บทที่ 13
ข้อมูลชนิดเซต
(Set)

<u>วัตถุประสงค์ของบทนี้</u>

1. ทราบถึงความหมายของเซต
2. ทราบถึงการประกาศข้อมูลชนิดเซต
3. ทราบถึงการให้ค่าแก่ตัวแปรชนิดเซต
4. ทราบถึงตัวกระทำของข้อมูลชนิดเซต
5. เครื่องหมายเปรียบเทียบที่ใช้กับข้อมูลชนิดเซต
6. สามารถเขียนโปรแกรมที่ใช้กับข้อมูลชนิดเซตได้

เซตคือ การจัดและรวบรวมสิ่งต่างๆชนิดเดียวกันเข้าด้วยกัน สิ่งต่างๆที่อยู่ในเซตเรียกว่า สมาชิก โดยปกติเซตจะใช้ทางคณิตศาสตร์ ซึ่งมีการกระทำต่างๆหลายลักษณะ เช่น

1. การหาสมาชิก (Membership) ในเซต
2. การหาเซตย่อย (Subset)
3. การหาเซตร่วม (Intersection)
4. การหาเซตรวม (Union)
5. การหาเซตต่าง (Differene)

จุดเด่นในภาษาปาสคาลอีกประการหนึ่งคือ ได้เตรียมให้ผู้ใช้สามารถใช้ข้อมูลที่มีโครงสร้างเป็นเซตได้ เช่น

```
if (A>=1) And (A<=100 ) then Write(A);
```

เงื่อนไขการทำงานนี้กรณีเป็นจริงทำงานตามคำสั่งหลัง then ผู้เขียนโปรแกรมสามารถกำหนดให้ตัวแปร A เป็นเซตของเลขจำนวนเต็ม การเปรียบเทียบเงื่อนไขโดยการตรวจสอบว่าค่า A เป็นสมาชิกในเซต 1 ถึง 100 หรือไม่เท่านั้น

13.1 การประกาศตัวแปรชนิดเซต

รูปแบบของข้อมูลชนิดเซต ใช้คำเฉพาะ Set ตามด้วย คำเฉพาะ of ตามด้วยชนิดของข้อมูลดังนี้

```
Type
        DayofMonth    = set of 1..31;
        Ch            = set of Char ;
        Sex           = set of (male,female);
        Month     = (Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec);
Var
        A , B   : DayofMonth ;
        C       : Ch;
        S       : Sex ;
        M       : Set of Month;
```

13.2 การให้ค่าแก่ตัวแปรชนิดเซต

ผู้เขียนโปรแกรมสามารถให้ค่าแก่ตัวแปรเซตโดยแสดงสมาชิกของเซตภายในวงเล็บปีกกาใหญ่ ดังนี้

M := [Mar,Apr,May]; หรือ M := [Mar..May] ก็ได้

กรณีที่เป็นเซตว่างเขียนดังนี้

S := [ ];

13.3 ตัวกระทำที่ใช้กับเซต

กำหนดให้ A และ B เป็นตัวแปรชนิดเซต

13.3.1 Intersection

หมายถึง การหาเซตร่วม ระหว่างเซต โดยใช้ ตัวกระทำ * เช่น A * B ผลของการทำงานได้เซตซึ่งสมาชิกในเซตนี้เป็นสมาชิกที่เหมือนกันของเซต A และ B

```
A := [1,3,5,7,8];
B := [3,6,8];
C := A * B ;
```

ผลของการทำงาน C := [3,8] ;

13.3.2 Union

หมายถึง เซตรวมของเซตสองเซต โดยใช้ตัวกระทำเครื่องหมาย + เช่น A + B โดยผลของการกระทำได้สมาชิกในเซตเป็นสมาชิกของเซตใดเซตหนึ่งหรือทั้งสองเซต

```
A := [1,3,5,7,8];
B := [3,6,8];
C := A + B ;
```

ผลของการทำงาน C := [1,3,5,6,7,8] ;

13.3.3 Difference

หมายถึงเซตต่างของเซตสองเซต ใช้ตัวกระทำเครื่องหมาย - เช่น A - B ได้เซตซึ่งเป็นสมาชิกของ A และไม่เป็นสมาชิกของ B ที่ต่างกัน

```
A := [1,3,5,7,8];
B := [3,6,8];
C := A - B ;
D := B - A ;
```

ผลของการทำงาน

```
C := [1,5,7] ;
D := [6];
```

13.3.4 ตัวกระทำ IN

ใช้สำหรับตรวจสอบข้อมูลว่าอยู่ในเซตหรือไม่ เช่น

```
If  Ch  IN  ['a'..'z']  then  ..... ;
If  N  IN  [1..100]  then  ..... ;
```

