บทที่ 8

โปรแกรมย่อย (Subprogram)

ในการเขียนโปรแกรมคอมพิวเตอร์บางครั้งจะต้องมีชุดคำสั่งบางชุดที่จะต้องทำบ่อย ๆ ถ้าหากเมื่อโปรแกรมต้องการทำชุดคำสั่งเหล่านั้นและต้องเขียนชุดคำสั่งนั้นใหม่จะทำให้โปรแกรมมีขนาดใหญ่ เราสามารถนำชุดคำสั่งที่ต้องทำบ่อย ๆ มารวมเป็นโปรแกรมย่อยได้ ถ้าหากต้องการทำชุดคำสั่งนั้นเมื่อใดก็ไม่ต้องเขียนใหม่ ให้เรียกชื่อโปรแกรมย่อยนั้นแทน นอกจากนี้การเขียนโปรแกรมในลักษณะโปรแกรมย่อยนั้นจะทำให้เราแบ่งโปรแกรมใหญ่ ๆ เป็นชุดย่อย ๆ ได้ แต่ในโปรแกรมทุกโปรแกรมจะต้องมีโปรแกรมหลักหนึ่งโปรแกรม โดยโปรแกรมย่อยจะถูกทำงานได้เมื่อโปรแกรมหลักเรียกใช้โปรแกรมย่อยนั้น นอกจากนี้การเขียนโปรแกรมในลักษณะโปรแกรมย่อยจะช่วยให้ลดความซับซ้อนของโปรแกรมลงได้ และทำให้ศึกษาการทำงานของโปรแกรมได้ง่าย

ในการเขียนโปรแกรมภาษาปาสคาลจะมีโปรแกรมย่อยอยู่สองประเภทคือ โปรแกรมย่อยที่ไม่มีการคืนค่าให้กับชื่อโปรแกรมเรียกว่าโพรซีเยอร์ (Procedure) และโปรแกรมย่อยที่มีการคืนค่าเรียกว่าฟังก์ชั่น (Function) โดยในบทที่ 3 ได้แสดงมาแล้วว่าฟังก์ชั่นและโพรซีเยอร์จะเป็นส่วนหนึ่งของโปรแกรมเช่นกัน แต่โปรแกรมที่ผ่านมายังไม่ได้กล่าวถึง

8.1 โครงสร้างของโพรซีเยอร์

ในการเขียนโปรแกรมทุกโปรแกรมจะต้องมีโปรแกรมหลัก ตัวอย่างที่ผ่านมาในบทก่อน ๆ โปรแกรมหลักจะอยู่ใน Begin กับ End. แต่ถ้ามีการสร้างโพรซีเยอร์หรือโปรแกรมย่อยขึ้น จะต้องเขียนโพรซีเยอร์หรือโปรแกรมย่อยไว้นอกโปรแกรมหลัก โครงสร้างของโพรซีเยอร์เป็นดังนี้

```
Procedure Precedure_Name;
Begin
        Statement;
End;
```

จะเห็นว่าโครงสร้างของโพรซีเยอร์จะเริ่มด้วยคำว่า Procedure และตามด้วยชื่อโพรซีเยอร์ โดยชื่อนี้จะต้องเป็นไปตามกฎการตั้งชื่อที่ได้ศึกษามา จากนั้นการทำงานต่าง ๆ ภายในโพรซีเยอร์จะเป็นสเตตเมนต์ที่อยู่ภายใน Begin กับ End;

ในการเขียนโปรแกรมที่มีขนาดใหญ่ โครงสร้างแบบโพรซีเยอร์จะช่วยทำให้โปรแกรมดูง่ายขึ้น สามารถแยกโปรแกรมออกเป็นส่วน ๆ ได้ อย่างเช่นถ้าหากต้องการให้คอมพิวเตอร์วาดรูปหน้าคน เราอาจแบ่งเป็นส่วนที่วาดหัว ส่วนที่วาดตา ส่วนที่วาดปาก ส่วนที่วาดจมูก จากนั้นให้โปรแกรมหลักนำแต่ละส่วนมารวมกัน ดังเช่นตัวอย่างที่ 8.1

