บทที่ Ã

หลักการพื้นฐานในการเขียนโปรแกรม

จุดประสงค์เชิงพฤติกรรม

1. เข้าใจพื้นฐานการเขียนโปรแกรม
2. เข้าใจการเขียนโปรแกรมเมนู เพื่อเรียกใช้งานโปรแกรมอื่น ๆ ที่พัฒนาขึ้น
3. เข้าใจการเชื่อมแฟ้มข้อมูล
4. เข้าใจการเขียนโปรแกรมเพื่อปรับปรุงข้อมูล เช่น เพิ่ม ลบ และแก้ไข

หัวข้อในบทเรียน

4.1 ศึกษาการทำซ้ำด้วยพีระมิด
4.2 สูตรคูณ และการคำนวณ
4.3 การอ่านข้อมูลมาพิมพ์
4.4 การแต่งจอภาพอย่างง่าย
4.5 การเขียนเมนู
4.6 การเชื่อมแฟ้ม 2 แฟ้ม
4.7 การสร้างเสียง
4.8 โปรแกรมข้อสอบ
4.9 การปรับปรุงข้อมูล
ตัวอย่างโปรแกรมเกี่ยวกับโปรแกรมที่น่าสนใจ 69 ตัวอย่าง

หมายเหตุ

เนื้อหาในบทนี้สร้างขึ้น เพื่อใช้เป็นแนวในการสอนพื้นฐานการเขียนโปรแกรมอย่างง่าย ตัวอย่างส่วนใหญ่ผู้เรียนสามารถนำความรู้เดิมมาเป็นพื้นฐานได้ทันที เช่น DBASE, C, BASIC และ PASCAL เป็นต้น ส่วนข้อสอบ Online และเมนู เป็นการประยุกต์มาจากความต้องการของผู้เขียน สำหรับคำสั่ง หรือฟังก์ชันที่ไม่มีการอธิบายไว้โดยละเอียด ผู้เรียนสามารถเปิดดูรายละเอียดจากบทที่ 2 และบทที่ 3 เพื่อทำความเข้าใจได้ทันที

บทที่ 4 หลักการพื้นฐานในการเขียนโปรแกรม

ในบทนี้จะรวมพื้นฐานการเขียนโปรแกรมในลักษณะต่าง ๆ เพื่อเป็นพื้นฐานที่จะเข้าใจการเขียนโปรแกรมในหลายแบบ เช่น การทำความเข้าใจการทำซ้ำโดยศึกษาจากพีระมิด การจัดการกับข้อมูลโดยเชื่อมข้อมูลของแฟ้ม การปรับปรุงแฟ้มข้อมูล หรือการอ่านข้อมูลมาทำรายงานเป็นต้น

& 4.1 ศึกษาการทำซ้ำด้วยพีระมิด

4.1.1 การพิมพ์ 1 ถึง 10

ในหัวข้อนี้จะแสดงให้เห็น 2 คำสั่ง เพื่อใช้ทำซ้ำ คือ FOR และ WHILE โดยยกตัวอย่างเกี่ยวกับการวนลูปเพื่อพิมพ์เลข 1 ถึง 10 เพราะเป็นโปรแกรมในรูปแบบนี้เข้าใจได้ง่าย ๆ

: ตัวอย่างที่ 4.1

```
//การพิมพ์ 1 ถึง 10 บรรทัดละ 1 จำนวน ด้วย FOR แบบธรรมดา
FOR I = 1 TO 10
   ? I
NEXT
// หรือเขียนอีกแบบหนึ่งได้ดังนี้  FOR I = 1 TO 10 ; ? I ; NEXT
```

: ตัวอย่างที่ 4.2

```
//การพิมพ์ 1 ถึง 10 บรรทัดละ 3 จำนวน ด้วย FOR แบบธรรมดา
//โดยแสดงให้เห็นการใช้คำสั่ง ? และ ??
//ซึ่ง ? หมายถึง การปัดบรรทัดแล้วพิมพ์
FOR I = 1 TO 10
   ?? I,I
   ?? I
   ? '=====' ;   ?
NEXT
```

: ตัวอย่างที่ 4.3

```
//การพิมพ์ 1 ถึง 10 โดย WHILE โดยกำหนดค่าเริ่มต้นเป็น 0
//แต่จะแสดง 1 ถึง 10 ทั้งหมด 2 รอบ โดยแยกคำสั่ง WHILE และ DO WHILE
I = 0
WHILE I < 10
  I++
  ? I
END
```

```
? '--------------------'
I = 0
DO WHILE I < 10
  I = I + 1
  ? I
ENDDO
```

: ตัวอย่างที่ 4.4

```
//การพิมพ์ 1 ถึง 10 โดย WHILE โดยกำหนดค่าเริ่มต้นเป็น 1
//ตัวอย่างนี้จะพิมพ์ 1 ถึง 10 ทั้งหมด  3 รอบ
I = 1
WHILE I <= 10
  ? I
  I++
END
? '-------------'
I = 1
DO WHILE I <= 10
  ? I
  I = I + 1
ENDDO
? '-------------'
I = 1
DO WHILE I < 11
  ? I
  I++
ENDDO
```

ต.ย.ผลลัพธ์

```
 1
 2
 3
 4
 5
 6
 7
 8
 9
10
---
 1
 2
 3
 4
 5
 6
 7
 8
 9
10
---
 1
 2
 3
 4
 5
 6
 7
 8
 9
10
```

4.1.2 พิมพ์ชุดของตัวเลขเป็นรูปสามเหลี่ยมชิดซ้าย

ในหัวข้อนี้ ต้องการให้เรียนรู้การใช้ลูปของ คำสั่ง FOR 2 ชุด โดยคำสั่ง FOR ชุดแรกใช้คุมจำนวนแถว และคำสั่ง FOR ชุดใน ใช้คุมหลักของแต่ละแถว

: ตัวอย่างที่ 4.5

// I เป็นตัวเลข ใช้เนื้อที่ในการพิมพ์มาก จึงทำเป็นตัวอักษร แล้วตัดช่องว่างทิ้ง

```
FOR I = 1 TO 3                          // คุมจำนวนบรรทัด
  FOR J = 1 TO 5                        // คุมจำนวนตัวเลขที่ใช้พิมพ์ในแต่ละบรรทัด
    ?? LTRIM(STR(I))
  NEXT                                  // ใช้คุมจำนวนหลักในแต่ละบรรทัด
  ?
NEXT                                    // ใช้คุมจำนวนบรรทัด
```

ต.ย.ผลลัพธ์

```
11111
22222
33333
```

: ตัวอย่างที่ 4.6

```
FOR I = 1 TO 3                          // คุมจำนวนบรรทัด
  FOR J = 1 TO I                        // คุมจำนวนตัวเลขที่ใช้พิมพ์ในแต่ละบรรทัด
    ?? LTRIM(STR(I))
  NEXT
  ?
NEXT
```

ต.ย.ผลลัพธ์

```
1
22
333
```

: ตัวอย่างที่ 4.7

```
FOR I = 1 TO 3                          // คุมจำนวนบรรทัด
  FOR J = 1 TO I                        // คุมจำนวนตัวเลขที่ใช้พิมพ์ในแต่ละบรรทัด
    ?? LTRIM(STR(J))
  NEXT
  ?
NEXT
```

ต.ย.ผลลัพธ์

```
1
12
123
```

: ตัวอย่างที่ 4.8

```
FOR I = 1 TO 3                          // คุมจำนวนบรรทัด
  FOR J = I TO 1   STEP -1              // คุมจำนวนตัวเลขที่ใช้พิมพ์ในแต่ละบรรทัด
    ?? LTRIM(STR(J))
  NEXT
  ?
NEXT
```

ต.ย.ผลลัพธ์

```
1
21
321
```

: ตัวอย่างที่ 4.9

```
FOR I = 1 TO 3
  FOR J = 1 TO (2*I-1)
    ?? LTRIM(STR(J))
  NEXT
  ?
NEXT
```

ต.ย.ผลลัพธ์

```
1
123
12345
```

: ตัวอย่างที่ 4.10

```
// ถ้าใช้ , กั้นระหว่างตัวแปรในคำสั่ง ?? จะทำให้เกิดช่องว่าง 1 ช่อง
FOR I = 1 TO 3
  ?? '*'
  FOR J = 1 TO (2*I-1)
     ?? LTRIM(STR(J))
  NEXT
  ?? '*',LTRIM(STR(I))
  ?
NEXT
```

ต.ย.ผลลัพธ์

```
*1* 1
*123* 2
*12345* 3
```

4.1.3 พิมพ์ชุดของตัวเลขเป็นรูปสามเหลี่ยมชิดขวา

ในหัวข้อนี้ ต้องการให้เรียนรู้การใช้ลูปของ FOR 3 ชุด โดยคำสั่ง FOR ชุดแรกใช้คุมจำนวนบรรทัด และจะคุมคำสั่ง FOR อีก 2 ชุด ในขณะที่คำสั่ง FOR ชุดในชุดแรกใช้คุมหลักของแต่ละบรรทัด เพื่อพิมพ์ช่องว่างและคำสั่ง FOR ชุดในชุดที่สองใช้คุมหลักของแต่ละบรรทัด เพื่อพิมพ์ตัวเลข

: ตัวอย่างที่ 4.11

```
FOR I = 1 TO 3                    // คุมจำนวนบรรทัด
  FOR J = 1 TO 5-I                // พิมพ์ช่องว่างหน้าตัวเลขของแต่ละบรรทัด
    ?? ' '
  NEXT
  FOR J = 1 TO I                  // พิมพ์ตัวเลขของแต่ละบรรทัด
    ?? LTRIM(STR(J))
  NEXT ;    ?
NEXT
```

ต.ย.ผลลัพธ์

```
  1
 12
123
```

: ตัวอย่างที่ 4.12

```
FOR I = 1 TO 3                    // คุมจำนวนบรรทัด
  FOR J = 1 TO 5-I                // พิมพ์ช่องว่างหน้าตัวเลขของแต่ละบรรทัด
    ?? ' '
  NEXT
  FOR J = 1 TO (2*I-1)            // พิมพ์ตัวเลขของแต่ละบรรทัด
    ?? '*'
  NEXT
  ?
NEXT
```

ต.ย.ผลลัพธ์

```
  *
 ***
*****
```

: ตัวอย่างที่ 4.13

```
FOR I = 1 TO 3                    // คุมจำนวนบรรทัด
  FOR J = 1 TO 5-I                // พิมพ์ช่องว่างหน้าตัวเลขของแต่ละบรรทัด
    ?? ' '
  NEXT
  ?? '*'
  FOR J = 1 TO I                  // พิมพ์ตัวเลขของแต่ละบรรทัด
    ?? LTRIM(STR(J))
  NEXT
  ?? '*'
  ?? LTRIM(STR(I))
  ?? '*'
  ?? LTRIM(STR(I*2))
  ?
NEXT
```