เป็นการเปรียบเทียบค่าของตัวแปร Ch ว่ามีค่าอยู่ในเซตของ a ถึง z หรือไม่กรณีเป็นจริงจะทำงานหลังคำสั่ง then อาจเขียนได้อีกลักษณะที่คุ้นเคยกันดีคือ

```
If  (Ch>='a') AND (Ch<='z')  then  .....;
If   (N>=1)  AND (N<=100)  then  .....;
```

13.3.5 เครื่องหมายเปรียบเทียบที่ใช้กับตัวแปรชนิดเซต

กำหนดให้ A และ B เป็นข้อมูลชนิดเซต

A = B ให้ผลเป็น TRUE เมื่อสมาชิกในเซต A เหมือนกับ เซต B ทุกประการ โดยไม่จำเป็นต้องมีลำดับเหมือนกัน เช่น [1,3,4] = [1,4,3] = [4,3,1]

A <> B ให้ผลเป็น TRUE เมื่อสมาชิกในเซต A กับเซต B ไม่เหมือนกัน เช่น

[1,4,5] <> [1,4] <> [1,5,8]

A >= B ให้ผลเป็น TRUE เมื่อเซต B เป็นเซตย่อยของ A เช่น

[1,4,9,5,3] >= [3,4]

A <= B ให้ผลเป็น TRUE เมื่อเซต A เป็นเซตย่อยของ B เช่น

[4,8] <= [1,2,3,4,5,6,7,8]

ตัวอย่างที่ 13.1

การสร้างเมนูเพื่อให้ผู้ใช้เลือกการทำงาน

```
Program  Test_Set(Input,output);
uses crt;
Var  Ch : Char;
Procedure  Find_Triangle;
        var Output :  real;
        begin
                clrscr;
                write('Base=');  readln(B);
                Write('High=');readln(H);
                Output := 1/2  * B *H ;
                writeln('Area = ',Output:7:2);
                readln;
        end;
```

ข้อมูลชนิดเซต

```
Procedure Find_Circle;
        Var Output : real;
        begin
                clrscr;
                write('radius='); readln(R);
                Output := Pi * R *R ;
                writeln('Area =',Output:7:2);
                readln;
        end;
begin
        Repeat
                clrscr;
                writeln('                    Menu ');
                Writeln('< press  1 >  Find area of triangle ');
                writeln('< press   2 >  Find area of circle ');
                writeln('< press  N >  End of Job ');
                Write('select choice ? ');readln(ch);
                if  ch = '1'          then   Find_Triangle
                else   if  ch = '2'  then  Find_Circle;
        Until   ch IN ['N', 'n' ];
End.
```

คำอธิบาย

โปรแกรมนี้เมื่อผู้ใช้รันจะปรากฏหน้าจอดังนี้

```
                    Menu
          < press  1 >  Find area of triangle
          < press   2 >  Find area of circle
          < press  N >  End of Job
          select choice ?  .....
```

เป็นเมนูให้เลือกการทำงานในกรณีที่ผู้ป้อน ตัวอักขระที่ไม่ใช่ 1 หรือ 2 หรือ N หรือ n เครื่องจะรอให้ผู้ใช้พิมพ์อักขระที่รับให้ถูกต้อง กรณีถ้าผู้ใช้โปรแกรมป้อน 1 เครื่องคอมพิวเตอร์จะทำงานโดยเรียกโพรซิเจอร์ Find_Triangle ทำงาน กรณีที่ผู้ใช้ป้อน 2 จะเรียกโพรซิเจอร์ Find_Circle ทำงาน กรณีป้อนอักขระ N หรือ n จะจบการทำงาน

ตัวอย่างที่ 13.2

รับอักขระทางแป้นพิมพ์ 1 บรรทัด จงหาว่ามีอักขระที่เป็นสระและอักขระที่เป็นตัวเลข อย่างละเท่าใด

```
Program  Count_Char (Input,Output) ;
Var
        Digit , Letter  :  Set  of  Char;                    .....(1)
        Ch                 : Char ;                          .....(2)
        Count_D ,Count_L : integer;                          .....(3)
Begin
        Letter := ['a','e','i','o','u','A','E','I','O','U'] ;   .....(4)
        Digit  := ['0'..'9'] ;                               .....(5)
        Count_D := 0; Count_L := 0;                          .....(6)
        while  not EOLN(Input)  do                           .....(7)
                begin
                read(ch);                                    .....(8)
                if  ch  in  Letter  then  Count_L := Count_L + 1      .....(9)
                else if  ch in Dagit then Count_D := Count_D + 1;     .....(10)
                End;
        writeln(' Letter Count = ', Count_L);                .....(11)
        Writeln('Digit  Count = ' , Count_D);                .....(12)
End.
```

คำอธิบาย

(1) ประกาศตัวแปร Letter และ Digit เป็นตัวแปรชนิดเซตของอักขระ

(2) ประกาศตัวแปร Ch เป็นตัวแปรชนิดอักขระ

(3) ประกาศตัวแปรที่ Count_L และ Count_D เป็นชนิดจำนวนเต็มเพื่อเก็บจำนวนของตัวอักขระที่เป็นสระ และ ตัวเลข ตามลำดับ