ตัวอย่างที่ 8.1 โปรแกรมนี้จะให้เลือกเลข 1 หรือ 2 ถ้าหากเลือกเลข 1 คอมพิวเตอร์จะวาดรูปคนยิ้ม ถ้าเป็นเลข 2 จะเป็นรูปคนหน้าดุ การทำงานของโปรแกรมจะสร้างโปรแกรมย่อยสำหรับวาดรูปส่วนต่าง ๆ ของใบหน้า และให้โปรแกรมหลักเรียกส่วนต่าง ๆ มาทำงาน

```
PROGRAM  Faces;
USES  CRT;
VAR   choice  :  integer;

PROCEDURE DRAW_TOP;                          { วาดผม }
BEGIN
        WRITELN('|^^^^^^|');
END;

PROCEDURE DRAW_EYES;                         { วาดรูปตา }
BEGIN
        WRITELN('  @  @  ');
END;

PROCEDURE DRAW_NOSE;                         { วาดรูปจมูก }
BEGIN
        WRITELN('   \/  ');
END;
PROCEDURE DRAW_SMILE;                        { วาดปากหน้ายิ้ม }
BEGIN
        WRITELN('  \__/ ');
END;
```

```
PROCEDURE DRAW_FROWN;                          { วาดปากหน้าดุ }
BEGIN
        WRITELN('   __  ');
        WRITELN('  /  \ ');
END;

BEGIN {MAIN}
        CLRSCR;                                { ลบจอภาพ }
        WRITE('Enter 1 for happy face or 2 for sad face ');
        READLN(choice);
        WRITELN;
        DRAW_TOP;                              { เรียกโปรแกรมย่อยวาดผม }
        DRAW_EYES;                             { เรียกโปรแกรมย่อยวาดตา }
        DRAW_NOSE;                             { เรียกโปรแกรมย่อยวาดจมูก }
        IF  choice  =  1
                then  DRAW_SMILE
        else  DRAW_FROWN
END.
```

ตัวอย่างที่ 8.2 การทำงานของโปรแกรมในโปรแกรมหลักจะเรียกโพซีเยอร์ชื่อ Line_text ซึ่งจะทำการลากเส้นเป็นเครื่องหมาย * จำนวน 16 จุด จากนั้นจะพิมพ์ข้อความและลากเส้นอีกครั้ง ซึ่งจะแสดงว่าการใช้โพซีเยอร์นั้นเราจะเขียนชุดคำสั่งลากเส้นเพียงครั้งเดียวโดยสร้างเป็นโพซีเยอร์และสามารถเรียกใช้ได้หลายครั้ง

```
PROGRAM TEST_PROCEDURE;
USES  CRT;

PROCEDURE LINE_TEXT;
VAR  i  :  integer;
BEGIN
        FOR  i := 1 TO 17 DO
        WRITE('*');
END;
```

```
BEGIN  {MAIN}
        CLRSCR;
        LINE_TEXT;
        WRITELN(' TEERAWAT  KMITL ');
        LINE_TEXT;
END.
```

ตัวอย่างที่ 8.3 เป็นการสร้างโพซีเยอร์ชื่อ Line_text ใหม่โดยใช้คำสั่ง chr(x) ซึ่งจะลากเส้นโดยใช้ตัวอักษรที่มีรหัส ASCII เป็น 176 ในโปรแกรมนี้จะเป็นการเรียกใช้โพซีเยอร์ chr ของเทอร์โบปาสคาล

```
PROGRAM TEST_PROCEDURE;
USES  CRT;
PROCEDURE LINE_TEXT;
VAR  i  :  integer;
BEGIN
        FOR  i := 1 TO 19 DO
        WRITE(chr(176));
        WRITELN;
END;

BEGIN  {MAIN}
        CLRSCR;
        LINE_TEXT;
        WRITELN(chr(176),' TEERAWAT  KMITL ',chr(176));
        LINE_TEXT;
END.
```