ต.ย.ผลลัพธ์

```
   *1*1*2
  *12*2*4
 *123*3*6
```

4.1.4 พิมพ์ชุดของตัวเลขเป็นรูปสามเหลี่ยมต่าง ๆ

ในหัวข้อนี้ ต้องการให้เรียนรู้การใช้ลูปของ FOR มากกว่า 3 ชุดขึ้นไป

: ตัวอย่างที่ 4.14

```
FOR I = 1 TO 3                    // คุมจำนวนบรรทัด
  FOR J = 1 TO 5-I                // พิมพ์ช่องว่างหน้าตัวเลขของแต่ละบรรทัด
    ?? ' '
  NEXT
  FOR J = 1 TO I                  // พิมพ์ตัวเลขของแต่ละบรรทัดชุดแรก 123
    ?? LTRIM(STR(J))
  NEXT
  FOR J = 1 TO 5                  // พิมพ์ตัวเลขของแต่ละบรรทัดชุดที่สอง 33333
    ?? LTRIM(STR(I))
  NEXT
  ?
NEXT
```

ต.ย.ผลลัพธ์

```
   111111
  1222222
 12333333
```

: ตัวอย่างที่ 4.15

```
FOR I = 1 TO 4
  FOR J = 1 TO 5-I
    ?? ' '
  NEXT
  FOR J = 1 TO I
    ?? LTRIM(STR(J))
  NEXT
  FOR J = I-1 TO 1 STEP -1
    ?? LTRIM(STR(J))
  NEXT
  ?
NEXT
```

ต.ย.ผลลัพธ์

```
    1
   121
  12321
```

: ตัวอย่างที่ 4.16

```
FOR I = 1 TO 3
  FOR J = 1 TO 4-I
    ?? LTRIM(STR(J))
  NEXT
  FOR J = 1 TO I-1
    ?? ' '
  NEXT
  ?? '*'
  FOR J = 1 TO 3-I
    ?? ' '
  NEXT
  FOR J = 1 TO I
    ?? LTRIM(STR(J))
  NEXT
  ?
NEXT
```

ต.ย.ผลลัพธ์

```
123*  1
12 * 12
1  *123
```

: ตัวอย่างที่ 4.17

```
FOR I = 1 TO 4
  FOR J = 1 TO I
    FOR K = 1 TO J
      ?? LTRIM(STR(K))
    NEXT
  NEXT
  ?
NEXT
```

ต.ย.ผลลัพธ์

```
1
112
112123
```

: ตัวอย่างที่ 4.18

```
CLS
LPOS = 10
CPOS = 40
ACCEPT TO X
X = VAL(X)
FOR I = 1 TO X
  Y = MOD(I,4)
  DO CASE
    CASE Y = 1                    // สำหรับแนวตั้งชิดซ้าย
      FOR J = 1 TO I
        SETPOS(LPOS+J,CPOS) ; ?? '*'
      NEXT
      SETPOS(LPOS+J,CPOS-1)
    CASE Y = 2                    // สำหรับแนวนอนเส้นล่าง
      FOR J = 1 TO I
        SETPOS(LPOS,CPOS+J) ; ?? '*'
      NEXT
      SETPOS(LPOS+1,CPOS+J)
    CASE Y = 3                    // สำหรับแนวตั้งชิดขวา
      FOR J = 1 TO I
        SETPOS(LPOS-J,CPOS) ; ?? '*'
      NEXT
      SETPOS(LPOS-J,CPOS+1)
```

ต.ย.ผลลัพธ์

IF X=3

```
   *
 * *
***
```

ต.ย.ผลลัพธ์

IF X=5

```
 *****
*     *
* * * *
*  ***
*
```

```
    CASE Y = 0                          // สำหรับแนวนอนเส้นบน
       FOR J = 1 TO I
          SETPOS(LPOS,CPOS-J) ; ?? '*'
       NEXT
       SETPOS(LPOS-1,CPOS-J)
  ENDCASE
  LPOS = ROW()
  CPOS = COL()
NEXT
```

& 4.2 สูตรคูณ และการคำนวณ

โปรแกรมเกี่ยวกับสูตรคูณ เป็นโปรแกรมที่น่าศึกษา เพราะทำความเข้าใจได้โดยง่าย และทบทวนการทำซ้ำโดยคำสั่งวนลูปได้เป็นอย่างดี และศึกษาการคำนวณด้วยเครื่องหมายต่าง ๆ

4.2.1 สูตรคูณในหลายลักษณะ

ในหัวข้อนี้ ต้องการให้เข้าใจการคำนวณอย่างง่าย ๆ โดยศึกษาจากสูตรคูณ

: ตัวอย่างที่ 4.19

```
// พิมพ์สูตรคูณแม่ 2 ธรรมดา
CLS
X = 2
FOR I = 1 TO 12
  ? X," * ",I," = ",X*I
NEXT
@ 0,1 TO 13,40                          // ตีเส้นล้อมรอบสูตรคูณ
```

: ตัวอย่างที่ 4.20

```
// พิมพ์สูตรคูณแม่ 2 และ 3 ให้ติดกัน
// หมายความว่าสูตรคูณแม่ 2 อยู่ซีกซ้าย และสูตรคูณแม่ 3 อยู่ซีกขวา
// โดย STR(2,3) มีหน้าที่ทำให้เลข 2 เป็นตัวอักษรขนาด 3 ช่อง ถูกใช้เพื่อความสวยงาม
CLS
FOR I = 1 TO 12
  ?  "2 * ",STR(I,2)," = ",STR(2*I,3)
  ?? " | "
  ?? "3 * ",STR(I,2)," = ",STR(3*I,3)
NEXT
```

: ตัวอย่างที่ 4.21

```
// พิมพ์สูตรคูณให้สวยงาม โดยเลือกสูตรคูณจากแป้นพิมพ์ได้
CLS
SET DELIMITERS TO "[ ]"
SET DELIMITERS ON
X = 2
@ 2,5 SAY "แม่สูตรคูณที่ต้องการคือ " GET X
READ
SET COLOR TO "B/W"
@ 5,10 CLEAR TO 16,40
FOR I = 1 TO 6
   SETPOS(4+I,15)
   ?? STR(X,2)," * ",STR(I,2)," = ",STR(X*I,3)
NEXT
FOR I = 7 TO 12                          // จะให้ผลต่างกันเพราะ , ใน ?? จะเพิ่มช่องว่างอีก 1 ช่อง
   @ 4+I,15 SAY STR(X,2)+" * "+STR(I,2)+" = "+STR(X*I,3)
NEXT
```

4.2.2 เครื่องคิดเลข

ในหัวข้อนี้ ต้องการให้เห็นการรับค่าจากแป้นพิมพ์มาคำนวณ เช่นเครื่องคิดเลข

: ตัวอย่างที่ 4.22

```
// รับค่าจากแป้นพิมพ์มา 2 ค่า แล้วหาผลบวก ลบ คูณ หาร และยกกำลัง
CLS
X = 0
Y = 0
@ 5,10 SAY "GET  FIRST NUMBER " GET X
@ 6,10 SAY "GET SECOND NUMBER " GET Y
READ
@ 7,10 SAY "RESULT OF + " + STR(X+Y)
@ 8,10 SAY "RESULT OF -" + STR(X-Y)
@ 9,10 SAY "RESULT OF *" + STR(X*Y)
@ 10,10 SAY "RESULT OF /" + STR(X/Y)
@ 11,10 SAY "RESULT OF ^" + STR(X**Y)
```

: ตัวอย่างที่ 4.23

```
// รับค่า 2 ค่า และรับตัวดำเนินการเป็นตัวที่ 3
// โดยใช้คำสั่ง CASE เลือกคำนวณ
DO WHILE .T.
  ? "PRESS 0 ON GET SIGN TO EXIT"
  ? "-------------------------"
  ACCEPT "   GET X : " TO X
  ACCEPT "   GET Y : " TO Y
  ACCEPT "GET SIGN : " TO S
  DO CASE
    CASE  S = "+"   ;  ? "YOUR RESULT IS ",VAL(X) + VAL(Y)
    CASE  S = "-"   ;  ? "YOUR RESULT IS ",VAL(X) - VAL(Y)
    CASE  S = "*"   ;  ? "YOUR RESULT IS ",VAL(X) * VAL(Y)
    CASE  S = "/"   ;  ? "YOUR RESULT IS ",VAL(X) / VAL(Y)
    CASE  S = "^"   ;  ? "YOUR RESULT IS ",VAL(X) ^ VAL(Y)
    CASE  S = "0"   ;  EXIT
    OTHERWISE       ;  ? "YOUR SIGN IS ERROR."
  ENDCASE
ENDDO
```

: ตัวอย่างที่ 4.24

```
// รับค่า 2 ค่า และรับตัวดำเนินการเป็นตัวที่ 3
// โดยใช้คำสั่ง CASE เลือกคำนวณ และสามารถกดปุ่ม ESC เพื่อออกจากโปรแกรม
CLS
DO WHILE LASTKEY() != 27
  X := Y := R := 0
  Z = " "
  @ 5,10 SAY "PRESS ESC TO EXIT"
  @ 6,10 SAY "----------------"
  @ 7,10 SAY "GET    X : " GET X
  @ 8,10 SAY "GET    Y : " GET Y
  @ 9,10 SAY "GET SIGN : " GET Z VALID Z $ '+-*/^'
  READ
```

```
  DO CASE
    CASE  Z = "+"  ;  R = X + Y
    CASE  Z = "-"  ;  R = X - Y
    CASE  Z = "*"  ;  R = X * Y
    CASE  Z = "/"  ;  R = X / Y
    CASE  Z = "^"  ;  R = X ^ Y
  ENDCASE
  IF Z $ '+-*/^'
    @ 10,10 SAY "RESULT IS : " + STR(R)
    INKEY(3)
    @ 10,10 CLEAR
  ENDIF
ENDDO
```

& 4.3 การอ่านข้อมูลมาพิมพ์

การอ่านข้อมูลจากแฟ้มเดียวมาพิมพ์

ให้แฟ้มนี้ชื่อแฟ้มพนักงาน EMPL.DBF		
1. รหัสพนักงาน	EMPL	9(7)
2. ชื่อพนักงาน	NAME	X(20)
3. สกุลพนักงาน	SURN	X(20)

4.3.1 การอ่านข้อมูลมาพิมพ์ทางจอภาพ

หัวข้อนี้ จะนำข้อมูลจากแฟ้มมาพิมพ์ โดยแสดงให้เห็นการวนลูปเพื่อนำข้อมูลมาแสดงผล

: ตัวอย่างที่ 4.25

```
// อ่านข้อมูลมาพิมพ์อย่างง่าย ๆ
USE EMPL
WHILE !EOF()
   ? EMPL,NAME,SURN
   SKIP
END
```

: ตัวอย่างที่ 4.26

```
// อ่านข้อมูลมาพิมพ์ และมีลำดับ โดยใช้การวนลูปของ WHILE
USE EMPL
I = 0
DO WHILE !EOF()
  I++
  ? I,EMPL,NAME,SURN
  SKIP
ENDDO
```

