บทที่ Â

ฟังก์ชัน (Function)

<u>จุดประสงค์เชิงพฤติกรรม</u>

1. เข้าใจพื้นฐานการฟังก์ชันใช้งาน
2. ใช้อ้างอิงเมื่อมีความต้องการ แต่นึกรูปแบบฟังก์ชันไม่ออก
3. ใช้อ้างอิงเมื่อต้องการหาฟังก์ชัน ไปแก้ปัญหาในการเขียนโปรแกรม

<u>หัวข้อในบทเรียน</u>

ฟังก์ชันทั้งหมด 194 ฟังก์ชัน

จาก ฟังก์ชัน AADD () จนถึง ฟังก์ชัน YEAR()

ตัวอย่างโปรแกรมเกี่ยวกับฟังก์ชัน 227 ตัวอย่าง

<u>หมายเหตุ</u>

มีฟังก์ชันใช้งานมากมายที่ Clipper มีให้บริการ มีฟังก์ชันหลายตัวที่มีหน้าที่คล้ายกันทั้งในฟังก์ชัน หรือคำสั่งเอง ก็เพื่อสร้างทางเลือกให้กับผู้ใช้ในการเขียนโปรแกรม และหลายคนมีพื้นฐานมาจากภาษาอื่น ก็สามารถเลือกใช้ฟังก์ชันที่ตนคุ้นเคยได้โดยง่าย บทนี้ต้องการเป็นแหล่งอ้างอิง เพราะมีนักเรียน หรือนักพัฒนาน้อยคน ที่จะจำฟังก์ชันทั้งหมดได้ครบ หรือจำรูปแบบการใช้ได้ทั้งหมด .. แหล่งอ้างอิงจึงมีความสำคัญ

บทที่ 3 ฟังก์ชัน (Function)

ฟังก์ชัน คือการประมวลผลกับค่าที่ส่งให้ฟังก์ชันดำเนินการ แล้วคืนค่าที่ผ่านการประมวลผลกลับมา หรือการหาค่าโดยใช้ฟังก์ชัน นั่นหมายความว่า ฟังก์ชันมีหน้าที่คืนค่าให้กับโปรแกรม แล้วโปรแกรมสามารถนำค่าไปใช้ได้ตามต้องการ แต่จะคือค่ามาในรูปแบบใด ขึ้นอยู่กับหน้าที่ของฟังก์ชันนั้น ๆ

& 3.1 AADD() [10]

! รูปแบบ

AADD(<ชุดอาเรย์เป้าหมาย>,<นิพจน์ที่เพิ่มเข้าไป>) --> ค่าของนิพจน์

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ เพิ่มสมาชิกอีก 1 ตัวเข้าไปต่อท้ายอาเรย์ชุดเดิมโดยจะเพิ่มค่าเข้าไปต่ออาเรย์ตัวสุดท้าย แล้วส่งค่าคืน ซึ่งเป็นค่าของนิพจน์ที่เพิ่มเข้าไป

: ตัวอย่างที่ 3.1

```
AR := {}                        // อาเรย์ว่างเปล่า
AADD(AR,'BAT')                  // {'BAT'}
AADD(AR,'RAT')                  // {'BAT','RAT'}
AADD(AR,2500)                   // {'BAT','RAT',2500}
FOR I = 1 TO 3
  ?? AR[I]                      // BATRAT     2500
NEXT
```

: ตัวอย่างที่ 3.2

```
AR := {}
TEST := 'ABC'
AADD(AR,'A')
AADD(AR,5)
? AADD(AR,'XYZ')                // XYZ
AADD(AR,{'B',TEST,23})
? AR[1]                         // A
? AR[2]                         //        5
? AR[3]                         // XYZ
? AR[4,1]                       // B
? AR[4,2]                       // ABC
? AR[4,3]                       //        23
? '====='                       // =====
```

---

[10] นิพนธ์ กิตติปภัสสร และ อนันต์ อุตตะมะ, อ้างแล้วในเชิงอรรถที่ (1), หน้า 135

& 3.2 ABS()

! รูปแบบ

ABS(<นิพจน์จำนวนเลข>) --> จำนวนเลขบวกหรือศูนย์

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ แปลงตัวแปรตัวเลขเป็นค่าสัมบูรณ์

: ตัวอย่างที่ 3.3

```
TEST1 := 10
TEST2 := 12
? (TEST1 - TEST2)              //    -2
? ABS(TEST1 - TEST2)           //     2
? ABS(TEST1 - 100)             //    90
? ABS(3 - 7)                   //     4
? ABS(0)                       //     0
? ABS(5)                       //     5
? ABS(22 / 7)                  //    3.14
? ABS(2.555 - 3.4444)          //    0.8894
```