ตัวอย่างที่ 8.4 ตัวอย่างนี้จะแสดงเครื่องหมาย * วิ่งไปมา โดยจะสร้างโพซีเยอร์ขึ้นสองโพซีเยอร์ โดยโพซีเยอร์ MoveRight จะทำให้ * วิ่งไปทางขวา และโพซีเยอร์ MoveLeft จะทำให้ * วิ่งไปทางซ้าย โดยการทำให้ * วิ่งนั้นจะใช้ฟังก์ชั่น Gotoxy ในการกำหนดตำแหน่งบนจอภาพ

```
PROGRAM Bounce3;
USES  Crt;
VAR     i   :  Integer;

PROCEDURE MoveRight;                    { วิ่งไปทางขวา }
BEGIN
        FOR  i := 1  TO  79  DO         { วิ่งไปจนถึงคอลัมพ์ที่ 79 }
        BEGIN
                GotoXY(i - 1, 5);
                WriteLn(' ');           { ลบ * ของเก่า }
                GotoXY(i, 5);
                WriteLn('*');           { พิมพ์ * ตัวใหม่ }
                Delay(140);             { หน่วงเวลาเพื่อให้มองเห็นทัน }
        END;
END;

PROCEDURE MoveLeft;                     { วิ่งไปทางซ้าย }
BEGIN
        FOR  i := 79  DOWNTO  1  DO     { วิ่งจากคอลัมพ์ 79 ถึงคอลัมพ์ที่ 1 }
        BEGIN
                GotoXY(i + 1, 5);
                WriteLn(' ');           { ลบ * ของเก่า }
                GotoXY(i, 5);
                WriteLn('*');           { พิมพ์ * ตัวใหม่ }
                Delay(140);             { หน่วงเวลาเพื่อให้มองเห็นทัน }
        END;
END;

BEGIN           { Main }
        CLRSCR;
        MoveRight;              { วิ่งไปทางขวา }
        MoveLeft;               { วิ่งไปทางซ้าย }
END.
```

8.2 ตัวแปรทั่วไปและตัวแปรเฉพาะที่

ตามที่ทราบมาแล้วว่าการเขียนโปรแกรมภาษาปาสคาลนั้นสามารถประกาศตัวแปรใช้ได้สำหรับการเขียนโปรแกรมแบบโปรแกรมย่อยหรือโพซีเยอร์นั้นตัวแปรบางตัวอาจจะมีการเปลี่ยนแปลงไปเมื่ออยู่ในโพซีเยอร์หนึ่ง เมื่อกลับมาโปรแกรมหลักอาจใช้ตัวแปรนั้นไม่ได้ ในเทอร์โบปาสคาลจะมีตัวแปรอยู่สองประเภทคือตัวแปรแบบทั่วไปและตัวแปรแบบเฉพาะที่

ตัวแปรแบบทั่วไป (global variables) เป็นตัวแปรที่ทุกส่วนของโปรแกรมสามารถเรียกใช้ได้ ในการเขียนโปรแกรมใหญ่ ๆ โดยมากแล้วจะใช้ในการประกาศค่าคงที่ในโปรแกรม การประกาศตัวแปรประเภทนี้คล้ายกับตัวแปรในโปรแกรมต่าง ๆ ที่ได้ศึกษามา พิจารณาตัวอย่างโปรแกรมที่ 8.4