: ตัวอย่างที่ 4.27

```
// อ่านข้อมูลมาพิมพ์ และมีลำดับ โดยใช้การวนลูปของ FOR
USE EMPL
FOR I = 1 TO RECCOUNT()
   ? I,EMPL,NAME,SURN
   SKIP
NEXT
```

: ตัวอย่างที่ 4.28

```
// อ่านข้อมูลมาพิมพ์ แล้วหยุดดูทีละหน้า หน้าละ 10 เรคอร์ด
USE EMPL
R = 7
WHILE !EOF()
    R++
    @ R,10 SAY STR(EMPL)+NAME+SURN
    SKIP
    IF R = 17
      INKEY(0)
      CLS
      R = 7
    ENDIF
END
```

: ตัวอย่างที่ 4.29

```
// อ่านข้อมูลมาพิมพ์บนพื้นสี ที่มีเงา
SET COLOR TO "/W"
CLS
SET COLOR TO "/N"
@ 5,10 CLEAR TO 22,70
SET COLOR TO "W/B"
@ 4,8 CLEAR TO 21,68
USE EMPL
I = 0
R = 6
WHILE !EOF()
   I++ ;   R++
   @ R,10 SAY STR(I)+STR(EMPL)+NAME+SURN
   SKIP
   IF R = 19
     INKEY(0)
     @ 4,8 CLEAR TO 21,68
     R = 6
   ENDIF
END
```

4.3.2 การอ่านข้อมูลมาพิมพ์ทางเครื่องพิมพ์

ในหัวข้อนี้ จะแสดงให้เห็นการพิมพ์ข้อมูลออกทางเครื่องพิมพ์ ในการนำข้อมูลออกเครื่องพิมพ์ ควรใช้คำส่ง SAY เพราะควบคุมจุดที่จะพิมพ์ได้แน่นอน ต่างกับการใช้ ? คู่กับ SET PRINT ON

: ตัวอย่างที่ 4.30

```
// อ่านข้อมูลมาพิมพ์ ออกทั้งจอภาพและเครื่องพิมพ์
USE EMPL
SET PRINT ON
WHILE !EOF()
  ? EMPL,NAME,SURN
  SKIP
END
SET PRINT OFF
```

: ตัวอย่างที่ 4.31

```
// อ่านข้อมูลมาพิมพ์ ออกทางเครื่องพิมพ์เท่านั้น
USE EMPL
SET DEVICE TO PRINTER
I = 0
WHILE !EOF()
   I++
   @ I,10 SAY STR(EMPL)+NAME+SURN
   SKIP
END
EJECT
SET DEVICE TO SCREEN
```

: ตัวอย่างที่ 4.32

```
// อ่านข้อมูลมาพิมพ์ ออกทางเครื่องพิมพ์ หรือจอภาพ โดยเลือกจากตัวเลือก
// โดยควบคุมการแสดงผลให้แยกกันได้อย่างชัดเจน และถูกต้อง
CLS
OPT = ALERT("เลือกผลลัพธ์ของรายงาน",{"จอภาพ","เครื่องพิมพ์","ยกเลิก"})
USE EMPL
IF OPT = 1
  SET COLOR TO "/W"
  CLS
  SET COLOR TO "/N"
  @ 5,10 CLEAR TO 22,70
  SET COLOR TO "W/B"
  @ 4,8 CLEAR TO 21,68
  PG = 15
  SETR  = 6
ELSE
  PG = 30
  SETR = 1
  SET DEVICE TO PRINTER
ENDIF
R = SETR
```

```
I = 0
WHILE !EOF() .AND. OPT != 3
   I++
   R++
   @ R,10 SAY STR(I)+STR(EMPL)+NAME+SURN
   SKIP
   IF R = PG
     IF OPT = 1
       INKEY(0)
       @ 4,8 CLEAR TO 21,68
     ELSE
       EJECT
     ENDIF
     R = SETR
   ENDIF
END
SET DEVICE TO SCREEN
```

4.3.3 การอ่านข้อมูลมาคำนวณแล้วพิมพ์

หัวข้อนี้ ได้นำแฟ้มมาคำนวณเพียงแฟ้มเดียว เพื่อหาค่าต่าง ๆ เช่น ภาษี โบนัส รายได้จริง เป็นต้น

แฟ้มเงินเดือน SALARY.DBF		
1. รหัสพนักงาน	EMPL	9(7)
2. เงินเดือนพนักงาน	SALARY	9(7)V99

: ตัวอย่างที่ 4.33

```
// คำนวณภาษี 7 % และรายได้ที่ต้องรับจริงของพนักงาน
USE SALARY
WHILE !EOF()
   TAX = SALARY * 0.07
   INCOME = SALARY - TAX
   ? EMPL,SALARY,TAX,INCOME
   SKIP
END
```

: ตัวอย่างที่ 4.34

```
// คำนวณผลรวมของเงินเดือน และผลรวมของรายได้
// สิ่งที่ต้องสังเกตในตัวอย่างนี้คือ การคำนวณผลรวมจะคำนวณในลูป
// แต่การพิมพ์ผลรวมต้องพิมพ์หลังจากที่อ่านข้อมูลจนหมดก่อน
USE SALARY
TINCOME = 0                          // ต้องกำหนด เพราะมีการเรียกใช้ตอนหาผลรวม
TSALARY = 0
WHILE !EOF()
   TAX = SALARY * 0.07
   INCOME = SALARY - TAX
   TINCOME = TINCOME + INCOME
   TSALARY = TSALARY + SALARY
   ? EMPL,SALARY,TAX,INCOME
   SKIP
END
? "TINCOME = ",TINCOME
? "TSALARY = ",TSALARY
```

& 4.4 การแต่งจอภาพอย่างง่าย

ความสวยงาม หรือสีสัน จำเป็นสำหรับการเขียนโปรแกรมใหญ่ ๆ เพื่อแยกความแตกต่าง หรือช่วยให้เข้าใจได้ง่ายขึ้นในรายละเอียดบางอย่าง จึงควรเข้าใจการใช้เส้น การตีกรอบ การระบายสี สีพื้น สีเงา วันที่ และเวลา

4.4.1 การตีกรอบ สีพื้น และสีเงา โดยทำเป็นนาฬิกา

ในหัวข้อนี้ จะทำนาฬิกาเดินบนพื้นสีตามวินาที

: ตัวอย่างที่ 4.35

```
// นาฬิกาบนพื้นสี เลิกทำงานเมื่อกดปุ่ม ESC
SET COLOR TO "N/W"
CLS                                  // ระบายสีพื้น
@ 0,0 TO 24,79                       // ตีกรอบนอก
SET COLOR TO "/N"
@ 5,7 CLEAR TO 23,74                 // ทำเงาก่อน
SET COLOR TO "GR+/B"
@ 4,5 CLEAR TO 22,72                 // ทำพื้นสีเป็นฉากหลัง
```

```
@ 1,1 SAY PADC("ABC COMPANY",78)
@ 2,1 TO 2,78                          // ขีดเส้นใต้ชื่อบริษัท
SET CURSOR OFF                         // ทำให้ตัวกระพริบ(CURSOR) หายไป
WHILE LASTKEY() != 27
  SETPOS(10,35)
  ?? TIME()                            // นาฬิกาจะเดินและเปลี่ยนเลขทุกวินาที
  INKEY(1)
END
```

4.4.2 การนำฟังก์ชันต่าง ๆ มาแสดงบนจอภาพ

หัวข้อนี้ จะนำฟังก์ชันที่น่าสนใจมาแสดงผลบนจอภาพ

: ตัวอย่างที่ 4.36

```
// โปรแกรมนี้มีฟังก์ชันต่าง ๆ ที่น่าสนใจหลายตัว
// โดยการระบายสียังคงใช้หลักการเดิม
SET COLOR TO W/B
CLS                                    // ระบายสีพื้น
@ 0,0 TO 24,79 DOUBLE                  // ตีกรอบนอก เส้นคู่
SET COLOR TO /N
@ 5,7 CLEAR TO 23,74                   // ทำเงาก่อน
SET COLOR TO GR+/BR
@ 4,5,22,72 BOX '..... '               // ทำพื้นสีเป็นฉากหลัง
@ 1,1 SAY PADC("CDF COMPANY",78)
@ 2,1 TO 2,78                          // ขีดเส้นใต้ชื่อบริษัท
SET CURSOR OFF                         // ทำให้ตัวกระพริบ(CURSOR) หายไป
@ 10,35 SAY "DATE   : "+DTOC(DATE())
@ 11,35 SAY "TIME   : "+TIME()
@ 12,35 SAY "SECOND  : "+STR(SECOND())
@ 13,35 SAY "OS     : "+OS()
@ 14,35 SAY "MEMORY  : "+STR(MEMORY())
@ 15,35 SAY "VERSION : "+VERSION()
@ 16,35 SAY "PRINTER : "               // ถ้าไม่เปิดเครื่องพิมพ์ ค่าที่ได้จะเห็น .F.
?? ISPRINTER()                         // ไม่สามารถแปลงด้วยฟังก์ชัน STR หรือ VAL ได้
```

& 4.5 การเขียนเมนู

การเขียนเมนูมีหลายวิธี ทั้งที่ใช้ฟังก์ชันของเครื่อง หรือควบคุมการกดแป้นพิมพ์เอง

4.5.1 เมนูมิติเดียว

หัวข้อนี้ จะแสดงให้เห็นการเขียนเมนูแบบต่าง ๆ ทั้ง GET , PROMPT และ ACHOICE

: ตัวอย่างที่ 4.37

```
// เมนูนี้ใช้การพิมพ์เมนูด้วยคำสั่ง TEXT และรับค่าด้วย ACCEPT
CLS
OPT = 0
DO WHILE .T.
  TEXT
   1. พิมพ์ 1 ถึง 10
   2. พิมพ์ สูตรคูณแม่ 2
   3. เลิกการทำงาน
  ENDTEXT
  ACCEPT TO OPT
  DO CASE
    CASE OPT = "1" ; FOR I = 1 TO 10 ; ? I ; NEXT
    CASE OPT = "2" ; FOR I = 1 TO 12 ; ? I," * 2 = ",I*2 ; NEXT
    CASE OPT = "3" ; QUIT
  ENDCASE
  INKEY(2)
ENDDO
```

: ตัวอย่างที่ 4.38

```
// เมนูนี้ใช้ WAIT รับค่า และใช้ IF เลือกทำงาน
CLS
OPT = 0
DO WHILE OPT != 51
  ? "1. พิมพ์ 1 ถึง 10"
  ? "2. พิมพ์ สูตรคูณแม่ 2"
  ? "3. เลิกการทำงาน"
  WAIT "รอรับ ...ตัวเลือก"                    // กด 1 จะได้เลข 49 เก็บใน OPT
  OPT = LASTKEY()
  IF OPT = 49 ; FOR I = 1 TO 10 ; ? I ; NEXT ; ENDIF
```

```
  IF OPT = 50 ; FOR I = 1 TO 12 ; ? I," * 2 = ",I*2 ; NEXT ; ENDIF
  INKEY(2)
ENDDO
```

