ส่วนที่ 1
ลักษณะทั่วไปของภาษาปาสคาล

ภาษาปาสคาลเป็นภาษาโปรแกรมที่มีความยืดหยุ่นในตัวเอง กล่าวคือไม่มีการกำหนดรูปแบบแน่นอนตายตัว คำสั่งหลายๆคำสั่งสามารถเขียนอยู่ในบรรทัดเดียวกันได้ ซึ่งแต่ละคำสั่งจะแยกจากกันด้วยเครื่องหมายเซมิโคลอน(;) ภาษาปาสคาลจัดเป็นภาษาโครงสร้าง มีคำสั่งดำเนินการเป็นบล๊อก (begin..end) ทำให้ผู้เขียนโปรแกรมสะดวกในการอ่านและแก้ไขโปรแกรมได้ง่ายกว่าภาษาโปรแกรมที่ไม่มีโครงสร้าง กรณีที่เกิดข้อผิดพลาดขึ้นภายในโปรแกรมผู้เขียนโปรแกรมสามารถหาจุดผิดพลาดได้ง่าย นอกจากนี้ภาษาปาสคาลยังมีส่วนของการประกาศตัวแปร กรณีผู้ใช้ป้อนข้อมูลเช่นชื่อตัวแปรผิด หรือใช้ชนิดข้อมูลผิดประเภท ตัวแปรภาษาหรือ Compiler จะสามารถตรวจสอบและแจ้งถึงข้อผิดพลาดได้ การสร้างคำใหม่ของภาษาปาสคาลผู้เขียนโปรแกรมสามารถสร้างได้ 2 ลักษณะ กล่าวคืออาจเป็นคำใหม่ในลักษณะของฟังก์ชั่นหรือ เป็นโปรแกรมการทำงานย่อยๆก็ได้ ผู้เขียนโปรแกรมสามารถเขียนในลักษณะของการเรียกตัวเองได้ ซึ่งทำให้โปรแกรมสั้น และกระทัดรัด ขึ้นอีกด้วย

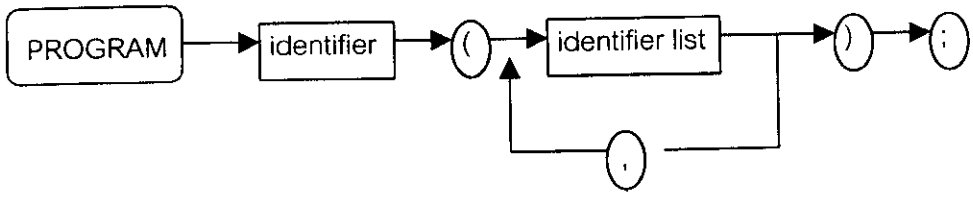
1.1 โครงสร้างโปรแกรมภาษาปาสคาล

ภาษาปาสคาลประกอบด้วยองค์ประกอบ 2 ส่วนคือ

1.1.1 Program heading เป็นจุดเริ่มต้นของโปรแกรม ต้องขึ้นต้นด้วยคำเฉพาะ Program ตามด้วยชื่อโปรแกรมซึ่งผู้เขียนโปรแกรมต้องตั้งชื่อเองเพื่อให้เหมาะสมหรือสอดคล้องกับการทำงานของโปรแกรม การตั้งชื่อผู้เขียนโปรแกรมจะตั้งชื่อซ้ำกันไม่ได้ ต้องเริ่มต้นด้วยตัวอักษรภาษาอังกฤษตัวใหญ่หรืออักษรภาษาอังกฤษตัวเล็กก็ได้ หรือขีดล่าง ตัวต่อไปเป็นอักษรที่เป็นตัวเลขหรืออักษรภาษาอังกฤษก็ได้หรือขีดล่าง ห้ามเป็นอักษรพิเศษอื่นๆ ความยาวไม่ควรยาวมากเกินไป และต้องไม่ใช่คำเฉพาะหรือคำที่ใช้ในหน้าที่อื่นๆของโปรแกรม

IT 257 (H) 1

รูปแบบของ Program heading เป็นดังนี้



ตัวอย่าง Program EX1;

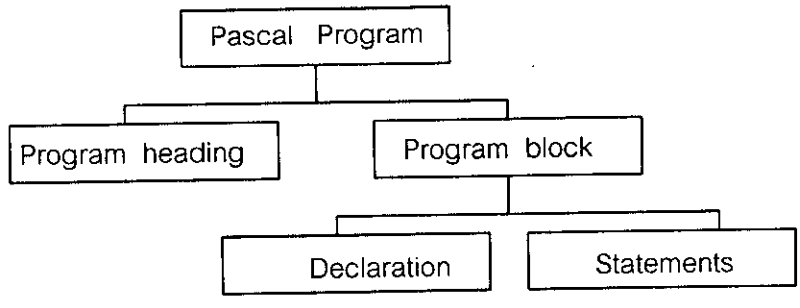
Program Test1(input,output);

1.1.2 Program block เป็นส่วนของโปรแกรมซึ่งแบ่งออกเป็น 2 ส่วนย่อยๆคือ

1.1.2.1 Declaration เป็นส่วนของการกำหนดและประกาศตัวแปรต่างๆ เช่นการประกาศค่าคงที่ การประกาศตัวแปร การกำหนดชนิดของข้อมูลใหม่ การสร้างฟังก์ชั่น การสร้างโพรซิเจอร์ เป็นต้น ซึ่งส่วนนี้เป็นส่วนที่บอกให้โปรแกรมทราบว่าเราใช้ตัวแปร ค่าคงที่ รวมทั้งสร้างคำใหม่อะไรบ้าง ในกรณีที่ผู้เขียนโปรแกรมป้อนหรือคีย์คำสั่งผิด ตัวแปลภาษาสามารถตรวจสอบได้ ทำให้เกิดความผิดพลาดน้อยลง