ตัวอย่างที่ 8.5 เป็นตัวอย่างการใช้ตัวแปรแบบทั่วไปร่วมกับโพซีเยอร์ ในโปรแกรมจะสร้างโพซีเยอร์ชื่อ EX ภายในโพซีเยอร์จะใส่ค่า 5 ให้กับตัวแปร A

```
PROGRAM TEST;
VAR   A  :  Integer;              { ประกาศตัวแปร A เป็นตัวแปรแบบทั่วไป }

PROCEDURE EX;
BEGIN
        A := 5;                   { ใส่ค่า 5 ให้กับตัวแปร A }
        WRITELN(A);               { พิมพ์ค่าในตัวแปร A }
END;

BEGIN
        A := 3;                   { ใส่ค่า 3 ในตัวแปร A }
        WRITELN(A);               { พิมพ์ค่าในตัวแปร A }
        EX;                       { เรียกโพซีเยอร์ EX }
        WRITELN(A);
END.
```

เมื่อรันโปรแกรมผลลัพธ์ที่ได้จะเป็นดังนี้

```
3
5
5
```

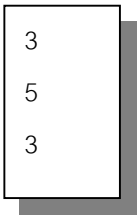
การทำงานของโปรแกรมนั้นจะมีตัวแปร A เป็นตัวแปรแบบทั่วไปซึ่งทุกส่วนของโปรแกรมสามารถเรียกใช้ได้ ในโปรแกรมหลักจะใส่ค่า 3 ให้ตัวแปร A และพิมพ์ค่าในตัวแปร A ซึ่งจะได้เป็น 3 ต่อมาเรียกโพซีเยอร์ EX และในโพซีเยอร์นี้จะใส่ค่า 5 ให้ตัวแปร A และพิมพ์ค่าจะได้เป็น 5 ต่อมาเมื่อจบโพซีเยอร์ EX โปรแกรมหลักจะพิมพ์ค่า A อีกครั้ง ซึ่งจะได้ผลลัพธ์เป็น 5 เนื่องจากค่าเปลี่ยนไปจากการทำโพซีเยอร์ EX

ตัวแปรเฉพาะที่ (local variables) เป็นตัวแปรที่ใช้ภายในโปรแกรมย่อย การประกาศตัวแปรประเภทนี้จะประกาศภายในโปรแกรมย่อยนั้น ๆ ซึ่งจะทำให้โปรแกรมย่อยต่าง ๆ มีชื่อตัวแปรชื่อเดียวกันได้เมื่อโปรแกรมย่อยหนึ่งเรียกใช้จะไม่ทำให้ค่าเปลี่ยนไป เมื่อโปรแกรมย่อยต่าง ๆ ประกาศชื่อตัวแปรชื่อเดียวกันแต่เมื่อโปรแกรมทำงานชื่อตัวแปรของโปรแกรมย่อยต่าง ๆ จะเป็นตำแหน่งหน่วยความจำที่มีตำแหน่งต่างกัน พิจารณาตัวอย่างโปรแกรมที่ 8.5

ตัวอย่างที่ 8.6 เป็นตัวอย่างการใช้ตัวแปรแบบเฉพาะที่ โดยในโพซีเยอร์ EX มีการประกาศตัวแปรแบบเฉพาะที่ชื่อ A ซึ่งเป็นตัวแปรชื่อเดียวกับตัวแปรแบบทั่วไป

```
PROGRAM TEST;
VAR   A  : Integer;              { ประกาศตัวแปร A เป็นตัวแปรแบบทั่วไป }
PROCEDURE EX;
VAR  A :  Integer;               { ประกาศตัวแปร A เป็นตัวแปรแบบเฉพาะที่ }
BEGIN
        A := 5;                  { ใส่ค่า 5 ให้กับตัวแปร A }
        WRITELN(A);              { พิมพ์ค่าในตัวแปร A }
END;
BEGIN
        A := 3;                  { ใส่ค่า 3 ให้กับตัวแปร A }
        WRITELN(A);              { พิมพ์ค่าในตัวแปร A }
        EX;                      { เรียกโพซีเยอร์ EX }
        WRITELN(A);              { พิมพ์ค่าในตัวแปร A }
END.
```