: ตัวอย่างที่ 4.39

```
// โปรแกรมจะวิ่งไปตลอดเวลา แต่จะทำงานเมื่อมีการกดตัวเลือก
// จุดดีของโปรแกรมนี้คือ นาฬิกาจะเดินขณะที่ทำงานอยู่
CLS
OPT = 0
DO WHILE .T.
  @ 4,5 SAY TIME()
  @ 5,5 SAY "1. พิมพ์ 1 ถึง 10"
  @ 6,5 SAY "2. พิมพ์ สูตรคูณแม่ 2"
  @ 7,5 SAY "3. เลิกการทำงาน หรือปุ่ม ESC"
  // กด 1 จะได้เลข 49 เก็บใน OPT
  INKEY(1)
  IF LASTKEY() != 0
    OPT = LASTKEY()
    IF OPT = 49 ; FOR I = 1 TO 10 ; ? I ; NEXT ; ENDIF
    IF OPT = 50 ; FOR I = 1 TO 12 ; ? I," * 2 = ",I*2 ; NEXT ; ENDIF
    IF OPT = 51 .OR. OPT = 27 ; QUIT ; ENDIF
    INKEY(3)
    KEYBOARD CHR(0)          // ส่งค่าเข้าหน่วยความจำของแป้นพิมพ์
    INKEY()                  // เพื่อล้างค่าใน LASTKEY()
    CLS                      // ถ้าไม่ล้างจะทำให้ค่าใน LASTKEY() เท่าเดิมตลอดเวลา
  ENDIF
  OPT = 0
ENDDO
```

: ตัวอย่างที่ 4.40

```
// เลือกตัวเลือกแบบดึงลง (PULLDOWN MENU)
// ด้วยคำสั่ง PROMPT
CLS
OPT = 0
DO WHILE LASTKEY() != 27
  @ 4,5 SAY TIME()
  @ 5,5 PROMPT [1. พิมพ์ 1 ถึง 10]
  @ 6,5 PROMPT [2. พิมพ์ สูตรคูณแม่ 2]
  @ 7,5 PROMPT [3. เลิกการทำงาน หรือปุ่ม ESC]
```

```
  MENU TO OPT
  DO CASE
    CASE OPT = 1 ; FOR I = 1 TO 10 ; ? I ; NEXT
    CASE OPT = 2 ; FOR I = 1 TO 12 ; ? I," * 2 = ",I*2 ; NEXT
    CASE OPT = 3 ; QUIT
  ENDCASE
  INKEY(2)
ENDDO
```

: ตัวอย่างที่ 4.41

```
// เลือกตัวเลือกแบบดึงลง (PULLDOWN MENU)
// ด้วยคำสั่ง ACHOICE
CLS
OPT = 0
CHOICE = {"1. พิมพ์ 1 ถึง 10","2. พิมพ์ สูตรคูณแม่ 2",;
        "3. เลิกการทำงาน หรือปุ่ม ESC"}
DO WHILE LASTKEY() != 27
  OPT = ACHOICE (5,5,7,40,CHOICE)
  DO CASE
    CASE OPT = 1 ; FOR I = 1 TO 10 ; ? I ; NEXT
    CASE OPT = 2 ; FOR I = 1 TO 12 ; ? I," * 2 = ",I*2 ; NEXT
    CASE OPT = 3 ; QUIT
  ENDCASE
  INKEY(2)
ENDDO
```

: ตัวอย่างที่ 4.42

```
// เลือกตัวเลือกโดยควบคุมด้วยคำสั่งรับค่า LASTKEY()
// นาฬิกาเดินได้ตลอดเวลา
SET COLOR TO W/B
CLS
SET CURSOR OFF
OPT = 1
ARCHOICE = ARRAY(3)
ARCHOICE[1] = [1. พิมพ์ 1 ถึง 10]
ARCHOICE[2] = [2. พิมพ์ สูตรคูณแม่ 2]
ARCHOICE[3] = [3. เลิกการทำงาน หรือปุ่ม ESC]
```

```
DO WRITEMENU                        // พิมพ์เมนูครั้งแรก
DO WHILE LASTKEY() != 27
  @ 3,5 SAY TIME()                  // แสดงเวลาตลอดเวลา
  INKEY(0.2)                        // หยุดรับแป้นพิมพ์ทุก 0.2 วินาที
  IF LASTKEY() != 0
    DO CASE
      CASE LASTKEY() = 24           // กดลง
        IF OPT = 3                  // ถ้าถึงตัวล่างสุดแล้วยังกดลง
          OPT = 1                   // ให้ OPT ไปที่ตัวแรก
        ELSE
          OPT = OPT + 1
        ENDIF
      CASE LASTKEY() = 5            // กดขึ้น
        IF OPT = 1                  // ถ้าถึงตัวแรกแล้วยังกดขึ้น
          OPT = 3                   // ให้ OPT ไปยังตัวสุดท้าย
        ELSE
          OPT = OPT - 1
        ENDIF
      CASE LASTKEY() = 13           // กดเอ็นเทอร์ (ENTER)
        DO CASE
          CASE OPT <= 2
            @ 20,5 SAY "YOUR CHOICE IS "+STR(OPT)
          CASE OPT = 3
            @ 20,5 SAY "BYE BYE BYE"
            EXIT
        ENDCASE
      ENDCASE
      DO WRITEMENU
      KEYBOARD CHR(0)
      INKEY()                       // เพื่อตั้งค่าของ LASTKEY() ใหม่
    ENDIF
ENDDO
PROCEDURE WRITEMENU
 SET COLOR TO W/B
 I = 7                              // เมนูเริ่มพิมพ์บรรทัดที่ 7 หลักที่ 10
 FOR J = I TO I+2                   // เลข 2 หมายถึงจำนวนตัวเลือกมี 3 ตัว
    @ J,10 SAY ARCHOICE[J-I+1]      // พิมพ์เมนูทั้งหมดบนจอ
 NEXT
 SET COLOR TO B/W
 @ OPT+I-1,10 SAY ARCHOICE[OPT]     // พิมพ์ตัวเลือกปัจจุบัน
RETURN
```

4.5.2 เมนูหลายมิติ

หัวข้อนี้ จะแสดงการสร้างเมนูย่อยของตัวเลือกในเมนูหลักให้เห็นอย่างชัดเจน ทำให้ผู้อ่านสามารถเข้าใจและนำไปพัฒนาโปรแกรมที่เป็นระบบ เป็นไปได้ยิ่งขึ้น

: ตัวอย่างที่ 4.43

```
// เมนูหลายมิติ โดยเลือกตัวเลือกด้วยคำสั่ง PROMPT
// ซึ่งเมนูย่อย จะเรียงไปด้านข้าง ตามหลักที่ 10, 30 และ 50
// สามารถกดตัวอักษรตัวแรกของตัวเลือก เพื่อเลือกได้
SET CURSOR OFF
ARMAIN = 3
ARCM = ARRAY(ARMAIN)
ARCM[1] = [1. CHOICE1]
ARCM[2] = [2. CHOICE2]
ARCM[3] = [3. EXIT]
ARS1 = 2
ARC1 = ARRAY(ARS1)
ARC1[1] = [1. CHOICE11]
ARC1[2] = [2. EXIT TO MAIN]
ARS2 = 4
ARC2 = ARRAY(ARS2)
ARC2[1] = [1. CHOICE21]
ARC2[2] = [2. CHOICE22]
ARC2[3] = [3. CHOICE23]
ARC2[4] = [4. EXIT TO MAIN]
OPT = 1
DO WHILE LASTKEY() != 27
  SET COLOR TO W/B
  CLS
  DO CASE
    CASE OPT >= 1  .AND. OPT <= 3
      @ 7,10 PROMPT ARCM[1]              // พิมพ์เมนูบนจอภาพ
      @ 8,10 PROMPT ARCM[2]
      @ 9,10 PROMPT ARCM[3]
      MENU TO OPT
```

```
    CASE OPT >= 11 .AND. OPT <= 12
       @ 7,30 PROMPT ARC1[1]
       @ 8,30 PROMPT ARC1[2]
       MENU TO OPT
       OPT = OPT + 10
    CASE OPT >= 21 .AND. OPT <= 24
       @ 7,50 PROMPT ARC2[1]              // พิมพ์เมนูบนจอภาพ
       @ 8,50 PROMPT ARC2[2]
       @ 9,50 PROMPT ARC2[3]
       @ 10,50 PROMPT ARC2[4]
       MENU TO OPT
       OPT = OPT + 20
  ENDCASE
  // ====================================
  DO CASE
    CASE OPT = 1
         OPT = 11
    CASE OPT = 2
         OPT = 21
    CASE OPT = 3
         @ 20,5 SAY "BYE BYE BYE"
         EXIT
    CASE OPT=12 .OR. OPT=24                 // กลับเมนูหลัก
         OPT = 1
    CASE (OPT>=11 .AND. OPT<=12) .OR.;
       (OPT>=21 .AND. OPT<=24)              // กระทำตัวเลือก
         @ 23,5 SAY "OPT "+STR(OPT)
         INKEY(1)
   ENDCASE
ENDDO
```