& 3.3 ACHOICE()

! รูปแบบ

ACHOICE(<มุมบน>, <มุมซ้าย>, <มุมล่าง>, <มุมขวา>,
<ชุดอาเรย์แสดงเมนู>) --> จำนวนเลขของตัวเลือกที่ถูกเลือก

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ สั่งให้นำ ARRAY มาเป็นเมนูแบบ PULLDOWN MENU

: ตัวอย่างที่ 3.4

```
CLS
DO WHILE .T.
  OPT := ACHOICE(5,10,7,30,{'ADD','DELETE';
      ,'CHANGE','PRINT','QUIT'})
  OPTPROGRAM := {'ADDPRO','DELPRO','CHGPRO','PRTPRO'}
  DO CASE
    CASE OPT = 1
        DO ADDPRO
    CASE OPT = 2
```

```
            DO DELPRO
        CASE OPT = 3
            DO CHGPRO
        CASE OPT = 4
            DO PRTPRO
        CASE OPT = 5
            EXIT
      ENDCASE
    ENDDO
```

: ตัวอย่างที่ 3.5

```
    CLS
    OPTPROGRAM := {'ADDPRO','DELPRO','CHGPRO','PRTPRO'}
    DO WHILE .T.
      OPT := ACHOICE(5,10,7,30,{'ADD','DELETE';
          ,'CHANGE','PRINT','QUIT'})
      IF OPT = 5 .OR. LASTKEY() = 27
        EXIT
      ENDIF
      ? (OPTPROGRAM[OPT])
      DO CASE
        CASE OPT = 1;   DO ADDPRO
        CASE OPT = 2;   DO DELPRO
        CASE OPT = 3;   DO CHGPRO
        CASE OPT = 4;   DO PRTPRO
      ENDCASE
    ENDDO
```

: ตัวอย่างที่ 3.6

```
    CLS
    OPTMENU    := {'ADD','DELETE','CHANGE','PRINT','QUIT'}
    OPTPROGRAM := {'ADDPRO','DELPRO','CHGPRO','PRTPRO'}
    DO WHILE .T.
      OPT := ACHOICE(5,10,7,30,OPTMENU)
      DO CASE
```

```
        CASE OPT = 5 .OR. LASTKEY() =27 ; EXIT
        CASE OPT = 1;   DO ADDPRO
        CASE OPT = 2;   DO DELPRO
        CASE OPT = 3;   DO CHGPRO
        CASE OPT = 4;   DO PRTPRO
      ENDCASE
    ENDDO
```

& 3.4 ACLONE()

! รูปแบบ

ACLONE(<ชุดอาเรย์ต้นแบบ>) --> ชุดอาเรย์ชุดใหม่

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ คัดลอกอาเรย์ทั้งชุดไปสร้างเป็นชุดใหม่

: ตัวอย่างที่ 3.7

```
OLD := {'ABC',12,2.5}
NEW := OLD                      // 2 ตัวแปร แต่ข้อมูลชุดเดียว
OLD[1] := 'DEF'
? OLD[1], OLD[2], OLD[3]        // DEF  12   2.5
? NEW[1], NEW[2], NEW[3]        // DEF  12   2.5
```

: ตัวอย่างที่ 3.8

```
OLD := {'ABC',12,2.5}
NEW := ACLONE(OLD)              // เก็บข้อมูลไว้ 2 ชุด
OLD[1] := 'DEF'
? OLD[1], OLD[2], OLD[3]        // DEF  12   2.5
? NEW[1], NEW[2], NEW[3]        // ABC  12   2.5
```

& 3.5 ACOPY()

! รูปแบบ

ACOPY(<ชุดอาเรย์ต้นแบบ>, <ชุดอาเรย์เป้าหมาย>,
[<เริ่มที่ตำแหน่ง>], [<จำนวนตำแหน่ง>],
[<ตำแหน่งปลายทาง>]) --> ชุดอาเรย์ปลาย

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ คัดลอกอาเรย์ไปทับอาเรย์ชุดใหม่