เมื่อรันโปรแกรมผลลัพธ์ที่ได้จะเป็นดังนี้



การทำงานของโปรแกรมนั้นในโปรแกรมหลักจะใส่ค่า 3 ในตัวแปร A โดยโปรแกรมหลักจะเรียกใช้ตัวแปรแบบทั่วไป จากนั้นโปรแกรมจะให้พิมพ์ค่าในตัวแปร A ทำให้ผลลัพธ์ที่ได้เป็น 3 ต่อมามีการเรียกใช้โพซีเยอร์ EX ในโพซีเยอร์นี้จะมีการประกาศตัวแปร A ซึ่งจะเป็นตัวแปรแบบเฉพาะที่ และใส่ค่า 5 ให้กับตัวแปร A เมื่อในโพซีเยอร์ EX ให้พิพม์ค่า A ค่าที่ได้จะเป็น 5 เมื่อจบโพซีเยอร์ในโปรแกรมหลักให้พิมพ์ค่า A อีกครั้งทำให้ค่าที่ได้เป็น 3 ตามเดิม เนื่องจากถ้ามีการประกาศตัวแปรแบบเฉพาะที่ในโพซีเยอร์จะใช้ตัวแปรตัวนั้น ทำให้ค่าในตัวแปร A ที่เป็นตัวแปรแบบทั่วไปมีค่าไม่เปลี่ยนแปลง

8.3 การส่งค่าตัวแปร

ในการเขียนโปรแกรมย่อยหรือโพซีเยอร์นั้นเราสามารถส่งค่าข้อมูลเข้าไปในโพซีเยอร์และให้โพซีเยอร์ส่งค่าข้อมูลกลับมาได้ แต่ในการเขียนโพซีเยอร์จะต้องมีการเขียนพารามิเตอร์ลิสต์ (Parameter list) ตามหลังชื่อโพซีเยอร์เพื่อบอกว่าข้อมูลที่จะส่งให้โพซีเยอร์นั้นจะส่งผ่านตัวแปรชื่ออะไร โดยการเขียนจะเริ่มต้นด้วยวงเล็บเปิดตามด้วยชื่อและชนิดของตัวแปรและมีวงเล็บปิดท้าย ถ้าหากมีการส่งค่าเข้าไปในตัวแปรหลายตัวซึ่งเป็นตัวแปรประเภทเดียวกันจะใช้เครื่องหมาย ‘,’ คั่นระหว่างตัวแปรนั้น แต่ถ้าเป็นตัวแปรต่างประเภทกันจะใช้เครื่องหมาย ‘;’ คั่น ตัวอย่างเช่น

```
PROCEDURE DRAWLINE(x : integer)
```

เป็นการประกาศโพซีเยอร์ชื่อ DRAWLINE และส่งข้อมูลเข้าไปในโพซีเยอร์ผ่านตัวแปร x ซึ่งเป็นตัวแปรแบบจำนวนเต็ม

```
PROCEDURE EX1(x,y : integer ; a : Real)
```

เป็นการประกาศโพซีเยอร์ชื่อ Ex1 และส่งข้อมูลเข้าไปสามตัวคือ x,y และ a โดย x และ y เป็นตัวแปรประเภทจำนวนเต็ม และ a เป็นตัวแปรประเภทจำนวนจริง

ถ้าหากในโพซีเยอร์มีการคือค่าข้อมูลออกมา ตัวแปรที่จะเป็นตัวส่งค่าออกมาจะต้องประกาศเอาไว้เป็นพารามิเตอร์ลิสต์เช่นกัน โดยใช้คำสงวน VAR นำหน้า