: ตัวอย่างที่ 4.44

```
// เมนูหลายมิติ โดยเลือกตัวเลือกด้วยคำสั่งตรวจสอบแป้นพิมพ์ (LASTKEY)
// เมนูนี้นาฬิกาจะเดินตลอดเวลา  แต่ไม่ตรวจสอบปุ่มลูกศรขึ้นหรือลง
SET CURSOR OFF
ARMAIN = 3
ARCM = ARRAY(ARMAIN)
ARCM[1] = [1. CHOICE1]
ARCM[2] = [2. CHOICE2]
ARCM[3] = [3. EXIT]
ARS1 = 2
ARC1 = ARRAY(ARS1)
ARC1[1] = [1. CHOICE11]
ARC1[2] = [2. EXIT TO MAIN]
ARS2 = 4
ARC2 = ARRAY(ARS2)
ARC2[1] = [1. CHOICE21]
ARC2[2] = [2. CHOICE22]
ARC2[3] = [3. CHOICE23]
ARC2[4] = [4. EXIT TO MAIN]
OPT = 1                          // กำหนดตัวเลือกเริ่มต้น
TOPLMT = OPT                     // กำหนดบรรทัดบนสุดของเมนูปัจจุบัน
BOTTOMLMT = ARMAIN               // กำหนดบรรทัดล่างสุดของเมนูปัจจุบัน
DO WRITEMENU                     // พิมพ์เมนูครั้งแรก
DO WHILE LASTKEY() != 27
  @ 3,5 SAY TIME()               // แสดงเวลาตลอดเวลา
  INKEY(0.2)                     // หยุดรับแป้นพิมพ์ทุก 0.2 วินาที
  IF LASTKEY() != 0
    DO CASE
      CASE LASTKEY() = 24        // กดลง
        IF OPT = BOTTOMLMT       // ถ้าถึงตัวล่างสุดแล้วยังกดลง
          OPT = TOPLMT           // ให้ OPT ไปที่ตัวแรก
        ELSE
          OPT = OPT + 1
```

```
   ENDIF
CASE LASTKEY() = 5                  // กดขึ้น
   IF OPT = TOPLMT                  // ถ้าถึงตัวแรกแล้วยังกดขึ้น
     OPT = BOTTOMLMT                // ให้ OPT ไปยังตัวสุดท้าย
   ELSE
     OPT = OPT - 1
   ENDIF
CASE LASTKEY() = 13                 // กดเอ็นเทอร์ (ENTER)
   DO CASE
     CASE OPT = 1
        OPT = 11  ;   TOPLMT = OPT
        BOTTOMLMT = ARS1 + OPT - 1
     CASE OPT = 2
        OPT = 21  ;   TOPLMT = OPT
        BOTTOMLMT = ARS2 + OPT - 1
     CASE OPT = 3
        @ 20,5 SAY "BYE BYE BYE"
        EXIT
     CASE OPT=10+ARS1 .OR. OPT=20+ARS2      // กลับเมนูหลัก
        OPT = 1
        TOPLMT = OPT
        BOTTOMLMT = ARMAIN
     CASE OPT >= 11 .AND. OPT<=10+ARS1      // กระทำตัวเลือก
        @ 20,5 SAY "OPT "+STR(OPT)
        INKEY(3)
     CASE OPT >= 21 .AND. OPT<=20+ARS2      // กระทำตัวเลือก
        @ 20,5 SAY "OPT "+STR(OPT)
        INKEY(3)
   ENDCASE
ENDCASE
DO WRITEMENU
KEYBOARD CHR(0)
INKEY()                             // เพื่อตั้งค่าของ LASTKEY() ใหม่
```

```
   ENDIF
ENDDO
PROCEDURE WRITEMENU
 SET COLOR TO W/B
 CLS
 DO CASE
   CASE OPT >= 1  .AND. OPT <= ARMAIN
      I = 7                                    // เมนูเริ่มพิมพ์บรรทัดที่ 7
      FOR J = I TO I+ARMAIN-1
         @ J,10 SAY ARCM[J-I+1]                // พิมพ์เมนูบนจอภาพ
      NEXT
      TEMPAR = ARCM
   CASE OPT >= 11 .AND. OPT <= ARS1 + 10
      I = 12                                   // เมนูเริ่มพิมพ์บรรทัดที่ 7
      FOR J = I TO I+ARS1-1
         @ J,10 SAY ARC1[J-I+1]                // พิมพ์เมนูบนจอภาพ
      NEXT
      TEMPAR = ARC1
   CASE OPT >= 21 .AND. OPT <= ARS2 + 20
      I = 18                                   // เมนูเริ่มพิมพ์บรรทัดที่ 18
      FOR J = I TO I+ARS2-1
         @ J,10 SAY ARC2[J-I+1]                // พิมพ์เมนูบนจอภาพ
      NEXT
      TEMPAR = ARC2
 ENDCASE
 SET COLOR TO B/W
 @ I+MOD(OPT,10)-1,10 SAY TEMPAR[MOD(OPT,10)]   // พิมพ์ตัวเลือกปัจจุบัน
RETURN
```

: ตัวอย่างที่ 4.45

```
// ตัวอย่างนี้ ได้รวมเอาความสามารถต่าง ๆ ของเมนูไว้
// สามารถตรวจสอบ การกดปุ่มซ้าย ขวา บน ล่าง
// สามารถ กดปุ่ม ESC เพื่อเลิกการทำงานได้ทันที
// ยังคงทำให้นาฬิกาเดินได้ตลอดเวลา
SET CURSOR OFF
ARMAIN = 3 ;  ARCM = ARRAY(ARMAIN)
ARCM[1] = [1. CHOICE1]
ARCM[2] = [2. CHOICE2]
ARCM[3] = [3. EXIT]
ARS1 = 2 ;  ARC1 = ARRAY(ARS1)
ARC1[1] = [1. CHOICE11]
ARC1[2] = [2. EXIT TO MAIN]
ARS2 = 4  ;  ARC2 = ARRAY(ARS2)
ARC2[1] = [1. CHOICE21]
ARC2[2] = [2. CHOICE22]
ARC2[3] = [3. CHOICE23]
ARC2[4] = [4. EXIT TO MAIN]
OPT = 1                          // กำหนดตัวเลือกเริ่มต้น
TOPLMT = OPT                     // กำหนดบรรทัดบนสุดของเมนูปัจจุบัน
BOTTOMLMT = ARMAIN               // กำหนดบรรทัดล่างสุดของเมนูปัจจุบัน
ALLSUB = 2                       // จำนวนเมนูย่อยทั้งหมด
DO WRITEMENU                     // พิมพ์เมนูครั้งแรก
DO WHILE LASTKEY() != 27
  @ 1,5 SAY TIME()               // แสดงเวลาตลอดเวลา
  INKEY(0.2)                     // หยุดรับแป้นพิมพ์ทุก 0.2 วินาที
  IF LASTKEY() != 0
    // กรณีกดปุ่มใด ๆ จะมาตรวจสอบในส่วนนี้
    DO CASE
      CASE LASTKEY() = 27
         EXIT
      CASE LASTKEY() = 4         // กดขวา
         IF OPT < 10             // ตัวเลือกในเมนูหลัก
```

```
      DO PRESSDOWN
    ELSE
      IF (OPT-MOD(OPT,10))/10 = ALLSUB      // ข้ามไปเมนูทางขวาหรือเริ่มใหม่
        OPT = 11
      ELSE
        OPT = (OPT-MOD(OPT,10))+11
      ENDIF
    ENDIF
CASE LASTKEY() = 19                 // กดซ้าย
    IF OPT < 10                     // ตัวเลือกในเมนูหลัก
      DO PRESSUP
    ELSE
      IF (OPT-MOD(OPT,10))/10 = 1   // ข้ามไปเมนูทางซ้ายหรือสุดท้าย
        OPT = ALLSUB*10+1
      ELSE
        OPT = (OPT-MOD(OPT,10))-9
      ENDIF
    ENDIF
CASE LASTKEY() = 24                 // กดลง
    DO PRESSDOWN
CASE LASTKEY() = 5                  // กดขึ้น
    DO PRESSUP
// กรณีกด ENTER จะมาเลือกประมวลผลในส่วนนี้
CASE LASTKEY() = 13                 // กดเอ็นเทอร์ (ENTER)
    DO CASE
      CASE OPT = 1
         OPT = 11
      CASE OPT = 2
         OPT = 21
      CASE OPT = 3
         @ 20,5 SAY "BYE BYE BYE"
         EXIT
      CASE OPT=10+ARS1 .OR. OPT=20+ARS2  // กลับเมนูหลัก
```

```
            OPT = 1
          CASE OPT >= 11 .AND. OPT<=10+ARS1  // กระทำตัวเลือก
            @ 20,5 SAY "OPT "+STR(OPT)
            INKEY(3)
          CASE OPT >= 21 .AND. OPT<=20+ARS2  // กระทำตัวเลือก
            @ 20,5 SAY "OPT "+STR(OPT)
            INKEY(3)
        ENDCASE
      ENDCASE
      // ตั้งค่า TOPLMT และ BOTTOMLMT ใหม่
      DO CASE
        CASE OPT <10
          TOPLMT = 1      ; BOTTOMLMT = ARMAIN
        CASE OPT >= 11 .AND. OPT<=10+ARS1
          TOPLMT = 11     ; BOTTOMLMT = 10+ARS1
        CASE OPT >= 21 .AND. OPT<=20+ARS2
          TOPLMT = 21     ; BOTTOMLMT = 20+ARS2
      ENDCASE
      DO WRITEMENU                    // เขียนเมนูใหม่
      KEYBOARD CHR(0)
      INKEY()                         // เพื่อตั้งค่าของ LASTKEY() ใหม่
    ENDIF
ENDDO
//===================================
PROCEDURE PRESSDOWN
  IF OPT = BOTTOMLMT                  // ถ้าถึงตัวล่างสุดแล้วยังกดลง
    OPT = TOPLMT                      // ให้ OPT ไปที่ตัวแรก
  ELSE
    OPT = OPT + 1
  ENDIF
RETURN
PROCEDURE PRESSUP
  IF OPT = TOPLMT                     // ถ้าถึงตัวแรกแล้วยังกดขึ้น
```

```
    OPT = BOTTOMLMT                    // ให้ OPT ไปยังตัวสุดท้าย
  ELSE
    OPT = OPT - 1
  ENDIF
RETURN
//====================================
PROCEDURE WRITEMENU
 SET COLOR TO W/B
 CLS
 FOR J = 1 TO ARMAIN
    @ 2,J*10 SAY ARCM[J]                    // พิมพ์เมนูบนจอภาพ
 NEXT
 I = 4                                      // เมนูเริ่มพิมพ์บรรทัดที่ 4
 DO CASE
   CASE OPT >= 11 .AND. OPT <= ARS1 + 10
      FOR J = I TO I+ARS1-1
         @ J,10 SAY ARC1[J-I+1]             // พิมพ์เมนูบนจอภาพ
      NEXT
      TEMPAR = ARC1
   CASE OPT >= 21 .AND. OPT <= ARS2 + 20
      FOR J = I TO I+ARS2-1
         @ J,20 SAY ARC2[J-I+1]             // พิมพ์เมนูบนจอภาพ
      NEXT
      TEMPAR = ARC2
 ENDCASE
 SET COLOR TO B/W
 IF OPT > 10                                // สั่งเลือกตัวเลือกย่อย
   @ I+MOD(OPT,10)-1,OPT-MOD(OPT,10);
    SAY TEMPAR[MOD(OPT,10)]                 // พิมพ์ตัวเลือกปัจจุบัน
   SET COLOR TO W/N
   @ 2,(OPT-MOD(OPT,10)) ;                  // พิมพ์หัวของเมนูย่อย
    SAY ARCM[(OPT-MOD(OPT,10))/10]
 ELSE                                       // ขณะเลือกเมนูหลัก
```

```
   @ 2,MOD(OPT,10)*10 SAY ARCM[MOD(OPT,10)]   // พิมพ์หัวของเมนูย่อย
 ENDIF
 SET COLOR TO B/W
RETURN
```

& 4.6 การเชื่อมแฟ้ม 2 แฟ้ม

การเชื่อมแฟ้มเป็นหลักการที่ต้องเข้าใจในการศึกษาฐานข้อมูล เพราะฐานข้อมูลคือการนำแฟ้มหลายแฟ้มมาใช้ร่วมกัน ทำให้การเก็บข้อมูลไม่ซ้ำซ้อน และสะดวกใช้ เช่นระบบเงินเดือน จะเก็บแฟ้มชื่อ และแฟ้มเงินเดือนไว้ต่างกัน หากต้องการพิมพ์เงินเดือนพร้อมชื่อ ต้องเชื่อมแฟ้มทั้ง 2 เข้าด้วยกัน