1.1.2.2 Executable statement เป็นส่วนของคำสั่งที่สั่งให้เครื่องปฏิบัติงาน ซึ่งเป็นคำสั่งที่ทำงานเป็นลำดับ หรือทางเลือกในการทำงาน หรือการทำซ้ำเป็นขั้นตอนในการดำเนินงานจนกระทั่งได้ผลลัพธ์ที่ต้องการ

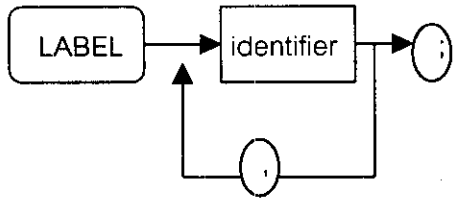
โครงสร้างโปรแกรมภาษาปาสคาลเป็นดังนี้



1.2 ตัวอย่างการประกาศและกำหนดตัวแปรต่าง ๆที่ใช้ในโปรแกรม

1.2.1 การประกาศ label

รูปแบบการประกาศ label

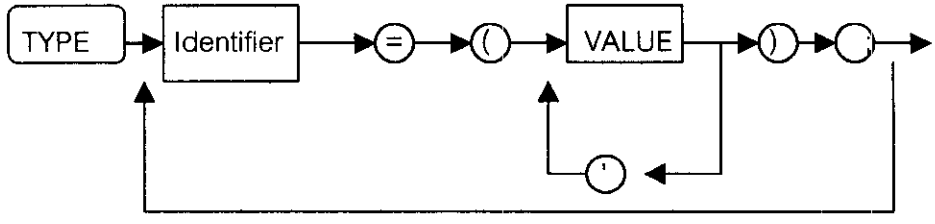


ตัวอย่าง LABEL exit,start;

เป็นการประกาศตำแหน่งอ้างอิงในโปรแกรมโดยมีการใช้คำสั่ง goto เช่น คำสั่ง goto exit การดำเนินงานของโปรแกรมจะกระโดดข้ามการทำงานไปทำคำสั่งหลัง exit ทันที

1.2.2 การกำหนดชนิดข้อมูลใหม่

รูปแบบการกำหนดชนิดข้อมูลใหม่



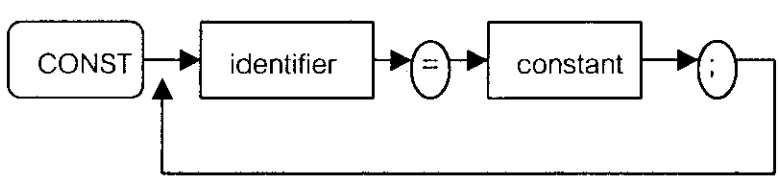
ตัวอย่าง

```
TYPE    Month = ( jan ,feb ,Mar ,Apr, May , Jun , Jul ,Aug ,Sep ,Oct
                  ,Nov,Dec);
        A     =   array [1..4]  of   integer;
        B     =   record
          Name  :  string [9] ;
          Code   :  string[9];
          Score  :  real ;
        End;
```

ส่วนของการกำหนดชนิดข้อมูลใหม่จะกำหนดในส่วนของ TYPE ข้อมูลใหม่นั้นเป็นข้อมูลที่โปรแกรมเมอร์ออกแบบโครงสร้างขึ้นมาใหม่อาจเป็นข้อมูลชนิดอาเรย์ หรือเรคอร์ดก็ได้ การประกาศข้อมูลใหม่นี้มีประโยชน์ในกรณีมีการส่งผ่านข้อมูลไปยังโมดูลหรือโปรแกรมย่อยเป็นข้อมูลที่เป็นโครงสร้างรวมทั้งนำกลุ่มของข้อมูลที่มีความสัมพันธ์กันมาจัดเก็บไว้รวมกันเพื่อประโยชน์ในการจัดเก็บและปฏิบัติการกับข้อมูลได้ง่าย

1.2.3 การกำหนดค่าคงที่

รูปแบบการกำหนดค่าคงที่



ตัวอย่าง

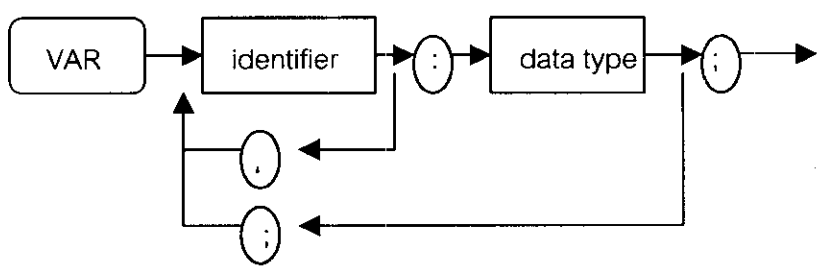
```
Const   Student  =  50 ;
        Tax      =  2.25;
```

```
Flag    =  True;
C       =  'Y' ;
```

การกำหนดค่าคงที่จะกำหนดให้กับตัวแปรในส่วนของ Const โดยค่าของตัวแปรเหล่านี้จะมีค่าคงที่ตลอดโปรแกรม ค่าคงที่ที่กำหนดอาจเป็นค่าที่เป็นตัวเลข ค่าที่เป็นอักขระหรือกลุ่มของอักขระ ค่าทางตรรกก็ได้

1.2.4 การประกาศตัวแปร

รูปแบบการประกาศตัวแปร



ตัวอย่าง

```
VAR  A , B   :  integer ;
       C     :  real ;
       X     :  array[1..5]  of  integer ;
       Y     :  array [1..5 ,1..5 ]  of  real;
       Z     :  Boolean;
```