: ตัวอย่างที่ 3.9

```
OLD := {'A','B','C'}
NEW := {1,2,3}
ACOPY(OLD,NEW,1,1,1)          // OLD ตัวแรกตัวเดียวทับ NEW ตัวเดียว
? NEW[1],NEW[2],NEW[3]        // A  2  3
ACOPY(OLD,NEW,1,1,2)          // OLD ตัวแรกทับ NEW 2 ชุด ไปตำแหน่งที่ 2
? NEW[1],NEW[2],NEW[3]        // A  A  3
ACOPY(OLD,NEW,1,2,1)          // OLD จากตัวแรกไป 2 ตัว ไปตำแหน่งที่ 1
? NEW[1],NEW[2],NEW[3]        // A  B  3
ACOPY(OLD,NEW,2,2,1)          // OLD จากตัวที่ 2 ไป 2 ตัว ไปตำแหน่งที่ 1
? NEW[1],NEW[2],NEW[3]        // B  C  3
ACOPY(OLD,NEW,3,1,1)          // OLD จากตัวที่ 3 ไป 1 ตัว ไปตำแหน่งที่ 1
? NEW[1],NEW[2],NEW[3]        // C  C  3
ACOPY(OLD,NEW,1,1,3)          // OLD จากตัวที่ 1 ไปตำแหน่งที่ 3
? NEW[1],NEW[2],NEW[3]        // C  C  A
```

& 3.6 ADEL()

! รูปแบบ

ADEL(<ชุดอาเรย์เป้าหมาย>, <ตำแหน่งที่>) --> ชุดอาเรย์ปลายทาง

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ลบอาเรย์บางตัวออกจากชุดอาเรย์ที่มีอยู่

: ตัวอย่างที่ 3.10

```
OLD := {'A','B','C'}
NEW := {1,2,3}
ACOPY(OLD,NEW,1,1,3)
? NEW[1],NEW[2],NEW[3]        //  1  2  A
ADEL(OLD,2)
ADEL(NEW,3)
? ; FOR I = 1 TO 3 ; ?? OLD[I] ; NEXT   //  A  C  NIL
? ; FOR I = 1 TO 3 ; ?? NEW[I] ; NEXT   //  1  2  NIL
```

& 3.7 ADIR()

! รูปแบบ

ADIR([<ชื่อแฟ้มที่ต้องการอ่านเข้ามา>],
[<ชุดอาเรย์ที่รับชื่อแฟ้ม>], [<ชุดอาเรย์ที่รับขนาด>],
[<ชุดอาเรย์ที่รับวันที่>], [<ชุดอาเรย์ที่รับเวลา>],
[<ชุดอาเรย์ที่รับลักษณะเฉพาะ>]) --> จำนวนแฟ้มที่รับเข้ามา

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ นำข้อมูลแฟ้มเช่น ชื่อแฟ้ม หรือขนาด เก็บลงอาเรย์

: ตัวอย่างที่ 3.11

```
LOCAL FPRG[20]                    // จองเนื้อที่เป็นอาเรย์ให้ FPRG
LOCAL FPRGTEST := ARRAY(20)       // จองเนื้อที่เป็นอาเรย์อีกแบบหนึ่ง
ADIR("*.PRG",FPRG)
FOR I = 1 TO 3                    // X.PRG
   ? FPRG[I]                      // Y.PRG
NEXT                              // Z.PRG
```

: ตัวอย่างที่ 3.12

```
LOCAL FINDDIR := "C:\WINDOWS\*.EXE"
LOCAL FDIRS[ADIR(FINDDIR)]        // ทำให้จองเนื้อที่ให้อาเรย์ได้ถูกต้อง
ADIR(FINDDIR,FDIRS)
AEVAL(FDIRS,{|ELEMENT| QOUT(ELEMENT)}) // QOUT ให้พิมพ์อาเรย์ได้
```

: ตัวอย่างที่ 3.13

```
LOCAL FINDDIR := "C:\WINDOWS\*.EXE"
LOCAL FDIRS[ADIR(FINDDIR)]        // ทำให้จองเนื้อที่ให้อาเรย์ได้ถูกต้อง
LOCAL FSIZES [ADIR(FINDDIR)]
LOCAL FDATES [ADIR(FINDDIR)]
LOCAL FTIMES [ADIR(FINDDIR)]
LOCAL FATTRIBUTES [ADIR(FINDDIR)]
ADIR(FINDDIR,FDIRS,FSIZES,FDATES,FTIMES,FATTRIBUTES)
AEVAL(FDIRS ,{|ELEMENT| QOUT(ELEMENT)})          ; INKEY(0)
AEVAL(FSIZES,{|ELEMENT| QOUT(ELEMENT)})          ; INKEY(0)
AEVAL(FDATES,{|ELEMENT| QOUT(ELEMENT)})          ; INKEY(0)
AEVAL(FTIMES,{|ELEMENT| QOUT(ELEMENT)})          ; INKEY(0)
AEVAL(FATTRIBUTES,{|ELEMENT| QOUT(ELEMENT)}) ; INKEY(0)
```