ตัวอย่างโครงสร้าง และข้อมูลจากแฟ้มชื่อ NAME.DBF

EMPID	EMPLNAME	EMPLSURN
101	นาย ปิติ	หมายตา
102	นาย สมปอง	น้ำฝน
103	นาย แท่งทอง	อติชาติ
104	น.ส.ชมพู	บางบัว
105	น.ส.วาสนา	ฤดี
106	นาย แก้วกล้า	หทัยชนัน
107	น.ส.คมคาย	มัธยม

ตัวอย่างโครงสร้าง และข้อมูลจากแฟ้มเงินเดือน SALARY.DBF

EMPID	SALARY
101	12500
103	21000
104	7500
106	8600
107	6750

ตัวอย่างโครงสร้าง และข้อมูลจากแฟ้มขาย SALE.DBF

SALEID	EMPID	PRODUCT	PRICE	QUANT
100001	101	รถเก๋ง	60,000	1
100002	104	วิทยุ	5,000	8
100003	101	รถเข็น	1,000	6
100004	103	รถเก๋ง	57,000	1
100005	107	เรือ	25,000	1
100006	107	ทีวี	5,600	4
100007	106	กระดาษ	100	300

4.6.1 การเชื่อมแฟ้มด้วยคำสั่ง SEEK

การใช้คำสั่ง SEEK ค้นหาข้อมูล ต้องกำหนดให้แฟ้มที่ถูกค้น เปิดแฟ้มข้อมูลพร้อมแฟ้มดรรชนีไว้ โดยมีฟิลด์หลักในแฟ้มดรรชนีตรงกับรหัสที่ต้องการค้นหา โดยส่งตัวแปรผ่านคำสั่ง SEEK เข้าไปค้นหาในแฟ้มดรรชนี

: ตัวอย่างที่ 4.46

```
// ตัวอย่างนี้ใช้แฟ้มชื่อและแฟ้มเงินเดือน โดยแฟ้มชื่อเปิดแฟ้มดรรชนีไว้
// ซึ่งหน้าที่หลักของโปรแกรมนี้คือ แสดงข้อมูลทั้งหมดจากแฟ้มเงินเดือน
SELE 1; USE NAME INDEX INAME
SELE 2; USE SALARY                    // ไม่จำเป็นต้องเปิดแฟ้มดรรชนีให้แฟ้มนี้
DO WHILE !EOF()
  _EMPID = EMPID
  ? EMPID,SALARY
  SELE 1
    SEEK _EMPID
    IF FOUND()
      ?? EMPLNAME,EMPLSURN
    ENDIF
  SELE 2 ;  SKIP
ENDDO
```

: ตัวอย่างที่ 4.47

```
// ตัวอย่างนี้จะค้นหาจนพบแล้ว รวบรวมค่าต่าง ๆ นำมาพิมพ์พร้อม ๆ กัน
// โดยใช้ตัวอย่างจากแฟ้มชื่อและแฟ้มเงินเดือน ต้องการทุกเรคอร์ดจากแฟ้มเงินเดือน
SELE 1; USE NAME INDEX INAME
SELE 2; USE SALARY                    // ไม่จำเป็นต้องเปิดแฟ้มดรรชนีให้แฟ้มนี้
SETPOS(5,2)
_TINCOME = 0
DO WHILE !EOF()
  _EMPID = EMPID
  _SALARY = SALARY
  SELE 1
    SEEK _EMPID
    IF FOUND()
      _EMPLNAME = EMPLNAME
      _EMPLSURN = EMPLSURN
```

```
    ELSE
      _EMPLNAME = SPACE(20)
      _EMPLSURN = SPACE(20)
    ENDIF
  SELE 2
  _TAX = _SALARY * 0.07
  _INCOME = _SALARY - _TAX
  _TINCOME = _TINCOME + _INCOME
  @ ROW()+1,2 SAY _EMPID
  @ ROW()  ,9 SAY _EMPLNAME + _EMPLSURN
  @ ROW()  ,50 SAY STR(_SALARY) + "  " + STR(_TAX)
  @ ROW()  ,70 SAY _INCOME
  SKIP
ENDDO
@ ROW()+1,70 SAY _TINCOME
```

4.6.2 การเชื่อมแฟ้มด้วยคำสั่ง LOCATE

การใช้คำสั่ง LOCATE ค้นหาข้อมูล ไม่จำเป็นต้องเปิดแฟ้มดรรชนีเหมือนคำสั่ง SEEK เพราะคำสั่งนี้จะวิ่งเข้าไปเปรียบเทียบกับข้อมูลทีละตัวจนกว่าจะพบ ทำให้ผลการทำงานช้ากว่าการค้นหาข้อมูลจากแฟ้มดรรชนี แต่การใช้ LOCATE จะมีความยืดหยุ่นมากเพราะค้นหาในฟิลด์ต่าง ๆ ได้ง่ายและเขียนได้ในหลาย ๆ รูปแบบ

: ตัวอย่างที่ 4.48

```
// ตัวอย่างนี้จะค้นหาจนพบแล้ว รวบรวมค่าต่าง ๆ นำมาพิมพ์พร้อม ๆ กัน
// โดยใช้ตัวอย่างจากแฟ้มชื่อและแฟ้มขาย ต้องการทุกเรคอร์ดจากแฟ้มขาย
SELE 1; USE NAME
SELE 2; USE SALE
DO WHILE !EOF()
  _EMPID = EMPID
  ? SALEID,EMPID,PRODUCT,PRICE,QUANT
  SELE 1  // แฟ้มชื่อ
    LOCATE FOR EMPID = _EMPID
    IF FOUND()
      ?? EMPLNAME,EMPLSURN
    ENDIF
  SELE 2 ;  SKIP                          // แฟ้มขาย
ENDDO
```

: ตัวอย่างที่ 4.49

```
// ตัวอย่างนี้ แสดงให้เห็นการใช้คำสั่ง LOCATE REST
// โดยระบุรหัสพนักงาน แล้วค้นหาในแฟ้มขายว่าผลงานการขายของพนักงานมีอะไรบ้าง
// ตัวอย่างนี้จะใช้แฟ้มขายแฟ้มเดียว แล้วค้นหารหัสพนักงานขายจนกว่าจะไม่พบ
_EMPID = 0
@ 5,5 SAY "รหัสพนักงาน : " GET _EMPID
READ
USE SALE
LOCATE FOR EMPID = _EMPID
DO WHILE FOUND()
  ? SALEID,EMPID,PRODUCT,PRICE,QUANT
  SKIP
  LOCATE REST FOR EMPID =_EMPID
ENDDO
```

: ตัวอย่างที่ 4.50

```
// ตัวอย่างนี้ต้องการรายงานข้อมูลการขายของพนักงานทุกคน
// โดยรายงานว่าพนักงานแต่ละคนมีสถิติการขายเป็นอย่างไร
// จึงใช้แฟ้มหลักเป็นแฟ้มชื่อ ไปค้นหาในแฟ้มขาย
SELE 1 ; USE SALE
SELE 2 ; USE NAME
DO WHILE !EOF()
  _EMPID = EMPID
  ? EMPID
  SELE 1
    LOCATE FOR EMPID = _EMPID
    DO WHILE FOUND()
      ? SALEID,PRODUCT,PRICE,QUANT
      SKIP
      LOCATE REST FOR EMPID =_EMPID
    ENDDO
  SELE 2
  SKIP
ENDDO
```

4.6.3 การเชื่อมแฟ้มด้วยคำสั่ง SET FILTER TO

คำสั่งนี้ใช้เลือกข้อมูลที่ตรงตามเงื่อนไข ซึ่งใช้งานได้กว้างมากกับระบบฐานข้อมูล แต่ใช้ร่วมกับคำสั่ง FOUND() ไม่ได้ หากต้องการตรวจสอบจำนวนเรคอร์ดที่ตรงกับช่วงข้อมูลที่ต้องการ สามารถใช้คำสั่ง COUNT

เมื่อใช้คำสั่งเลือกช่วงของข้อมูล เรคอร์ดทั้งหมดที่โปรแกรมมองเห็นจะมีเฉพาะเรคอร์ดที่ตรงตามเงื่อนไข จึงนำไปใช้หรือตรวจสอบค่าต่าง ๆ ได้ สิ่งที่ต้องระวังกับคำสั่งนี้คือ หลังจากระบุขอบเขตของข้อมูลแล้วต้องเลื่อนตัวชี้ไปที่เรคอร์ดแรก ด้วยคำสั่ง GO TOP

คำสั่งนี้ต่างกับคำสั่ง SEEK และ LOCATE ในวัตถุประสงค์ นั่นคือคำสั่งนี้ไม่ได้ตรวจสอบว่ามีข้อมูลในแฟ้มตามที่ต้องการค้นหาหรือไม่ แต่คำสั่งนี้ใช้ในการเลือกข้อมูลตามช่วงที่ระบุเงื่อนไข เช่นเลือกข้อมูลที่เงินเดือนมากกว่า 5000 บาท แต่คำสั่ง SEEK และ LOCATE จะค้นหาว่ามีเรคอร์ดที่ต้องการหรือไม่ หากมีจะเลื่อนตัวชี้ไปที่เรคอร์ดนั้น

: ตัวอย่างที่ 4.51

```
// ตัวอย่างนี้จะใช้แฟ้มชื่อและแฟ้มเงินเดือน โดยนำรหัสจากแฟ้มเงินเดือนไป
// ค้นหาในแฟ้มชื่อโดยใช้คำสั่ง SET FILTER TO
// ซึ่งหน้าที่หลักของโปรแกรมนี้คือ แสดงข้อมูลทั้งหมดจากแฟ้มเงินเดือน
SELE 1; USE NAME
SELE 2; USE SALARY
DO WHILE !EOF()
  _EMPID = EMPID
  ? EMPID,SALARY
  SELE 1
    SET FILTER TO EMPID = _EMPID
    COUNT TO CNTREC
    GO TOP
    IF CNTREC > 0
      ?? EMPLNAME,EMPLSURN
    ENDIF
  SELE 2
  SKIP
ENDDO
```

: ตัวอย่างที่ 4.52

```
// ตัวอย่างนี้ แสดงให้เห็นการใช้คำสั่ง SET FILTER TO
// โดยระบุรหัสพนักงาน แล้วค้นหาในแฟ้มขายว่าผลงานการขายของพนักงานมีอะไรบ้าง
// ตัวอย่างนี้จะใช้แฟ้มขายแฟ้มเดียว แล้วค้นหารหัสพนักงานขายจนกว่าจะไม่พบ
```

```
_EMPID = 0
@ 5,5 SAY "รหัสพนักงาน : " GET _EMPID
READ
USE SALE
SET FILTER TO EMPID = _EMPID
GO TOP
DO WHILE !EOF()
  ? SALEID,EMPID,PRODUCT,PRICE,QUANT
  SKIP
ENDDO
```