ตัวแปรทั้งหมดที่ใช้ในโปรแกรมจะต้องมีการประกาศชนิดของตัวแปรในส่วนของ VAR ตัวแปรที่กำหนดนี้จะมีค่าจากการทำงานของโปรแกรม การประกาศเป็นการจองเนื้อที่ในหน่วยความจำเพื่อเก็บข้อมูลเท่านั้น ค่าของตัวแปรต่างๆ เหล่านี้ไม่คงที่ สามารถเปลี่ยนแปลงได้ตามการทำงานของคำสั่งในโปรแกรม โดยจะเก็บค่าสุดท้ายในการทำงานเสมอ

1.3 ชนิดของข้อมูล

1.3.1 Integer เป็นชนิดข้อมูลที่เป็นเลขจำนวนเต็ม ซึ่งเป็นตัวเลขที่นำไปใช้ในการกำหนดค่าและการคำนวณ ตัวกระทำที่ใช้ในการคำนวณได้แก่ บวก(+) ลบ(-) คูณ(*) และหาร ซึ่งได้ค่าเป็นเลขจำนวนเต็ม (mod การหารได้เป็นเลขจำนวนเต็มเศษ, div การหารได้เป็นจำนวนครั้งที่หารได้กรณีที่หารไม่ได้มีค่าเท่ากับ 0) ฟังก์ชั่นที่ใช้ในการคำนวณที่ได้ผลลัพธ์เป็นเลขจำนวนเต็มเช่น ฟังก์ชั่น ABS(x) เป็นฟังก์ชั่นที่หาค่าสัมบูรณ์ของ X ,ฟังก์ชั่น SQR(X) เป็นการหาค่ากำลังสองของ X , ฟังก์ชั่น ROUND(X) หรือ TRUNC(X) เป็นการปัดเศษขึ้นและปัดเศษทิ้งของค่า X เป็นต้น การใช้ค่าของตัวเลขจำนวนเต็มนั่นในโปรแกรมส่วนมากจะใช้สำหรับประกาศตัวแปรที่ทำหน้าที่นับ หรือการควบคุมในการทำงานวนรอบ หรือเป็นจำนวนที่ใช้ในการคำนวณ

1.3.2 Real เป็นข้อมูลที่เป็นจำนวนจริง ซึ่งส่วนมากเป็นข้อมูลที่มีจุดทศนิยมใช้ในการคำนวณ ตัวกระทำที่ใช้ในการคำนวณได้แก่ บวก(+) ลบ(-) คูณ(*) หาร (/) ได้ผลลัพธ์เป็นเลขทศนิยม สำหรับฟังก์ชั่นที่ให้ผลลัพธ์เป็นเลขทศนิยมประกอบด้วย ฟังก์ชั่นทางตรีโกณมิติ ฟังก์ชั่นหาค่าลอการิทึม ฟังก์ชั่นหารากที่สองของ X เป็นต้น ส่วนมากในการคำนวณทางด้านคณิตศาสตร์ และสถิติ จะมีค่าเป็นเลขจำนวนจริง

1.3.3 Boolean เป็นข้อมูลที่ใช้ในการเปรียบเทียบผลของการทำงานได้ค่าเป็นจริง (true) หรือ เท็จ (false) ค่าใดค่าหนึ่ง ส่วนมากในการเขียนโปรแกรมจะใช้นิพจน์ทางบูลีนในการเปรียบเทียบค่าเพื่อเลือกการทำงาน ตัวกระทำในนิพจน์บูลีนได้แก่ มากกว่า(>) น้อยกว่า(<) มากกว่าหรือเท่ากับ(>=) น้อยกว่าหรือเท่ากับ (<=) เท่ากับ(=) และไม่เท่ากับ(<>) นอกจากนี้การเปรียบเทียบยังสามารถนำนิพจน์บูลีนมาเปรียบเทียบกันได้โดยใช้ตัวกระทำ และ(and) หรือ(or) ไม่(not) ซึ่งผลของการทำงานได้ผลลัพธ์เป็นจริงหรือเท็จเช่นกัน เช่น

A := 7 <> x ;

B := (N>=1) AND (N<=99) ;

การเปรียบเทียบนี้มีลำดับการทำงานก่อนหลังเช่นกันกล่าวคือจะกระทำ not -> and -> or กรณีของการเปรียบเทียบของข้อมูลจะกระทำเป็นลำดับสุดท้าย สังเกตุว่าการเปรียบเทียบข้อมูลจะต้องใส่วงเล็บเสมอ สำหรับตัวกระทำ AND นั้น จะให้ผลลัพธ์เป็น TRUE เมื่อนิพจน์ที่เปรียบเทียบเป็น TRUE ทั้งคู่ กรณีอื่นๆมีค่าเป็น FALSE สำหรับ OR ให้ผลลัพธ์เป็น FALSE เมื่อนิพจน์ที่เปรียบเทียบเป็น FALSE ทั้งคู่กรณีอื่นๆมีค่าเป็น TRUE