ตัวอย่างที่ 8.7 จะสร้างโพซีเยอร์ชื่อ SHOW_TEXT และรับข้อมูลเข้าไปผ่านทางตัวแปร num โดยโพซีเยอร์นี้จะพิมพ์เครื่องหมาย * ตามจำนวนที่กำหนด

```
PROGRAM TEST_PROCEDURE;
USES  CRT;
PROCEDURE SHOW_TEXT(num : integer);
VAR  i  :  integer;
BEGIN
        FOR  i := 1 TO num DO
        WRITE('*');                     { พิมพ์ * เป็นจำนวน num ครั้ง }
        WRITELN;                        { ขึ้นบรรทัดใหม่ }
END;
BEGIN  {MAIN}
        CLRSCR;
        SHOW_TEXT(5);                   { พิมพ์  * จำนวน 5 ครั้ง }
        SHOW_TEXT(15);                  { พิมพ์ * จำนวน 15 ครั้ง }
END.
```

เมื่อรันโปรแกรมและเรียกโพซีเยอร์ LINE_TEXT ครั้งแรกจะพิมพ์เครื่องหมาย * เป็นเส้นตรงจำนวน 5 จุด เนื่องจากมีการส่งค่า 5 เข้าไปในโพซีเยอร์ โดยค่า 5 นี้จะถูกเก็บในตัวแปร num ซึ่งเป็นพารามิเตอร์ของโพซีเยอร์ LINE_TEXT และเมื่อเรียกโพซีเยอร์ LINE_TEXT อีกครั้งจะพิมพ์ * จำนวน 15 จุด

ตัวอย่างที่ 8.8 ตัวอย่างโปรแกรมนี้จะพิมพ์คำว่า COMPUTER บนจอภาพในตำแหน่งที่กำหนด โดยสร้าโพซีเยอร์ชื่อ SHOW_TEXT และผ่านค่าเข้าไปสองค่าคือ x และ y โดยเป็นค่ากำหนดตำแหน่งที่จะแสดงผลบนจอภาพ ในโพซีเยอร์ SHOW_TEXT จะใช้ gotoxy() เพื่อกำหนดเคอร์เซอร์ที่ตำแหน่งต่าง ๆ บนจอภาพ

```
PROGRAM TEST_PROCEDURE;
USES  CRT;
PROCEDURE SHOW_TEXT(x,y : integer);
VAR  i  :  integer;                     { ตัวแปรใช้ภายในโพซีเยอร์ }
BEGIN
        FOR  i := 1 TO 5 DO             { ทำซ้ำ 5 ครั้งเพื่อพิมพ์ข้อความ }
        BEGIN
                GOTOXY(x+i,y+i);        { กำหนดตำแหน่งที่จะพิมพ์ }
```

```
                WRITE('COMPUTER');
        END;
END;

BEGIN  {MAIN}
        CLRSCR;
        SHOW_TEXT(20,5);
END.
```

เมื่อรันโปรแกรมจะพบว่าคอมพิวเตอร์จะพิมพ์คำว่า COMPUTER เยื้องกัน 5 ครั้ง สำหรับตำแหน่งที่จะให้แสดงข้อความจะส่งเป็นพารามิเตอร์เข้าไปในโพซีเยอร์ โดย SHOW_TEXT(20,5) จะเป็นการใส่ค่า 20 ให้กับ x และ 5 ให้กับ y

ตัวอย่างที่ 8.9 จะนำโปรแกรมตัวอย่างที่ 8.8 มาแก้ไขโดยเขียนโพซีเยอร์ที่มีการส่งพารามิเตอร์เข้าไปสามตัว โดย x และ y เป็นจำนวนเต็ม และ st เป็นสตริง

```
PROGRAM TEST_PROCEDURE;
USES  CRT;
PROCEDURE SHOW_TEXT(x,y : integer ; st : string);
VAR  i  :  integer;                    { ตัวแปรใช้ภายในโพซีเยอร์ }
BEGIN
        FOR  i := 1 TO 5 DO
        BEGIN
                GOTOXY(x+i,y+i);
                WRITE(st);                          { พิมพ์ค่าสตริงที่รับเข้ามา }
        END;
END;
BEGIN  {MAIN}
        CLRSCR;
        SHOW_TEXT(20,5,'TEERAWAT');          { ส่งตัวเลขและสตริงเข้าไปในโพซีเยอร์ }
END.
```