: ตัวอย่างที่ 3.14

```
LOCAL FINDDIR
ACCEPT TO FINDDIR
FDIRS       = ARRAY(ADIR(FINDDIR))
FSIZES      = ARRAY(ADIR(FINDDIR))
FDATES      = ARRAY(ADIR(FINDDIR))
FTIMES      = ARRAY(ADIR(FINDDIR))
FATTRIBUTES = ARRAY(ADIR(FINDDIR))
ADIR(FINDDIR,FDIRS,FSIZES,FDATES,FTIMES,FATTRIBUTES)
FOR I = 1 TO ADIR(FINDDIR)
  ? FDIRS[I],FSIZES[I],FDATES[I],FTIMES[I],FATTRIBUTES[I]
NEXT
```

& 3.8 AEVAL()

! รูปแบบ

AEVAL(<ชุดอาเรย์>, <นิพจน์ที่นำอาเรย์ไปใช้>,
[<เริ่มที่ตำแหน่ง>], [<จำนวนตำแหน่ง>]) --> ค่าจากนิพจน์

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ สั่งประมวลผลกับอาเรย์ด้วยชุดคำสั่งแบบบล็อก(BLOCK)

: ตัวอย่างที่ 3.15

```
LOCAL     ELE    := "{|ELEMENT| QOUT(ELEMENT)}"
LOCAL     ELEMIN :="{|ELEMENT| IF(ELEMENT<MIN,MIN:=ELEMENT,) }"
LOCAL     ELECNT :="{|ELEMENT| CNT:=CNT+1}"
// LOCAL  ELECNT:="{|ENIL|IF(ENIL<>NIL,CNT:=CNT+1)}"เฉพาะที่ != NIL
LOCAL     ELESUM :="{|ELEMENT| SUM := SUM + ELEMENT}"
MAX = 0
MIN = 9999
CNT = 0
SUM = 0
X := {40,12,8,5,21}
Y := ACLONE(X)
AEVAL( X , {|ELEMENT| QOUT(ELEMENT)})  ; INKEY(0)
AEVAL( X , &ELE)                       ; INKEY(0)
AEVAL( X , {|ELEMENT| IF(ELEMENT > MAX ,MAX := ELEMENT,)})
```

```
AEVAL( X , &ELEMIN)
AEVAL( X , &ELECNT)
AEVAL( X , &ELESUM)
? MAX , MIN , CNT , SUM/CNT
```

& 3.9 AFIELDS()

! รูปแบบ

AFIELDS([<ชุดอาเรย์เก็บชื่อฟิลด์>], [<ชุดอาเรย์เก็บแบบ>],
[<ชุดอาเรย์เก็บความกว้าง>], [<ชุดอาเรย์เก็บจำนวนทศนิยม>])
--> จำนวนฟิลด์ของแฟ้มปัจจุบัน

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ใส่โครงสร้างแฟ้มปัจจุบันลงไปใน ARRAY

: ตัวอย่างที่ 3.16

```
USE FILEA
AFLDNAME := ARRAY(AFIELDS())
AFLDTYPE := ARRAY(AFIELDS())
AFLDWIDTH:= ARRAY(AFIELDS())
AFLDDEC  := ARRAY(AFIELDS())
AFIELDS(AFLDNAME,AFLDTYPE,AFLDWIDTH,AFLDDEC)
FOR I = 1 TO AFIELDS()              // FIELD1      C  10
   ?  AFLDNAME[I],AFLDTYPE[I]       // FIELD2      N  10  0
   ?? STR(AFLDWIDTH[I],10)          // FIELD3      N  10  0
   ?? IF(AFLDTYPE[I]='N',STR(AFLDDEC[I],10),'')
NEXT
```

& 3.10 AFILL()

! รูปแบบ

AFILL(<ชุดอาเรย์เป้าหมาย>, <ค่าที่จะนำเข้าชุดอาเรย์>,
[<เริ่มที่ตำแหน่ง>], [<จำนวนตำแหน่ง>]) --> ชุดอาเรย์เป้าหมาย

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ใส่ค่าที่กำหนดลงไปใน ARRAY โดยรูปแบบข้อมูลแต่ละครั้งกำหนดได้เพียงแบบเดียว