1.3.4 Char เป็นข้อมูลชนิดอักขระ 1 ตัวอักษรโดยตัวอักขระนี้เป็นตัวเลข ตัวอักษรภาษาอังกฤษ หรืออักขระพิเศษใดก็ได้ แม้กระทั่งช่องว่าง 1 ช่อง ก็ได้แต่อักขระเหล่านี้ต้องเขียนอยู่ในเครื่องหมาย '' เช่น 'A' , ' ' , '*' เป็นต้น ข้อมูลลักษณะนี้โดยมากนำมาใช้ในการเลือกโดยให้ผู้ใช้เลือกสิ่งที่ต้องการจากเมนู เช่นต้องการทำงานต่อหรือไม่ Y/N ผู้ใช้เพียงกดแป้น Y หรือ N ลักษณะแบบนี้เป็นต้น นอกจากนี้อักขระต่างๆเหล่านี้สามารถนำมาเปรียบเทียบกันได้ให้ผลลัพธ์เป็นจริงและเท็จค่าใดค่าหนึ่ง โดยการเปรียบเทียบค่านั้นภาษาปาสคาลจะกระทำโดยอัตโนมัติโดยเปรียบเทียบจากรหัสแอสกี้ที่แทนอักขระเหล่านี้นั่นเอง

1.3.5 String เป็นกลุ่มของอักขระ ซึ่งส่วนมากผู้ใช้มีการป้อนเป็นกลุ่มอักขระที่ยาวๆ เช่นชื่อ-สกุล ที่อยู่ เบอร์โทรศัพท์ เป็นต้น อักขระที่เป็นกลุ่มสามารถเปรียบเทียบกันได้เช่นกันโดยภาษาปาสคาลจะเปรียบเทียบและให้ผลลัพธ์โดยอัตโนมัติ กรณีที่ความยาวน้อยกว่าจะมีค่าน้อยกว่าข้อความที่ความยาวมากกว่า สังเกตุจาก dictionary จะเรียง A->B->...->Z จากข้อความสั้นไปถึงข้อความที่ยาว สำหรับใน Standard pascal จะไม่มีข้อมูลชนิด String กรณีที่เป็นกลุ่มของอักขระผู้เขียนโปรแกรมต้องประกาศเป็นอาเรย์ของอักขระแทน เช่น

VAR NAME : ARRAY[1..20] of CHAR;

1.3.6 Subrange เป็นแบบย่อยของชนิดข้อมูลที่มีลำดับก่อนและลำดับหลังแน่นอน ที่นิยมใช้ในโปรแกรม โปรแกรมเมอร์จะประกาศตัวแปรเป็นแบบย่อยของ Integer และ Char เช่น

```
VAR   Month  : 1..12 ;
      Day    : 1..31 ;
      Letter : 'A'..'Z' ;
```

ข้อมูลชนิดนี้จริงๆแล้วเป็นการกำหนดขอบเขตของข้อมูลให้แคบลงนั่นเอง เพื่อง่ายต่อการตรวจสอบความผิดพลาดของข้อมูล ถ้าเราทราบอยู่แล้วว่าเดือนมีเพียง 12 เดือนเท่านั้น ก็จำกัดขอบเขตให้ค่าต่ำสุดเริ่มจาก 1 สูงสุดมีค่าเป็น 12 ทำให้การตรวจสอบความผิดพลาดของข้อมูลนั้นง่ายยิ่งขึ้น

1.3.7 Array หรือข้อมูลแถวอันดับ เป็นโครงสร้างข้อมูลที่เป็นโครงสร้าง ซึ่งเกิดจากการนำข้อมูลในรูปแบบมาตรฐานมาจัดกลุ่มข้อมูลให้มีลักษณะเป็นโครงสร้างที่เก็บอยู่ภายในหน่วยความจำอย่างต่อเนื่อง ทำให้โปรแกรมมีขนาดเล็กลง การอ้างถึงตัวแปรต่างๆนั้นง่ายขึ้น ข้อมูลแถวอันดับมีหลายลักษณะที่นิยมใช้กันส่วนมาก มี 2 ลักษณะคือ

1. ข้อมูลแถวอันดับ 1 มิติ เป็นการจัดเก็บข้อมูลที่ต่อเนื่องกันไปเป็นแถวเดียวกันตลอด โดยทั่วไปโปรแกรมเมอร์ต้องกำหนดโครงสร้างในส่วนของ Type

```
TYPE  EXAM = ARRAY[1..100] of REAL ;
      GOODS = ARRAY[1..50]  of INTEGER;
VAR   SCORE : EXAM ;
      SALE  : GOODS;
```

การอ้างถึงข้อมูลใช้ตัวชี้ เช่น SCORE[50] เป็นการอ้างถึงตัวแปร SCORE ในอันดับที่ 50 จากการประกาศตัวแปร SCORE นี้โปรแกรมจะทำการจองเนื้อที่เป็นแถวความยาวที่เก็บข้อมูลได้สูงสุด 100 ค่า สำหรับตัวแปร SALE จะจองเนื้อที่เก็บข้อมูลที่เป็นเลขจำนวนเต็มเพื่อเก็บข้อมูลได้ทั้งหมด 50 ค่าโดยเก็บต่อเนื่องกันไป

2. ข้อมูลแถวอันดับ 2 มิติ เป็นการจัดเก็บข้อมูลในลักษณะของแถวอันดับซ้อนแถวอันดับ โครงสร้างมีตัวชี้ข้อมูล 2 ตัวด้วยกัน

```
TYPE  MATRIX = ARRAY[1..4,1..4] of INTEGER;
VAR   A, B     : MATRIX ;
```

การจัดเก็บในหน่วยความจำจะเก็บข้อมูลเป็นแถวยาวทั้งหมด 16 ค่าด้วยกันแต่มีการอ้างถึงโดยใช้ตัวชี้ 2 ตัว ตัวชี้ตัวแรกเปรียบเสมือนแถว ตัวชี้ตัวที่สองเปรียบเสมือน สดมภ์ เช่น A[2,1] คือข้อมูลที่เก็บในอันดับที่ 5 เป็นต้น