ตัวอย่างที่ 8.10 ตัวอย่างนี้จะสร้างโพซีเยอร์ในการนำอักขระมาประกอบเป็นรูปสามเหลี่ยม โดยโปรแกรมหลักจะเรียกใช้โพซีเยอร์สองครั้งและผ่านค่าแบบต่าง ๆ เข้าไป

```
PROGRAM Triangle;
USES Crt;

PROCEDURE LineDraw(Length: Integer);
VAR      i  : Integer;
BEGIN
          FOR   i := 1   TO   Length * 1   DO
          Write('>');
          WriteLn;
END;

VAR    Count : Integer;                { ส่วนประกาศตัวแปรของโปรแกรม }
BEGIN  { main }
          ClrScr;
          FOR Count := 1 TO 10 DO
                    LineDraw(Count);
          FOR Count := 10 DOWNTO 1 DO
                    LineDraw(Count);
          WriteLn;
          WriteLn('Press ENTER to end this program');
          ReadLn;
END.
```

ตัวอย่างที่ 8.11 โปรแกรมนี้จะเป็นการพิมพ์ข้อความให้ล่วงลงมาจากด้านบนของจอภาพ โดยจะใช้วิธีการเขียนข้อความที่ตำแหน่งถัดไปและลบข้อความเก่าทิ้ง โดยตำแหน่งข้อความบนหน้าจอจะใช้ gotoxy เป็นตัวกำหนด

```
PROGRAM TEST_PROCEDURE;
USES  CRT;
VAR   data  :  string[10];
```

```
PROCEDURE SHOW_TEXT(x : integer ; st : string);
VAR  i  :  integer;                          { ประกาศตัวแปร i เพื่อใช้ภายในโพซีเยอร์ }
BEGIN
        FOR  i := 1 TO 24 DO
        BEGIN
                GOTOXY(x,i);
                WRITE(st);                   { พิมพ์ข้อความที่รับเข้ามา }
                DELAY(500);                  { หน่วงเวลาไป 0.5 วินาที }
                GOTOXY(x,i);
                WRITE('              ');     { ลบข้อความที่ตำแหน่งเดิมทิ้ง }
        END;
END;                                         { จบโพซีเยอร์ }

BEGIN  {MAIN}
        CLRSCR;                              { ลบจอภาพ }
        WRITE('Input String  : ');
        READLN(data);                        { รับข้อมูลสตริงมาเก็บใน Data }
        CLRSCR;
        SHOW_TEXT(20,data);                  { เรียกโปรแกรมย่อย SHOW_TEXT }
END.
```

เมื่อรันโปรแกรมเครื่องจะให้ใส่ข้อมูลสตริงเข้าไปจากนั้นจะเรียกโพรซีเยอร์ SHOW_TEXT ให้พิมพ์ข้อมูลที่รับเข้าไปที่ตำแหน่งคอลัมพ์ x เท่ากับ 20 และให้ข้อความนั้นล่วงลงมา จากโปรแกรมจะเห็นว่ามีการประกาศตัวแปรชื่อ data เป็นตัวแปรแบบทั่วไปให้รับข้อความได้ไม่เกิน 10 ตัวอักษร โดยในโปรแกรมหลักจะเรียกใช้ตัวแปรนี้ และในโพซีเยอร์จะรับค่าสตริงผ่านทางตัวแปร st เมื่อเรียกใช้โพซีเยอร์จะเป็นการส่งค่าจากตัวแปร data ให้กับตัวแปร st เพื่อนำไปใช้ในโพซีเยอร์ ตัวแปร st ในโพซีเยอร์บางครั้งจะถูกเรียกว่าตัวแปรหุ่น (dummy variable) การส่งค่าให้กับตัวแปรในโพซีเยอร์แบบนี้เรียกว่าการส่งค่าโดยการอ้างอิง