: ตัวอย่างที่ 3.17

```
LOCAL  X[5]                     // NIL NIL NIL NIL NIL
LOCAL  Y[5]                     // NIL NIL NIL NIL NIL
AFILL(X,12)                     // นำเลข 12 ใส่ในอาเรย์ทุกตัว
FOR I = 1 TO 5 ; ?X[I] ; NEXT   // 12 12 12 12 12
```

```
AFILL(X,5,2,3)                  // ใส่เลข 5 เริ่มที่ตำแหน่ง 2 ไป 3 ตัว
FOR I = 1 TO 5 ; ?X[I] ; NEXT   // 12  5  5  5 12
AFILL(X,.T.,3,2)                // ใส่ .T. เริ่มที่ตำแหน่ง 3 ไป 2 ตัว
FOR I = 1 TO 5 ; ?X[I] ; NEXT   // 12  5 .T. .T. 12
```

& 3.11 AINS()

! รูปแบบ

AINS(<ชุดอาเรย์เป้าหมาย>, <ตำแหน่งที่>) --> ATARGET

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ แทรกค่าว่าง(NIL) ลงไปในอาเรย์ ทำให้ตัวสุดท้ายหายไปเสมอ

: ตัวอย่างที่ 3.18

```
X := {1,2,3,4,5}
AINS(X,2)                       // แทรกเข้าไปในตำแหน่งที่ 2
FOR I = 1 TO 5 ; ?X[I] ; NEXT   // 1  NIL  2  3  4
AINS(X,4)                       // แทรกเข้าไปในตำแหน่งที่ 4
FOR I = 1 TO 5 ; ?X[I] ; NEXT   // 1  NIL  2  NIL  3
```

& 3.12 ALERT()

! รูปแบบ

ALERT(<ข้อความ> [, <ชุดอาเรย์ตัวเลือก>]) --> ค่าตัวเลือก

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ การรับค่าจากตามตัวเลือกในแบบไดอาลอกบล็อก (DIALOG BOX)

: ตัวอย่างที่ 3.19

```
X = {'A','B','C','D'}                  // ต้องไม่เกิน 4 ค่า
OPT = ALERT('TEST ALERT',X)
? OPT                                  // ผลคือ 1 ถึง 4 เท่านั้น
HEADOPT = 'ตัดสินใจสำหรับรายงานนี้'
TXTOPT = {'SCREEN','PRINTER','EDIT OPTION','CANCEL'}
OPT = ALERT(HEADOPT,TXTOPT)
DO CASE
  CASE OPT = 1 ; ? 'พิมพ์รายงานทางจอภาพ'
  CASE OPT = 2 ; ? 'พิมพ์รายงานทางเครื่องพิมพ์'
```

```
  CASE OPT = 3 ; ? 'กลับไปแก้ไขตัวเลือก'
  CASE OPT = 4 ; ? 'ยกเลิก และกลับเมนู'
  OTHERWISE    ; ? 'ท่านไม่เลือกปุ่มใดเลย รู้ตัวรึเปล่า'
ENDCASE
```

: ตัวอย่างที่ 3.20

```
HEADOPT = 'ตัดสินใจสำหรับรายงานนี้ และไม่อนุญาติให้กด ESC'
TXTOPT = {'SCREEN','PRINTER','EDIT OPTION','CANCEL'}
OPT = 4
DO WHILE .T.
  OPT = ALERT(HEADOPT,TXTOPT)
  DO CASE
    CASE OPT = 1 ; ? 'พิมพ์รายงานทางจอภาพ'
    CASE OPT = 2 ; ? 'พิมพ์รายงานทางเครื่องพิมพ์'
    CASE OPT = 3 ; ? 'กลับไปแก้ไขตัวเลือก'
    CASE OPT = 4 ; ? 'ยกเลิก และกลับเมนู'; EXIT
    OTHERWISE
     ALERT('ท่านไม่เลือกปุ่มใดเลย รู้ตัวรึเปล่า')
  ENDCASE
ENDDO
```

& 3.13 ALIAS()

! รูปแบบ

ALIAS([<พื้นที่ทำงานที่>]) --> สมนาม

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ส่งชื่อพื้นที่ทำงานปัจจุบัน

: ตัวอย่างที่ 3.21

```
SELE 1
USE FILEA
? ALIAS()                       // FILEA
SELE 2
USE FILEB ALIAS AREANAME2
? ALIAS()                       // AREANAME2
SELE 3
USE FILEC ALIAS AREANAME3       // ชื่อ ALIAS ต้องไม่เกิน 10
```

```
? ALIAS()                    // AREANAME3
SELE AREANAME2
? ALIAS()                    // AREANAME2
```

& 3.14 ALLTRIM()

! รูปแบบ

ALLTRIM(<ข้อความ>) --> ข้อความที่ผ่านการตัดช่องว่าง

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ นำค่าว่างออกจากตัวแปรตัวอักษร ทั้งด้านซ้าย และด้านขวา