1.3.8 ฟังก์ชัน เป็นโปรแกรมย่อยที่โปรแกรมเมอร์สามารถสร้างขึ้นเพื่อคำนวณหรือปฏิบัติการเพื่อให้ได้ผลลัพธ์ โดยทั่วไปเป็นการปฏิบัติการที่ใช้งานบ่อยๆในโปรแกรม ทำให้สะดวกต่อการพัฒนาโปรแกรม การสร้างฟังก์ชั่นเพื่อใช้ในโปรแกรมนั้นต้องตั้งชื่อฟังก์ชันให้สอดคล้องกับการทำงาน กำหนดชนิดของพารามิเตอร์ในการรับค่าและชนิดของผลลัพธ์ นอกจากนี้ต้องมีชุดคำสั่งหรือตัวแปรที่ประกาศขึ้นเพื่อดำเนินการให้ได้ผลลัพธ์ที่ต้องการ การทำงานของฟังก์ชันสิ่งที่เราสนใจคือผลลัพธ์เท่านั้น ภาษาปาสคาลจะส่งผลการทำงานของฟังก์ชันส่งกลับไปยังจุดเรียกใช้ในชื่อของฟังก์ชันเท่านั้น ตัวแปรต่างๆที่กำหนดขึ้นภายในฟังก์ชันจะใช้ได้เฉพาะในฟังก์ชันเท่านั้น และจะคืนเนื้อที่นี้เมื่อการทำงานของฟังก์ชันสิ้นสุดลง สำหรับการส่งผ่านค่าต่างๆจากจุดเรียกใช้ไปยังฟังก์ชันนั้น เราสามารถส่งข้อมูลที่เป็นลักษณะโครงสร้าง เช่น ข้อมูลแถวอันดับ หรือ ระเบียนไปยังฟังก์ชันได้โดยกำหนดตัวแปรให้มีโครงสร้างในลักษณะเดียวกันไว้รองรับ นอกจากนี้ในลักษณะการแก้ปัญหาบางลักษณะที่มีการทำงานในลักษณะที่ซ้ำๆกัน เราสามารถเขียนโปรแกรมในลักษณะเรียกตัวเองก็ได้

ตัวอย่าง

```
Function   Tan(X:real):real;
Begin
        Tan := Sin(X) /Cos(X);
End;
```

การเรียกใช้ฟังก์ชัน ผู้ใช้สามารถเรียกใช้ได้ดังนี้

```
Output := Tan(22/7) ;
```

การเรียกใช้ฟังก์ชันต้องสอดคล้องกับการสร้างฟังก์ชัน กล่าวคือชนิดของตัวแปรที่ส่งกับที่รับในฟังก์ชันต้องเป็นชนิดเดียวกัน และค่าที่ส่งกลับกับตัวแปรที่รับค่าจากจุดเรียกใช้ต้องมีชนิดเดียวกันด้วย

ตัวอย่าง

```
Function   Fac(X:integer); integer;
Begin
        If  X=0  then  Fac := 1
        Else  Fac := X * Fac(X-1) ;
End;
```

การเรียกใช้

```
Output := Fac(3) ;
```

ตัวอย่างนี้เป็นการเรียกฟังก์ชัน Fac โดยส่ง 3 ไปยังฟังก์ชันซึ่งตัวแปร X จะรับค่าที่ส่งมา การทำงานภายในฟังก์ชันจะเรียกตัวเองจนกระทั่งค่า X มีค่าเท่ากับ 0 จึงหยุดการเรียกตัวเอง และส่งค่า 1 กลับมา ต่อจากนั้นจะส่งกลับไปเรื่อยๆ จนกระทั่งจบการทำงาน

1.3.9 โพรซิเจอร์ เป็นโปรแกรมย่อยอีกลักษณะหนึ่งที่ไม่มีค่าในตัวเอง โปรแกรมเมอร์สร้างเพื่อขจัดการปฏิบัติงานที่ซ้ำๆกันในโปรแกรม เพื่อให้โปรแกรมมีความคล่องตัวและง่ายต่อการพัฒนา ซึ่งการสร้างโปรแกรมย่อยในรูปของโพรซิเจอร์มีหลายลักษณะ เช่น อาจไม่มีการส่งผ่านค่าไปให้โพรซิเจอร์ การเรียกใช้เพียงแต่เรียกชื่อโพรซิเจอร์ โปรแกรมจะปฏิบัติงานตามคำสั่งของโพรซิเจอร์ทันที ในบางลักษณะมีการส่งผ่านค่าไปยังโพรซิเจอร์ การทำงานในโพรซิเจอร์จะนำค่าที่ส่งมาปฏิบัติการอาจเป็นการจัดรูปแบบของผลลัพธ์ การนำค่าไปควบคุมการทำงานในโพรซิเจอร์ เป็นต้น และในบางลักษณะเป็นการส่งผ่านแบบอ้างอิงโดยการทำงานของโพรซิเจอร์มีผลกระทบกับค่าของตัวแปรที่ส่ง กล่าวคือถ้าค่าของ

ข้อมูลในโพรซิเจอร์เปลี่ยนมีผลให้ค่าของข้อมูลของตัวเรียกใช้เปลี่ยนไปด้วย เช่น การรับข้อมูลจากแป้นพิมพ์ การสลับค่าข้อมูล เป็นต้น

ตัวอย่าง

```
Procedure   Line(X:integer);
VAR  I  :  1..80;
BEGIN
        FOR I := 1 to X  do
                WRITE('*');
END;
```

การเรียกใช้

```
Line(30);
```

การทำงานของโพรซิเจอร์นี้จะนำค่า 30 ไปให้แก่ตัวแปร X ต่อจากนั้นจะกระทำงานวนรอบโดยพิมพ์ * จำนวนเท่ากับค่าของ X นั่นคือ 30 ครั้ง