โพซีเยอร์ต่าง ๆ ที่ได้ศึกษามาเป็นโพซีเยอร์ที่เราสร้างขึ้นมาเอง ในเทอร์โบปาสคาลมีโพซีเยอร์สำเร็จรูปให้เราใช้มากมาย ซึ่งสามารถศึกษาได้จากคู่มือของเทอร์โบปาสคาลโดยตรง โดยการใช้โพซีเยอร์ต่าง ๆ ที่มีให้เราต้องทราบว่าจะต้องส่งข้อมูลกี่ตัวให้กับโพซีเยอร์ และข้อมูลที่ส่งให้เป็นข้อมูลประเภทใด ที่ผ่านมาเราได้ทดลองใช้โพซีเยอร์ของเทอร์โบปาสคาลมาบ้างแล้วเช่น gotoxt , clrscr , delay เป็นต้น โพซีเยอร์สำเร็จรูปเหล่านี้เรียกว่า standard procedure

แบบฝึกหัด

1. จงบอกเอาต์พุตจากการทำโปรแกรมต่อไปนี้

```
PROCEDURE  TEST;
PROCEDURE  PRINT_AVG(name : string;s1,s2 : integer);
VAR      avg  :  real;
BEGIN
          avg := (s1 + s2)/2;
          WRITELN(name,avg:5:1)
END;

BEGIN
          PRINT_AVG('Smith ',80,90);
          PRINT_AVG('Jones ',70,73)
END.
```

2. จงบอกเอาต์พุตจากการทำโปรแกรมต่อไปนี้

```
PROGRAM TEST;
VAR years  :  integer;

PROCEDURE HOW_LONG(yrs : integer);
BEGIN
          WRITELN(yrs, '  years');
END;

BEGIN
          yrs  :=  20;
          HOW_LONG (yrs);
END.
```

3. จงบอกเอาต์พุตจากการทำโปรแกรมต่อไปนี้

ก.

```
PROGRAM TEST;
VAR  x  :  integer;
PROCEDURE JOB (w : integer);
BEGIN
        w  :=  w + 1;
        WRITELN (w)
END;
BEGIN
        x  :=  0;
        JOB (x);
        WRITELN (x);
END.
```

ข.

```
PROGRAM TEST;
VAR  x  :  integer;
PROCEDURE JOB(x : integer);
BEGIN
        x  :=  x + 1;
        WRITELN (x)
END;
BEGIN
        x  :=  0;
        JOB (x);
        WRITELN (x)
END.
```

4. จงบอกเอาต์พุตจากการทำโปรแกรมต่อไปนี้

```
PROGRAM TEST;
VAR   x , y  :  integer;
PROCEDURE PROB (a,b  :  integer);
BEGIN
        if  a = b
                then  WRITELN('SUM  ', a + b)
                else  WRITELN('Prod  ', a * b);
END;
BEGIN
        x  :=  3;
        y  :=  5;
        while (x >= 0) and (y >= 0) do
        BEGIN
                PROB (x,y);
                y  :=  y – 2;
                x  :=  x – 1
        END
END.
```

5. จงบอกเอาต์พุตเมื่อคอมพิวเตอร์ทำคำสั่งต่อไปนี้

```
PROGRAM TEST;
VAR  a,b,c  :  integer;


PROCEDURE  MIX (var  x,y : integer; z : integer);
VAR  t  :  integer;
BEGIN
          t  :=  z + 1;
          z  :=  y;
          y  :=  x;
          x  :=  t;
          WRITELN (x,'    ',y,'  ',z,'    ',t)
END;


BEGIN
          a  :=  2;
          b  :=  4;
          c  :=  6;
          MIX (a,b,c);
          MIX (b,c,a);
          WRITELN (a,'    ',b,'     ',c)
END.
```