: ตัวอย่างที่ 3.22

```
X = "ABC "
Y = " ABC"
Z = " ABC "
? LEN(ALLTRIM(X))                    //    3
? ALLTRIM(X)+ALLTRIM(Y)              // ABCABC
? LEN(LTRIM(X))                      //    4
? LEN(RTRIM(X))                      //    3
? LEN(LTRIM(Z))                      //    4
? LEN(RTRIM(Z))                      //    4
? LEN(ALLTRIM(Z))                    //    3
? ALLTRIM(X)+ALLTRIM(Y)+ALLTRIM(Z)   // ABCABCABC
? RTRIM(X)+RTRIM(Y)+RTRIM(Z)         // ABC ABC ABC
? LTRIM(X)+LTRIM(Y)+LTRIM(Z)         // ABC ABCABC
```

& 3.15 ARRAY()

! รูปแบบ

ARRAY(<จำนวนสมาชิก> [, <จำนวนสมาชิก>...]) --> ชุดอาเรย์

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ กำหนดให้มีการใช้ตัวแปร ARRAY

: ตัวอย่างที่ 3.23

```
AR := ARRAY(3)               // การกำหนดอาเรย์มิติเดียวมาใช้
AR := {NIL , NIL , NIL}      // หรือกำหนดแบบใช้ {}
FOR I = 1 TO 3
   ACCEPT TO AR[I]
```

```
NEXT
FOR I = 1 TO 3
   ? AR[I]
NEXT
```

: ตัวอย่างที่ 3.24

```
AR := ARRAY(2,3)                    // การกำหนดอาเรย์หลายมิติมาใช้
AR := {{NIL,NIL,NIL},{NIL,NIL,NIL}}
AR := {ARRAY(3),ARRAY(3)}
// นำข้อมูลจากโครงสร้างแฟ้มเก็บในอาเรย์หลายมิติ
USE FILEA
FLDAR := ARRAY(4,AFIELDS())
AR1 := FLDAR[1]
AR2 := FLDAR[2]
AR3 := FLDAR[3]
AR4 := FLDAR[4]
AFIELDS(AR1,AR2,AR3,AR4)
FOR I = 1 TO AFIELDS()
   ? FLDAR[1,I],FLDAR[2,I],FLDAR[3,I],FLDAR[4,I]
NEXT
FOR I = 1 TO AFIELDS()
   ?
   FOR J = 1 TO 4
      ?? FLDAR[J,I]
   NEXT
NEXT
AEVAL(FLDAR,{|ELE|QOUT(ELE[1],ELE[2])})
```

& 3.16 ASC()

! รูปแบบ

ASC(<ตัวอักษร>) --> เลขลำดับในตารางแอสกี (ASCII)

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ แปลงค่าให้เป็นรหัส ASCII

: ตัวอย่างที่ 3.25

```
? ASC("A")          //    65
? ASC("A")          //    97
? ASC("ABCDEF")     //    97
? ASC("Z")          //    90
```

```
? ASC("Z123")                    //    90
? ASC("Z")                       //   122
? ASC("Z") - ASC("A")            //    25
? ASC("Z") - ASC("A")            //    25
FOR I = ASC("A") TO ASC("Z")
   ? I                           //  97 - 122
NEXT
```

& 3.17 ASCAN()

! รูปแบบ

ASCAN(<ชุดอาเรย์เป้าหมาย>, <ข้อความที่ต้องการหา>,
[<เริ่มที่ตำแหน่ง>], [<จำนวนตำแหน่ง>]) --> เลขแสดงตำแหน่งที่พบ

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ค้นหาค่าที่ต้องการในอาเรย์

: ตัวอย่างที่ 3.26

```
AR := {'LION','TIGER','STAR','COCONUT','KEYBOARD'}
? ASCAN(AR,'LION')               //    1
? AR[ASCAN(AR,'LION')]           //  LION
? ASCAN(AR,'LION')               //    0
```

& 3.18 ASIZE()

! รูปแบบ

ASIZE(<ชุดอาเรย์เป้าหมาย>, <เลขกำหนดจำนวนสมาชิก>) --> ชุดอาเรย์เป้าหมาย

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ สั่งกำหนดขนาดของชุดอาเรย์ ซึ่งเพิ่มหรือลดจากเดิมได้