ตัวอย่าง

```
Procedure   Swap(VAR  X,Y : INTEGER);
VAR  T : integer;
BEGIN
        T := X;
        X := Y;
        Y := T;
END;
```

การเรียกใช้

```
A := 7; B:= 9;
Swap(A,B);
```

การทำงาน ในที่นี้การส่งผ่านค่าพารามิเตอร์จะเป็นลักษณะผ่านค่าแบบอ้างอิง กล่าวคือไม่มีการส่งผ่านค่าไป แต่จะเป็นลักษณะอ้างอิงโดยตัวแปรที่ทำหน้าที่

IT 257 (H)

รับค่าที่อยู่ภายในโพรซิเจอร์จะใช้เนื้อที่เดียวกับตัวแปรจากจุดเรียกใช้ ตำแหน่งที่ตรงกันตัวต่อตัว ดังนั้นถ้าค่าของตัวแปรที่อ้างอิงในโพรซิเจอร์เปลี่ยนแปลงมีผลให้ตัวแปรจากจุดเรียกใช้เปลี่ยนแปลงไปด้วยเพราะใช้เนื้อที่เดียวกัน สรุปได้ว่าการทำงานของโพรซิเจอร์นี้จะเป็นการสลับค่าระหว่างค่า A และ B

1.3.10 เรคอร์ด เป็นโครงสร้างข้อมูลที่เก็บหรือรวบรวมข้อมูลตามรายละเอียดของงานอาจประกอบด้วยหลายฟิลด์ซึ่งมีชนิดของข้อมูลแตกต่างกันไป

```
TYPE   STUDENTS = RECORD
          NAME  :  STRING[20];
          CODE  :  STRING[10];
          SCORE : INTEGER;
       END;
VAR  STUDENT : STUDENTS;
```

การอ้างถึงข้อมูลโดย ใช้ชื่อตัวแปรตามด้วยจุดทศนิยมและตามด้วยชื่อฟิลด์ เช่น STUDENT.NAME เป็นต้น ข้อมูลชนิดเรคอร์ดสามารถประกาศฟิลด์ให้มีโครงสร้างเป็นชนิดเรคอร์ดได้ด้วย รวมทั้งการจัดเก็บข้อมูลในหน่วยความจำสามารถเก็บเป็นโครงสร้างแถวอันดับ ชนิดเรคอร์ดได้อีกด้วย เช่น

```
TYPE   DATE  =  RECORD
                  DAY     : 1..31 ;
                  MONTH : 1..12 ;
                  YEAY    :  1940..2010;
       END;
       STUDENTS = RECORD
                  NAME  :  STRING[20];
                  CODE  :  STRING[10];
                  BIRTHDAY  : DATE;
                  SCORE : INTEGER;
       END;
```

VAR STUDENT : STUDENTS;

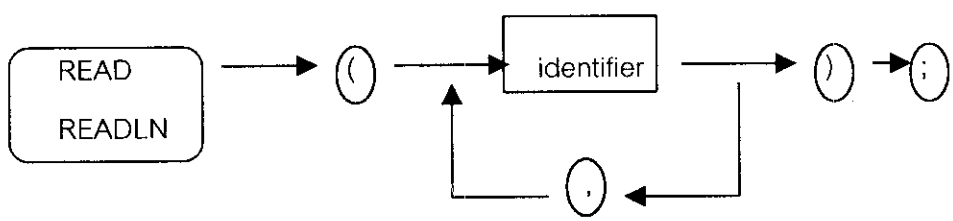
TERM : ARRAY[1..500] of STUDENTS;

การประกาศตัวแปรชนิดนี้เป็นการนำกลุ่มข้อมูลที่มีความสัมพันธ์กันมาเก็บไว้อย่างเป็นระเบียบ เพื่อให้ง่ายต่อการจัดการ

1.4 คำสั่งพื้นฐานในการปฏิบัติการ

1.4.1 READ หรือ READLN เป็นคำสั่งที่ให้เครื่องคอมพิวเตอร์รับข้อมูลจากภายนอกเครื่องเข้าไปเก็บในหน่วยความจำในเนื้อที่ของตัวแปรที่ระบุในวงเล็บหนึ่งค่าต่อหนึ่งชื่อตามลำดับ เช่น read(a,b,c); เป็นการอ่านค่าของข้อมูล 3 ค่าไปเก็บในตัวแปร a, b, c ตามลำดับ

รูปแบบของคำสั่ง READ



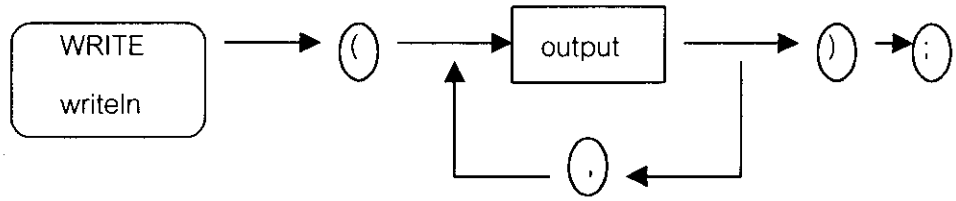
1.4.2 คำสั่ง Assignment (:=) เป็นคำสั่งที่ใช้ในการให้ค่าแก่ตัวแปร ซึ่งอาจเป็นการกำหนดค่าให้โดยตรงหรือเป็นค่าที่เกิดจากการคำนวณก็ได้ โดยทั่วไปการคำนวณค่านิพจน์ของคอมพิวเตอร์มีลำดับการทำงานไม่เท่ากัน กรณีที่เป็นวงเล็บจะกระทำก่อนเป็นลำดับแรก ถ้าวงเล็บซ้อนวงเล็บจะกระทำวงเล็บในสุดก่อน กรณีที่เป็นการคูณหรือการหาร จะกระทำก่อนการบวกและการลบ กรณีที่ลำดับการทำงานเท่ากันจะกระทำจากซ้ายไปขวา เช่น