(4) ให้ค่าแก่ตัวแปรชนิดเซต Letter สมาชิกในเซตเป็นตัวอักขระที่เป็นสระในภาษาอังกฤษ ทั้งตัวเล็ก และตัวใหญ่

(5) ให้ค่าแก่ตัวแปรชนิดเซต Digit สมาชิกในเซตเป็นอักขระที่เป็นตัวเลข

(6) กำหนดค่าเริ่มต้นของตัวแปรที่นับจำนวนให้มีค่าเท่ากับ 0

(7) ขณะที่ไม่จบบรรทัดให้ทำ (8) มิฉะนั้นทำ (11)

(8) อ่านอักขระหนึ่งตัวอักษรจากแป้นพิมพ์

(9) ถ้าอักขระที่อ่านมานั้นอยู่ในเซตของ Letter ให้เพิ่มตัวนับ Count_L ขึ้น 1 ทำซ้ำ (7) กรณีเป็นเท็จให้ทำ (10)

(10) ถ้าอักขระที่อ่านมาอยู่ในเซตของ Digit ให้เพิ่มตัวนับ Count_D ขึ้น 1 ทำซ้ำ(7) กรณีเป็นเท็จให้ทำ (7)

(11) พิมพ์จำนวนของสระที่อ่านได้ทางจอภาพ

(12) พิมพ์จำนวนของตัวเลขที่อ่านได้ทางจอภาพ

13.4 คำสั่ง goto

โดยปกติไม่นิยมใช้คำสั่งนี้เพราะทำให้โปรแกรมไม่เป็นโครงสร้างที่ดี แต่ในบางกรณีการทำงานของโปรแกรมอาจมีการทำงานซับซ้อนมากๆ เช่น มีบล็อกซ้อนกันเป็นระดับ ซึ่งเป็นการแสดงการควบคุมที่ผิดปกติมากเกินไป ภาษาปาสคาลได้เตรียมคำสั่งกระโดดข้ามเพื่อช่วยในการเขียนโปรแกรม

ข้อจำกัดของคำสั่ง goto

1. คำสั่งนี้ไม่สามารถกระโดดจากภายนอก เข้าไปในคำสั่งที่เป็นโครงสร้างได้

```
for  I := 1 to 10 do
        begin
        .....
        .....
        1: .....        <------ ผิด
        end;
.....;
goto 1;
```

2. คำสั่งทุกคำสั่งกำหนด Label ได้ยกเว้นคำสั่ง goto
3. คำสั่งนี้ไม่สามารถกระโดดจากภายนอกเข้าไปในโปรแกรมย่อยได้
4. คำสั่งนี้อาจกระโดดจากโปรแกรมย่อยได้ แต่ต้องเป็นคำสั่งสุดท้ายในโปรแกรมย่อยนั้น

ตัวอย่างที่ 13.3

เป็นโพรซิเจอร์ที่แสดงถึงการใช้คำสั่ง goto

```
Procedure ReadMatrix(N:integer ;Var M : Matrix);
Label 9;
Var
        Data : Integer,
        I , J : integer,
Begin
for I := 1 to N do
        for J := 1 to N do
                Begin
                Read(Data);
                if Data >= 0 then M[I,J] := Data
                else
                        Begin
                        Writeln('Too few matrix elements');
                        got0 9;
                        End;
                read(Data):
                if Data >= 0 then
                        Writeln('Too many matrix elements);
                9 :
End;
```

คำอธิบาย

โพรซิเจอร์นี้มีการใช้คำสั่ง goto ในกรณีที่ข้อมูลที่รับมามีค่าน้อยกว่า 0 จะกระโดดข้ามคำสั่งอื่นมาที่ Label 9 เพื่อจบการทำงานของโพรซิเจอร์

ในการเขียนโปรแกรมนั้นควรวางสไตล์การเขียนที่ทำให้อ่านและเข้าใจได้ง่าย ควรพยายามหลีกเลี่ยงการใช้คำสั่ง goto ให้มากที่สุด

แบบฝึกหัดท้ายบท

1. จงเขียนโปรแกรมเพื่อตรวจสอบหมายเลขของการเล่นบิงโก โดยการเล่นนี้ผู้เข้าแข่งขันต้องทายหมายเลข 4 ตัวโดยไม่ซ้ำกันให้ถูกต้องโดนให้ผู้เข้าแข่งขันทายได้เพียง 3 ครั้งเท่านั้น กรณีที่หมายเลขตรงกับที่กรรมการตั้งไว้โดยไม่คำนึงถึงลำดับแสดงว่าได้รางวัล

2. จงเขียนโปรแกรมใช้สำหรับหาว่าสีที่เลือกเป็นสีใดสีหนึ่งของธงชาติหรือไม่ โดยสีที่เลือกให้กำหนดจำนวนสีขึ้นมาเอง