: ตัวอย่างที่ 3.27

```
AR  := {'A',1}
CNT := 0
AEVAL(AR,{|ELEMENT| CNT:=CNT+1})  ; ? CNT    // 2
ASIZE(AR,4)                       // A  1  NIL  NIL
CNT := 0
AEVAL(AR,{|ELEMENT| CNT:=CNT+1})  ; ? CNT          // 4
ASIZE(AR,3)                       // A  1  NIL
CNT := 0
```

```
AEVAL(AR,{|ELEMENT| CNT:=CNT+1}) ; ? CNT          // 3
ASIZE(AR,CNT-1)                  // A  1
CNT := 0
AEVAL(AR,{|ELEMENT| CNT:=CNT+1}) ; ? CNT          // 2
FOR I = 1 TO CNT
   ? AR[I]
NEXT
```

& 3.19 ASORT()

! รูปแบบ

ASORT(<ชุดอาเรย์เป้าหมาย>, [<เริ่มที่ตำแหน่ง>],
[<จำนวนตำแหน่ง>], [<ตรรกเลือกแบบในการจัดเรียง>]) --> ชุดอาเรย์เป้าหมาย

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ สั่งจัดเรียงสมาชิกในอาเรย์

: ตัวอย่างที่ 3.28

```
AR := {5,3,1,2,4}
X  := ACLONE(AR)
ASORT(AR)                        // จัดเรียงสมาชิกทุกตัวจากน้อยไปมาก
ASORT(AR,,,{|X,Y| X > Y})        // จัดเรียงสมาชิกทุกตัวจากมากไปน้อย
ASORT(AR,2,3)                    // จัดเรียงตัวที่ 2 ถึง 4 จากน้อยไปมาก
AEVAL(AR,{|ELE|QOUT(ELE)})       // 5 2 3 4 1
```

: ตัวอย่างที่ 3.29

```
AR := ARRAY(4,2)
AR[1,1] := 'DANG'  ; AR[1,2] := 12
AR[2,1] := 'BOY'   ; AR[2,2] := 24
AR[3,1] := 'JOJO'  ; AR[3,2] := 11
AR[4,1] := 'ANON'  ; AR[4,2] := 18
ASORT(AR)                        // จัดเรียงตามชื่อ จากน้อยไปมาก
ASORT(AR,,,{|X,Y|X[2]<Y[2]}) // จัดเรียงตามอายุจากน้อยไปมาก
AEVAL(AR,{|ELE|QOUT(ELE[1],ELE[2])})
ASORT(AR,,,{|X,Y|X[2]>Y[2]})     // จัดเรียงตามอายุจากมากไปน้อย
AEVAL(AR,{|ELE|QOUT(ELE[1],ELE[2])})     // นำอาเรย์มาพิมพ์
```

& 3.20 AT()

! รูปแบบ

AT(<ตัวอักษรที่ต้องการหา>, <ข้อความเป้าหมาย>) --> ตำแหน่งที่พบ

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ส่งตำแหน่งของข้อความย่อย ในข้อความที่ต้องการ

: ตัวอย่างที่ 3.30

```
? AT('A','IJUSTCALLTOSAYILOVEYOU')        // 7
TXT := 'MR.SILICON VLSI'
? SUBSTR(TXT,AT('.',TXT)+1)               // SILICON VLSI
? SUBSTR(TXT,1,AT(' ',TXT)-1)             // MR.SILICON
X := SUBSTR(TXT,1,AT(' ',TXT)-1)
? SUBSTR(X,AT('.',X)+1)                   // SILICON
```

& 3.21 ATAIL()

! รูปแบบ

ATAIL(<ชุดอาเรย์>) --> ค่าของสมาชิกตัวสุดท้าย

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ส่งค่าของสมาชิกอาเรย์ตัวสุดท้าย

: ตัวอย่างที่ 3.31

```
AR := {'C','A','F','D','B'}
? ATAIL(AR)                    // B
ASORT(AR)
? 'MAX : ',ATAIL(AR)           // F
? 'MIN : ',AR[1]               // A
```

& 3.22 BOF()

! รูปแบบ

BOF() --> ผลการตรวจสอบเรคอร์ดแรกของแฟ้ม

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ตรวจสอบค่าเป็นจริงเมื่อถึง RECORD แรกของแฟ้ม

: ตัวอย่างที่ 3.32

```
USE FILEA
? BOF() , RECNO()              // .F.    1
SKIP -1
? BOF() , RECNO()              // .T.    1
```

```
GO TOP
? BOF() , RECNO()                  // .F.      1
GO BOTTOM
DO WHILE .NOT. BOF()               //       104      800      700
  ? FIELD1,FIELD2,FIELD3           //       103      785      200
  SKIP -1                          //       102     6500      420
ENDDO                              //       101     1000      150
```

& 3.23 BREAK()