A := 3 + B * X - 4 ;

การทำงานจะกระทำนิพจน์ B * X เป็นลำดับแรก ต่อจากนั้นทำการ บวก และ ลบ ตามลำดับ ส่งผลของการคำนวณไปเก็บในตัวแปร A เป็นต้น

1.4.3 WRITE หรือ WRITELN เป็นคำสั่งที่ให้แสดงผลซึ่งอาจเป็นค่าของตัวแปร ค่าคงที่ ค่าที่ได้จากการคำนวณนิพจน์ทางคณิตศาสตร์ ค่าที่ได้จากฟังก์ชั่นก็ได้ เช่น Write('a=', a); Write(sqr(x)); Write(6+7);

รูปแบบของคำสั่ง WRITE



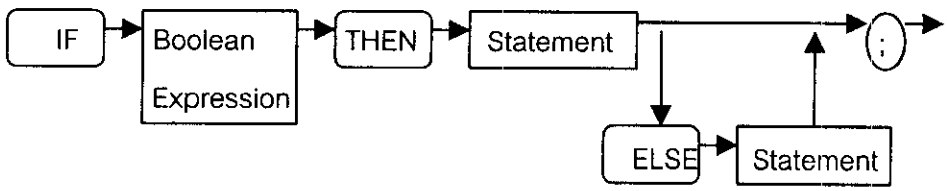
1.4.4 IF..THEN..ELSE เป็นคำสั่งเงื่อนไขที่ใช้สำหรับเปรียบเทียบค่าของข้อมูล ผลลัพธ์ ของการเปรียบเทียบมีค่าเป็น TRUE หรือ FALSE ค่าใดค่าหนึ่งซึ่งส่งผลต่อการทำงาน เช่น IF A>100 then A := A+1 else A := A-1 ; การทำงานจะทำการเปรียบเทียบเงื่อนไข กรณีที่ A>100 จริงจะทำงานหลังคำสั่ง then ในที่นี้คือเพิ่มค่า A ขึ้น 1 กรณีที่เงื่อนไขเป็นเท็จ จะทำงานหลังคำสั่ง else คือการลดค่าของ A ลง 1 ในกรณีที่มีทางเลือกหลายๆทางอาจใช้คำสั่งได้ดังนี้

```
IF  (score >= 0) and (score <= 59)  then  Grade := 'F'
ELSE IF (score >= 60) and (score <= 79)  then  Grade := 'P'
ELSE   Grade := 'G';
```

การทำงานจะเปรียบเทียบเงื่อนไขแรกก่อนว่ามีค่าเป็นจริงหรือไม่ กรณีที่เป็นจริงแสดงว่าคะแนนอยู่ในช่วง 0 ถึง 59 จริง เกรดของนักศึกษามีค่าเท่ากับ F แต่ใน กรณีที่เป็นเท็จ จะทำการเปรียบเทียบเงื่อนไขต่อมา ในกรณีที่เป็นจริง

แสดงว่าคะแนนอยู่ในช่วง 60-79 นักศึกษาจะได้เกรดเป็น P ในกรณีที่เป็นเท็จ แสดงว่านักศึกษาได้คะแนนเกิน 79 คะแนน ผลคือเกรดได้เป็น G

รูปแบบของคำสั่ง IF..THEN..ELSE

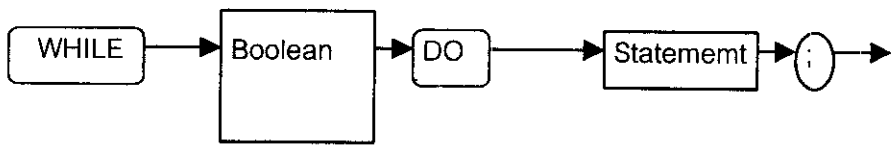


1.4.5 While ..do เป็นคำสั่งที่กระทำซ้ำๆตามเงื่อนไข กล่าวคือถ้าเงื่อนไขเป็นจริงจะกระทำงานซ้ำๆในบล๊อกคำสั่ง จนกระทั่งผลของเงื่อนไขเป็นเท็จ จึงหยุดการทำซ้ำ ดังตัวอย่างต่อไปนี้

```
I := 1
while I<=10  do  begin
        Write('*');
        I := I+1;
End;
```

การทำงานจะทำงานซ้ำคำสั่งที่อยู่ในบล๊อก(begin..end) จำนวน 10 รอบด้วยกัน

รูปแบบของคำสั่ง While..do



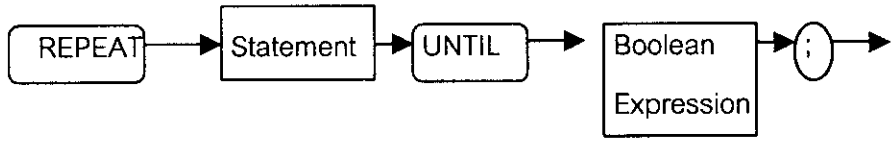
IT 257 (H) 15

1.4.6 Repeat..until เป็นคำสั่งที่กระทำซ้ำๆตามเงื่อนไขเหมือนกัน แต่จะกระทำงานก่อนอย่างน้อย 1 รอบแล้วถึงมีการเปรียบเทียบเงื่อนไขหลังการทำงาน โดยกระทำซ้ำเมื่อเงื่อนไขเป็นเท็จ กรณีที่เป็นจริงจะหยุดการทำซ้ำ เช่น

```
I := 1;
Repeat
        Write('*');
        I := I + 1;
Until  I>10 ;
```