: ตัวอย่างที่ 4.53

```
// ตัวอย่างนี้ต้องการรายงานข้อมูลการขายของพนักงานทุกคน
// โดยรายงานว่าพนักงานแต่ละคนมีสถิติการขายเป็นอย่างไร
// จึงใช้แฟ้มหลักเป็นแฟ้มชื่อ ไปค้นหาในแฟ้มขาย
// ตัวอย่างนี้จะเลือกข้อมูลการขายโดยใช้คำสั่ง SET FILTER
SELE 1 ; USE SALE
SELE 2 ; USE NAME
DO WHILE !EOF()
  _EMPID = EMPID
  ? EMPID
  SELE 1
    SET FILTER TO EMPID = _EMPID
    COUNT TO CNTREC                    // เก็บจำนวนเรคอร์ดที่อยู่ในกลุ่ม
    GO TOP
    FOR I = 1 TO CNTREC
      ? I,SALEID,PRODUCT,PRICE,QUANT
    NEXT
  SELE 2
  SKIP
ENDDO
```

4.6.4 การเชื่อมแฟ้มด้วยคำสั่ง SET RELATION

การระบุความสัมพันธ์อย่างน้อย 2 แฟ้ม ทำให้ง่ายต่อการอ้างถึงข้อมูลที่สัมพันธ์กัน โดยไม่ต้องใช้คำสั่งเชื่อมข้อมูลแบบต่าง ๆ แต่แฟ้มระบุความสัมพันธ์ด้วย ต้องมีการเปิดแฟ้มพร้อมแฟ้มดรรชนีไว้แล้ว และต้องมีฟิลด์หลักของแฟ้มดรรชนีตรงกับฟิลด์หนึ่งในแฟ้มหลัก จึงจะอ้างข้อมูลถึงกันได้

: ตัวอย่างที่ 4.54

```
// การเชื่อมแฟ้มชื่อและเงินเดือนเข้าด้วยกัน
// โดยอ่านข้อมูลจากแฟ้มเงินเดือนเป็นหลัก
SELE 1 ; USE NAME INDEX INAME
SELE 2 ; USE SALARY
      SET RELATION TO EMPID INTO NAME
DO WHILE !EOF()
  ? SALEID,EMPID,NAME->EMPLNAME,NAME->EMPLSURN,PRODUCT,PRICE,QUANT
  SKIP
ENDDO
```

: ตัวอย่างที่ 4.55

```
// การเชื่อมแฟ้มชื่อ เงินเดือน และขายเข้าด้วยกัน
// โดยใช้แฟ้มขายเป็นหลัก
SELE 1 ; USE NAME INDEX INAME
SELE 2 ; USE SALARY INDEX ISALARY
SELE 3 ; USE SALE
      SET RELATION TO EMPID INTO NAME,EMPID INTO SALARY
DO WHILE !EOF()
  _TAX = SALARY->SALARY * 0.07
  ? SALEID
  ?? EMPID,NAME->EMPLNAME,NAME->EMPLSURN
  ?? SALARY->SALARY , _TAX
  ?? PRODUCT,PRICE,QUANT
  SKIP
ENDDO
```

& 4.7 การสร้างเสียง

คำสั่ง TONE สามารถสร้างเสียงตามความถี่ เพื่อให้เกิดเสียงต่าง ๆ ได้ตามต้องการ สามารถสร้างเสียงเพลงต่าง ๆ ได้ แต่ลักษณะเสียงขึ้นกับลำโพงที่เครื่องคอมพิวเตอร์ใช้งาน และความเข้าใจในเรื่องของดนตรี ตัวโน๊ต และการทำนองของเพลง

: ตัวอย่างที่ 4.56

```
// ตัวอย่างการสร้างเสียงเพลงอย่างง่าย ๆ โดยขอใช้ท่อนแรกของเพลงใกล้รุ่งมาเป็นตัวอย่าง
// ตัวอย่างนี้ให้เสียงไม่ดีนัก ต้องการให้ผู้ศึกษาปรับปรุงให้ดีขึ้นต่อไป
// ตัวโน๊ตเหล่านี้ ผู้เขียนเทียบเสียงกับตัวเลขในหนังสือ "เล่นดนตรีวิธีง่าย"
  N1 =130.80 ;  N1X =138.60 ;   N2 =146.80 ;  N2X =155.60 ;   N3 =164.80
  N4 =174.60 ;  N4X =185.00 ;   N5 =196.00 ;  N5X =207.70
  N6 =200.00 ;  N6X =233.10 ;   N7 =246.90
  U1 =261.70 ;  U1X =277.20 ;   U2 =293.70 ;  U2X =311.10 ;   U3 =329.60
  U4 =349.20 ;  U4X =370.00 ;   U5 =392.00 ;  U5X =415.30
  U6 =440.00 ;  U6X =466.20 ;   U7 =493.90 ;  UU1 =523.30
// เพลงพระราชนิพนธ์ : ใกล้รุ่ง
  ?T(N6);?T(U1);?T(U2);?T(N6);INKEY(0.5)                    //ได้ยินเสียงแว่ว
  ?T(N5);?T(N4);?T(N5);?T(N5X);?T(N6);?T(N6);INKEY(0.5)     //ดังแผ่วมาแต่ไกล ๆ
  ?T(N4);?T(N5);?T(N6);?T(U1);INKEY(0.5)                    //ชุ่มชื่นฤทัย
  ?T(U2);?T(U1);?T(N6);?T(N5);INKEY(0.5)                    //หวานใดจะปาน
  ?T(N6);?T(U1);?T(N6);?T(N5);?T(N1);?T(N2);?T(N1);?T(N6)   //ฟังเสียงบรรเลงขับเพลงประสาน
  ?T(N6);?T(U1);?T(N6);?T(N5);?T(N1);?T(N2);?T(N1);?T(N4)   //จากทิพย์วิมาน ประทานกล่อมใจ [11]
FUNCTION T
PARAM K
  DU = 9
  TONE(K,DU)
  INKEY(0.02)
  IF LASTKEY() = 27
    QUIT
  ENDIF
RETURN (K)
```

[11] สิงขร สอนขัน, **เล่นดนตรีอย่างง่าย เล่ม 1**, รุ่งแสงการพิมพ์, กรุงเทพฯ, หน้า 36

& 4.8 โปรแกรมข้อสอบ

โปรแกรมข้อสอบเป็นโปรแกรมที่มีหลักการง่าย ๆ คืออ่านข้อมูลจากแฟ้มมาแสดง แล้วรอรับค่า หากค่าที่รับเข้ามาไม่ตรงกับที่ระบุไว้จะไม่ได้คะแนน หากตอบถูกคะแนนจะเพิ่มขึ้น ทำให้โปรแกรมนี้น่าสนใจในการศึกษาหรือเพิ่มส่วนเก็บประวัติของผู้ทำข้อสอบไว้ และมีการเฉลยทุกครั้งที่ทำเสร็จ บางครั้งอาจเพิ่มส่วนของการสุ่มข้อสอบไม่ให้การทำข้อสอบในแต่ละครั้งออกข้อที่ซ้ำ ๆ กัน ท้าทายยิ่งขึ้น

4.8.1 โปรแกรมข้อสอบธรรมดา

แฟ้มคำถามแบบที่ 1 QUES1.DBF		
1. คำถาม	QUES	X(20)
2. ตัวเลือกที่ 1	CHOICE1	X(20)
3. ตัวเลือกที่ 2	CHOICE2	X(20)
4. ตัวเลือกที่ 3	CHOICE3	X(20)
5. คำตอบ	ANS	X(1)

: ตัวอย่างที่ 4.57

```
// อ่านข้อมูลจากแฟ้มมาแสดงทีละเรคอร์ด หากตอบถูกจะได้คะแนนเพิ่ม 1 คะแนน
 USE QUES1
 OK = 0
 DO WHILE !EOF()
   _ANS = " "
   ? "คำถาม",QUES
   ? "1.",CHOICE1
   ? "2.",CHOICE2
   ? "3.",CHOICE3
   ACCEPT TO _ANS
   IF ANS = _ANS
     OK++
   ENDIF
   ? "คะแนนของท่านคือ ",OK
   SKIP
 ENDDO
```

: ตัวอย่างที่ 4.58

```
// อ่านข้อมูลจากแฟ้มมาแสดงทีละเรคอร์ด หากตอบถูกจะได้คะแนนเพิ่ม 1 คะแนน
// โดยแสดงข้อมูลด้วย SAY
  USE QUES1
  OK = 0
  DO WHILE !EOF()
    CLS ;    _ANS = " "
    @ 5,5 SAY "คำถาม" + QUES
    @ 6,5 SAY "1." + CHOICE1
    @ 7,5 SAY "2." + CHOICE2
    @ 8,5 SAY "3." + CHOICE3
    @ 9,5 SAY "ท่านเลือก : " GET _ANS
    READ
    IF ANS = _ANS  ;    OK++ ;   TONE(200,5)  ;   ENDIF
    @ 10,5 SAY "คะแนนของท่านคือ " + STR(OK)
    SKIP  ;     INKEY(5)
  ENDDO
```

4.8.2 โปรแกรมข้อสอบสุ่ม

ในส่วนนี้ยังใช้แฟ้ม QUES1.DBF เหมือนเดิม แต่มีการสุ่มตัวเลขจากวินาทีมาใช้ ทำให้ทุกครั้งที่ทำข้อสอบจะขึ้นคำถามไม่ซ้ำกัน ท้าทายยิ่งขึ้น

: ตัวอย่างที่ 4.59

```
// ตัวอย่างนี้เป็นการสุ่มคำถาม จากคำถามทั้งหมด โดยทำไปจนหมดทุกข้อ
// บรรทัดสุดท้ายสรุปให้รู้ว่าทำได้กี่เปอร์เซนต์
  USE QUES1
  ARNUM = ARRAY(RECCOUNT())
  FOR I = 1 TO RECCOUNT()                    // ส่วนของการสุ่มค่า
    RND = MOD(SECOND() * 100,RECCOUNT())+1
    IF ASCAN(ARNUM,RND) = 0
      ARNUM[I] = RND
    ELSE
      I--
    ENDIF
  NEXT
```

```
OK = 0
FOR I = 1 TO RECCOUNT()                    // ส่วนของการตั้งคำถามให้ตอบ
  GO ARNUM[I]
  _ANS = " "
  ? "คำถาม",QUES
  ? "1.",CHOICE1
  ? "2.",CHOICE2
  ? "3.",CHOICE3
  ACCEPT TO _ANS
  IF ANS = _ANS
    OK++
  ENDIF
  ? "คะแนนของท่านคือ ",OK
NEXT
? "ท่านทำได้ทั้งหมด ",OK/RECCOUNT()*100," เปอร์เซนต์"
```

