป็นดังนี้

คะแนน	**เกรด**
> 90	A
> 80 และ <90	B
> 70 และ <80	C
> 60 และ <70	D
< 60	F

ในโปรแกรมจะนำคะแนนสอบ คะแนนการบ้าน และคะแนนแบบฝึกหัดมารวมกัน โดยคะแนนการบ้านคิดเป็น 20% คะแนนสอบ 50% คะแนนแบบฝึกหัด 30 % ในโปรแกรมจะกำหนดให้แฟกเตอร์ของคะแนนทั้งสามประเภทสำหรับนำไปคิดคะแนนรวมเป็นค่าคงที่ดังนี้ HWWeight เป็น 0.2 TestWeight เป็น 0.5 และ ExamWeight เป็น 0.3



```
PROGRAM GradeAssignment;
USES  CRT;
CONST
        HWWeight        =  0.2;
        TestWeight      =  0.5;
        ExamWeight      =  0.3;

VAR
        Homework,Tests,Exam,Average    :  Real;
        Grade                          :  Char;

BEGIN
        CLRSCR;
        WRITE('Enter homework, Test and Exam Scores ( 0   to stop ) : ');
        READLN(Homework,Tests,Exam);
        WHILE Homework  >  0 DO
        BEGIN
                Average  :=  HWWeight * Homework + TestWeight * Tests + ExamWeight *
                Exam;
                CASE trunc(Average) DIV 10 OF
                        9,10     :  Grade  := 'A';
                           8     :  Grade  := 'B';
                           7     :  Grade  := 'C';
                           6     :  Grade  := 'D';
                     0,1,2,3,4,5 :  Grade  := 'F';
                END;
                WRITELN('Average  =  ',Average:5:1,' Grade  =  ',Grade);
                WRITELN;
                WRITE('Enter homework, test and exam score (0  to stop) : ');
                READLN(Homework,Tests,Exam);
        END;
END.
```

เมื่อรันโปรแกรมเครื่องจะให้ป้อนคะแนนทั้งสามค่า โดยการใส่แต่ละค่าให้กดคีย์เว้นวรรค ส่วนค่าสุดท้ายให้กดคีย Enter จากนั้นเครื่องจะคำนวณค่าคะแนนเฉลี่ยและเกรดทางหน้าจอ ถ้าหากป้อนคะแนนทั้ง 3 เป็นค่าศูนย์จะเป็นการออกจากโปรแกรม ตัวอย่างการรันเป็นดังนี้

คำถามท้ายบท

1. จงบอกผลลัพธ์จากการทำชุดคำสั่งโปรแกรมต่อไปนี้

1.1

```
for I := 1 to 3 do
        WRITELN ('Hello ');
        WRITELN ('good day');
WRITELN ('So long');
```

1.2

```
Sum := 0;
For I := 1 to 4 do
        Sum := Sum + I * I;
WRITELN (Sum);
```

1.3

```
for I := 1 to 4 do
   BEGIN
        Sum := 0;
        Sum := Sum + I;
    END;
WRITELN (Sum);
```

1.4

```
Sum := 0;
c := 0;
for I := 1 to 4 do
   begin
        Sum := Sum + I;
        if I > 2 then c := c + 1;
   end;
WRITELN (Sum / c : 4 : 2);
```

2. การทำคำสั่งต่อไปนี้จะพิมพ์คำว่า TWAT กี่ครั้ง

2.1
```
for I := 5 to 15 do
        Writeln('TWAT');
```

2.2
```
for := 1 to 4 do;
        Writeln('TWAT');
```

3. จงบอกเอาต์พุตจากการทำโปรแกรมต่อไปนี้

3.1
```
x := 30;
for I := 1 to 5 do
    Begin
        x := x – 2;
        Write ('x = ',x);
        if x mod 4 = 0
            Then Writeln ('OK')
             Else Writeln;
    END;
```

3.2
```
PROGRAM drill;
Var I , m , p : integer;
Begin
    m := 5;
    for I := 1 to 3 do
        Begin
            p := m – 2;
            m := m + p;
            p := p + 6;
            if m > p
                then writeln (p)
                else writeln (m)
        End;
    writeln (m, '  ',p);
End.
```

4.ถ้าหากต้องการให้โปรแกรมแสดงผลดังตารางนี้จะเขียนโปรแกรมอย่างไร

Yards	Inches
1	36
2	72
3	108
..	…
..	…
10	360

5. จงเขียนโปรแกรมหาค่าผลรวมต่อไปนี้

1 + 1/2 + 1/3 + 1/4 + 1/5 + …… + 1/50

6. จงเขียนโปรแกรมให้เอาต์พุตแสดงผลดังตารางต่อไปนี้

Number	Square	Cube
1	1	1
2	4	8
3	9	27
..	…	…
..	…	…
15	225	3375

7. ถ้าหากรันชุดคำสั่งต่อไปนี้โปรแกรมจะแสดงผลอย่างไร

7.1

```
x := 99;
repeat
    writeln (x);
    x := x + 1
until x > 1;
```

7.2

```
x := 99;
while x <= 1 do
    begin
        writeln (x);
        x := x + 1
   end;
```

8. ถ้าหากรันชุดคำสั่งต่อไปนี้โปรแกรมจะแสดงผลอย่างไร

8.1

```
n := 1;
while n < 9 do
   begin
         write (n, ' ');
         n := n + 2;
   end;
```

8.2

```
n := 1;
while n <= 9 do
   begin
      n := n + 2;
      write (n,' ');
   end;
```

9. ถ้าหากคอมพิวเตอร์ทำชุดคำสั่งต่อไปนี้ผลลัพธ์ที่ได้จะเป็นอย่างไร

```
x := 30;
y := 7;
repeat
   x := x - y;
   if x < y then
         begin
            x := x + 1;
            writeln (x, ' ',y)
         end
    else
         begin
            y := y + 1;
            writeln (y)
         end
until x < 10;
writeln ('x = ',x,'y = ',y);
```

10. จงบอกผลลัพธ์จากการทำชุดคำสั่งต่อไปนี้

```
for n := 1 to 3 do
   for I := 5 to 7 do
         write (n , ' ',I,' ');
```