การทำงานจะกระทำซ้ำๆกัน 10 รอบเหมือนกันแต่หยุดการทำงานเมื่อ I มีค่าเท่ากับ 11

รูปแบบของคำสั่ง Repeat..Until



1.4.7 For..Do เป็นคำสั่งทำงานวนรอบซึ่งใช้กับชนิดของข้อมูลที่มีลำดับแน่นอน โดยกำหนดตัวแปรที่มีการเพิ่มขึ้นหรือลดลงจากค่าเริ่มต้นจนกระทั่งถึงค่าสุดท้าย ทำให้การทำงานซ้ำๆจำนวนรอบแน่นอน เช่น

```
FOR I := 1 to 10 do
        WRITE('*');
```

หรือ

```
FOR I := 10 downto 1 do
        WRITE('*');
```

การทำงานนี้จะกำหนดค่าเริ่มต้นให้ตัวแปร I มีค่าเริ่มต้นเป็น 1 และเพิ่มขึ้นทีละ ลำดับจนกระทั่งมีค่าเท่ากับ 10 จะกระทำในรอบสุดท้าย รวมทั้งสิ้น 10 รอบ สำหรับ การใช้ downto จะกลับกันกล่าวคือ ค่า I เริ่มต้นที่ 10 และลดลงเรื่อยๆ จนกระทั่งมีค่าเท่ากับ 1 รวมทำซ้ำ 10 รอบเช่นกัน

รูปแบบของคำสั่ง For..do

1.4.8 CASE..OF เป็นคำสั่งที่ใช้กับตัวแปรที่มีลำดับแน่นอน โดยคำสั่งนี้เหมาะสำหรับการตัดสินใจเลือกกระทำทางหนึ่งจากหลายทาง ซึ่งการที่โปรแกรมมีทางเลือกหลายทางหากใช้คำสั่ง IF..then..else จะทำให้โปรแกรมนั้นยุ่งยาก แก่การเข้าใจ

```
IF  (score >= 0) and (score <= 59)  then  Grade := 'F'
ELSE IF (score >= 60) and (score <= 79)  then  Grade := 'P'
ELSE   Grade := 'G';
```

หรือ

```
IF  (Ch >= 'A') and (Ch <= 'Z')  or (Ch >= 'a') and ( Ch <= 'z') then
        Letter := Letter +1
ELSE  if  (Ch >= '0') and (Ch <= '9')  then  Digit := Digit + 1;
```

สามารถเขียนโดยใช้ CASE..OF ได้ดังนี้

```
CASE   score   OF
        0..59   :  Grade := 'F' ;
        60..79  :  Grade := 'P';
        ELSE
                Grade := 'G' ;
END;
```

และ

```
CASE   Ch   OF
        'A'..'Z'  , 'a'..'z'     :  Letter := Letter+1;
        '0'..'9'                 :  Digit   := Digit + 1;
END;
```

สำหรับทางเลือกของการทำงานในโปรแกรมนั้น คำสั่ง IF..THEN..ELSE จะใช้สำหรับเปรียบเทียบและทำงานได้ดีสำหรับข้อมูลทุกชนิด แต่กรณีที่มีทางเลือกการทำงานหรือเงื่อนไขมากๆ การใช้ CASE..OF จะสะดวกกว่า และในกรณีที่ข้อมูลที่เปรียบเทียบเป็นจำนวนจริง คำสั่ง CASE..OF ไม่สามารถกระทำได้ ดังนั้นการเลือกใช้คำสั่งใดนั้นขึ้นอยู่กับชนิดของข้อมูล ค่าของข้อมูล เงื่อนไขการทำงาน ด้วยการเลือกใช้คำสั่งที่ดีทำให้โปรแกรมมีประสิทธิภาพดีเพิ่มขึ้นด้วย

1.4.9 WITH..DO เป็นคำสั่งที่ใช้กับตัวแปรชนิดเรคอร์ด โดยลดการอ้างถึงตัวแปรเรคอร์ด ทำให้เกิดความคล่องตัวมากยิ่งขึ้น

ตัวอย่าง

```
STUDENT.NAME   := ' URAI  THONGHUAPHAI' ;
STUDENT.CODE   := '2451030582'
STUDENT.SCORE := 85 ;
STUDENT.DATE.DAY      := 29;
STUDENT.DATE.MONTH  := 12;
STUDENT.DATE.YEAR     := 2544;
```

สามารถเขียนได้ใหม่ได้ดังนี้

```
WITH STUDENT DO
BEGIN
        NAME = 'URAI  THONGHUAPHAI' ;
        CODE = '2451030582'
        SCORE = 85 ;
        WITH DATE DO
        BEGIN
                DAY    := 29;
                MONTH := 12;
                YEAR   := 2544;
        END;
  END;
```

สำหรับคำสั่งนี้ใช้เฉพาะข้อมูลชนิดเรคอร์ด เพื่อลดการอ้างถึงตัวแปร ในกรณีที่เป็นเรคอร์ดซ้อนเรคอร์ดจะได้สะดวกทำให้โปรแกรมเมอร์เขียนโปรแกรมได้โดยไม่สับสน

เพื่อให้นักศึกษาได้เข้าใจถึงการเขียนโปรแกรมภาษาปาสคาลได้ดียิ่งขึ้น ขอให้ทำความเข้าใจกับตำราวิชา IT257 และทดลองทำแบบฝึกหัดท้ายบท ถ้าทำไม่ได้ให้พยายามทำความเข้าใจกับเฉลยแบบฝึกหัดในส่วนที่ 2 ของคู่มือเล่มนี้ต่อไป