: ตัวอย่างที่ 4.60

```
// ตัวอย่างนี้จะเลือกข้อสอบมาเพียง 5 ข้อจากคำถามทั้งหมด
// ส่งเสียงเมื่อทำข้อสอบถูกต้อง
  USE QUES1
  ALLQUEST = 5                             // จำนวนข้อสอบที่ต้องการทำ
  ARNUM = ARRAY(ALLQUEST)
  I = 0
  DO WHILE I < ALLQUEST                    // ส่วนของการสุ่มค่า
    RND = MOD(SECOND() * 100,RECCOUNT())+1
    IF ASCAN(ARNUM,RND) = 0
      I++
      ARNUM[I] = RND
    ENDIF
  ENDDO
  OK := I := 0
  DO WHILE I < ALLQUEST
    CLS ; I++
    GO ARNUM[I]
```

```
  _ANS = " "
  @ 5,5 SAY "คำถามข้อที่ " + STR(I) + ". " + QUES
  @ 6,5 SAY "1." + CHOICE1
  @ 7,5 SAY "2." + CHOICE2
  @ 8,5 SAY "3." + CHOICE3
  @ 9,5 SAY "ท่านเลือก : " GET _ANS
  READ
  IF ANS = _ANS
    OK++ ;    TONE(200,5)
  ENDIF
  @ 10,5 SAY "คะแนนของท่านคือ " + STR(OK)
  INKEY(5)
ENDDO
```

4.8.3 โปรแกรมข้อสอบมีเฉลยเมื่อทำเสร็จ

เมื่อทำข้อสอบเสร็จแล้วในแต่ละข้อย่อมได้ผล 2 ประการคือ ถูกหรือผิด ไม่ว่าจะได้ผลอย่างไร ผู้ทำข้อสอบย่อมต้องการทราบเหตุผล หรือคำเฉลย ในหัวข้อนี้จึงแสดงวิธีการเขียนโปรแกรมโดยเฉลยข้อสอบเมื่อทำเสร็จแล้ว

แฟ้มคำถามแบบที่ 2 QUES2.DBF

1. คำถาม	QUES	X(20)
2. ตัวเลือกที่ 1	CHOICE1	X(20)
3. ตัวเลือกที่ 2	CHOICE2	X(20)
4. ตัวเลือกที่ 3	CHOICE3	X(20)
5. คำตอบ	ANS	X(1)
6. เฉลยบรรทัดที่ 1	EXPLAIN1	X(20)
7. เฉลยบรรทัดที่ 2	EXPLAIN2	X(20)
8. เฉลยบรรทัดที่ 3	EXPLAIN3	X(20)

: ตัวอย่างที่ 4.61

```
// แสดงส่วนเฉลยอย่างง่าย เมื่อทำข้อสอบเสร็จแล้วในแต่ละข้อ
  USE QUES2
  OK = 0
  DO WHILE !EOF()
    _ANS = " "
    ? "คำถาม",QUES
```

```
    ? "1.",CHOICE1
    ? "2.",CHOICE2
    ? "3.",CHOICE3
    ACCEPT TO _ANS
    IF ANS = _ANS
      ? "ท่านตอบถูก"
      OK++
    ENDIF
    ? "คะแนนของท่านคือ ",OK
    ? EXPLAIN1                          // ส่วนที่พิมพ์คำเฉลยบนจอภาพ
    ? EXPLAIN2
    ? EXPLAIN3
    SKIP
  ENDDO
```

: ตัวอย่างที่ 4.62

```
// แสดงส่วนเฉลยในอีกรูปแบบหนึ่ง และเลือกตัวเลือกแบบ PULLDOWN CHOICE
  USE QUES2
  OK = 0
  DO WHILE !EOF()
    CLS
    _ANS = 0
    @ 5,5 SAY "คำถาม" + QUES
    @ 6,5 PROMPT "1." + CHOICE1
    @ 7,5 PROMPT "2." + CHOICE2
    @ 8,5 PROMPT "3." + CHOICE3
    MENU TO _ANS                        // ค่าที่ได้จะเป็นตัวเลข
    IF ANS = LTRIM(STR(_ANS))
      OK++ ;    TONE(200,5)
    ENDIF
    @ 10,5 SAY "คะแนนของท่านคือ " + STR(OK)
    @ 13,5 SAY "คำเฉลย :"
    @ 14,5 SAY EXPLAIN1
    @ 15,5 SAY EXPLAIN2
    @ 16,5 SAY EXPLAIN3
    SKIP  ;     INKEY(5)
  ENDDO
```

4.8.4 โปรแกรมข้อสอบเก็บประวัติ

บ่อยครั้งที่ผู้ทำข้อสอบ ต้องการเก็บผลการทำข้อสอบของตน เพื่อดูความก้าวหน้า หรือต้องการแข่งขันกับเพื่อนหลาย ๆ คน เมื่อทำข้อสอบเสร็จแล้วโปรแกรมเก็บข้อมูลต่าง ๆ เช่นทำเมื่อใด โดยใคร และได้คะแนนเท่าใด

แฟ้มประวัติ HIST.DBF		
1. วันที่ทำ	QDATE	X(8)
2. เวลาที่ทำ	QTIME	X(8)
3. ทำโดย	NAMET	X(20)
4. คะแนนที่ได้	SCORE	99

: ตัวอย่างที่ 4.63

```
// ตัวอย่างนี้สามารถเก็บข้อมูลการทำข้อสอบของผู้ทำข้อสอบ
// เมื่อเริ่มโปรแกรมจะถามชื่อผู้ทำ ตอนจบจะเก็บข้อมูลโดยอัตโนมัติ
// โดยไม่สนใจว่าจะได้คะแนนเท่าใด หรือเต็มใจที่จะบันทึกหรือไม่
 _NAMET = SPACE(20)
 @ 5,5 SAY "ชื่อของผู้ทำข้อสอบ : " GET _NAMET
 READ
 _QDATE = DTOC(DATE())   ;   _QTIME = TIME()
 USE QUES1
 OK = 0
 DO WHILE !EOF()
   _ANS = " "
   ? "คำถาม",QUES
   ? "1.",CHOICE1
   ? "2.",CHOICE2
   ? "3.",CHOICE3
   ACCEPT TO _ANS
   IF ANS = _ANS  ;  ? "ท่านตอบถูก"  ;  OK++  ;    ENDIF
   ? "คะแนนของท่านคือ ",OK
   SKIP
 ENDDO
 USE HIST
 APPEND BLANK
 REPLACE QDATE WITH _QDATE,QTIME WITH _QTIME,;
       NAMET WITH _NAMET,SCORE WITH OK
 CLOSE ALL
```

& 4.9 การปรับปรุงข้อมูล

การปรับปรุงข้อมูลเป็นหลักการที่สำคัญอย่างยิ่ง ในการเขียนโปรแกรมเพื่อจัดการกับฐานข้อมูล เพื่อให้ข้อมูลมีความถูกต้องและเป็นปัจจุบันอยู่ตลอดเวลา ทำให้นำข้อมูลไปใช้ได้เหมาะสมกับเวลาและสถานที่ ... ที่สุด

การปรับปรุงข้อมูลในตอนนี้จะไม่นำตัวอย่างที่ซับซ้อนมาแสดง ผู้อ่านศึกษาเพิ่มเติมได้จากการปรับปรุงข้อมูลในส่วนของกรณีศึกษา ทั้ง 2 เรื่อง ซึ่งมีวิธีการหลาย ๆ อย่างในการปรับปรุงข้อมูล

4.9.1 การเพิ่มข้อมูลในแฟ้ม

โดยปกติการเพิ่มข้อมูลจะทำการเก็บค่าต่าง ๆ ในตัวแปร เมื่อพอใจแล้วจะทำการเขียนทับลงไปบนเรคอร์ดใหม่ที่เป็นเรคอร์ดว่างเปล่า

: ตัวอย่างที่ 4.64

```
// ถ้าแฟ้มนี้ชื่อ FILE1 และต้องการเพิ่มข้อมูลเพียง 1 เรคอร์ดเท่านั้น
  USE FILE1
  _FLD1 = 0
  _FLD2 = SPACE(20)
  @ 5,5 GET _FLD1
  @ 6,5 GET _FLD2
  READ
  APPEND BLANK
  REPLACE FLD1 WITH _FLD1, ;
        FLD2 WITH _FLD2
  CLOSE ALL
```

เปล่า

: ตัวอย่างที่ 4.65

```
// ถ้าแฟ้มนี้ชื่อ FILE1 และต้องการเพิ่มข้อมูลเพียง 2 เรคอร์ด ที่มีข้อมูลซ้ำกัน
  USE FILE1
  _FLD1 = 123
  _FLD2 = “ABC”
  APPEND BLANK
  REPLACE FLD1 WITH _FLD1, FLD2 WITH _FLD2
  APPEND BLANK
  REPLACE FLD1 WITH _FLD1, FLD2 WITH _FLD2
  CLOSE ALL
```

4.9.2 การลบข้อมูล

การลบข้อมูลคือ การระบุรหัสที่ต้องการลบ แล้วนำไปค้นหาในแฟ้ม หากพบจะลบข้อมูล

: ตัวอย่างที่ 4.66

```
// ตัวอย่างนี้ลบได้เพียงเรคอร์ดเดียว
  _FLD1 = 0
  @ 5,5 SAY "ระบุข้อมูลที่ท่านต้องการลบ " GET _FLD1
  READ
  USE FILE1
  LOCATE FOR FLD1 = _FLD1
  IF FOUND()
    DELETE ;       PACK
  ENDIF
```

: ตัวอย่างที่ 4.67

```
// ตัวอย่างนี้ลบเป็นกลุ่มตามเงื่อนไข
  _FLD1 = 0
  @ 5,5 SAY "ระบุกลุ่มข้อมูลที่ท่านต้องการลบ " GET _FLD1
  READ
  USE FILE1
  DELETE  ALL  FOR  E STATUS  = _FLD1
  PACK
```

4.9.3 การแก้ไขข้อมูล

การแก้ไขคือ การระบุรหัสที่ต้องการลบ แล้วนำไปค้นหา หากเจอจะนำค่าในเรคอร์ดทั้งหมดมาเก็บในตัวแปร แล้วรอรับการปรับปรุงตัวแปร เมื่อปรับปรุงแล้วจะเขียนทับลงไปในเรคอร์ดเดิม

: ตัวอย่างที่ 4.68

```
// ตัวอย่างนี้แก้ไขข้อมูลได้เพียงเรคอร์ดเดียว
  _FLD1 = 0
  @ 5,5 SAY "ระบุข้อมูลที่ท่านต้องการแก้ไข " GET _FLD1
  READ
  USE FILE1
  LOCATE FOR FLD1 = _FLD1
  IF FOUND()
    _FLD2 = FLD2
    @ 7,5 SAY "แก้ไข FLD1 " GET _FLD1
    @ 8,5 SAY "แก้ไข FLD2 " GET _FLD2
```

```
    READ
    IF UPDATED()
      REPLACE FLD1 WITH _FLD1 , FLD2 WITH _FLD2
    ENDIF
  ENDIF
  CLOSE ALL
```

: ตัวอย่างที่ 4.69

```
// ตัวอย่างนี้แก้ไขข้อมูลทุกระเบียนด้วยเงื่อนไขเดียวกัน
  _FLD1 = 0
  @ 5,5 SAY "ระบุจำนวนเงินเดือนที่ขึ้นให้ทุกคน" GET _FLD1
  READ
  USE FILE1
  REPLACE FLD1 WITH  FLD1  +  _FLD1
  CLOSE ALL
```