! รูปแบบ

BREAK() --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ หยุดการทำงาน

: ตัวอย่างที่ 3.33

```
USE FILEA
DO WHILE .NOT. EOF()
  BEGIN SEQUENCE
    ? FIELD1                       //       101    ONE RECORD ONLY
    BREAK                          // TO END OR RECOVER
    SKIP
  END
  BREAK // TO ENDDO OF DO WHILE
ENDDO
```

: ตัวอย่างที่ 3.34

```
USE FILEA
DO WHILE .NOT. EOF()
  ?? '%'                           // %
  BEGIN SEQUENCE                   // 101      *%
    IF FIELD1 = '103'              // 102      *%
      BREAK                        // WOW*%
    ENDIF                          // 104      *
    ? FIELD1
    SKIP                           // WHEN IT SKIP, IT WILL GOTO END
  RECOVER                          // IT COME FROM BREAK SO IT LOOK LIKE ELSE
```

```
      SKIP
      ? 'WOW'                    // IT WILL SKIP THIS LINE
    END
    ?? '*'
  ENDDO
```

& 3.24 BROWSE()

! รูปแบบ

BROWSE([<มุมบน>], [<มุมซ้าย>], [<มุมล่าง>], [<มุมขวา>]) --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ แสดงข้อมูลแบบ BROWSE ในหน้าต่างที่กำหนด

: ตัวอย่างที่ 3.35

```
USE FILEA
BROWSE(5,5,20,70)              // การเรียกดูข้อมูลจากแฟ้ม FILEA
SET FILTER TO FIELD2 >= 800
BROWSE(6,5,15,70)              // สามารถใช้ FILTER ร่วมกับ BROWSE ได้
```

& 3.25 CDOW()

! รูปแบบ

CDOW(<นิพจน์วันที่>) --> ชื่อวันในสัปดาห์

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ แปลงวันที่ เป็นวันในสัปดาห์

: ตัวอย่างที่ 3.36

```
? DATE()                       // 12/15/96
? DOW(DATE())                  //  2
? CDOW(DATE())                 // MONDAY
? CDOW(DATE()-3)               // SATURDAY
? DOW(DATE()-3)                //  7
? DOW(CTOD("06/04/69"))        //  4
X = "06/30/97"
? CDOW(CTOD(X))                // MONDAY
```

& 3.26 CHR()

! รูปแบบ

CHR(<จำนวนเลข>) --> ตัวอักษรในตารางแอสกี (ASCII)

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ แปลงรหัส ASCII เป็นตัวอักษร

: ตัวอย่างที่ 3.37

```
? CHR(65)                       // A
? CHR(7)                        // SOUND BEEP
? CHR(65) + CHR(66)             // AB
? CHR(ASC("A")+25)              // Z
? "*"                           // *ABCDEFGHIJKLMNOPQRSTUVWXYZ
FOR I = 0 TO 25
   ?? CHR(ASC("A")+I)
NEXT
```

& 3.27 CMONTH()

! รูปแบบ

CMONTH(<นิพจน์วันที่>) --> ชื่อเดือนภาษาอังกฤษ

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ แปลงวันที่เป็นชื่อเดือน

: ตัวอย่างที่ 3.38

```
? DATE()                              // 03/09/96
? CMONTH(DATE())+STR(YEAR(DATE()))    // MARCH 1996
? CMONTH(DATE()-2)                    // MARCH
? CMONTH(DATE()-10)                   // FEBRUARY
? CMONTH(DATE()-250)                  // JULY
? CMONTH(CTOD("06/04/69"))            // JUNE
X = "06/30/97"
? CMONTH(CTOD(X))                     // JUNE
? SUBSTR(CMONTH(DATE()),1,3)          // MAR
```

& 3.28 COL()

! รูปแบบ

COL() --> ตำแหน่งหลัก

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ส่งค่าตำแหน่งหลักปัจจุบัน ไปยังตัวแปร

: ตัวอย่างที่ 3.39

```
? COL()                                    //     0 CURRENT COLUMN =     30    20
_COL = COL()
_COL = _COL + 10
@ ROW() , COL()+1 SAY "CURRENT COLUMN = "
@ ROW() , COL()+2 SAY COL()
@ ROW() , COL()+2 SAY _COL
```

& 3.29 CTOD()

! รูปแบบ

CTOD(<นิพจน์อักขระ>) --> วันที่

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ แปลงตัวอักษรเป็นวันที่

: ตัวอย่างที่ 3.40

```
X = CTOD("12-31-2002")
Y = CTOD("12/31/2002")
? X,Y                                      // 12/31/02 12/31/02
SET CENTURY ON
? X,Y                                      // 12/31/2002  12/31/2002
GDATE = DATE()
@ ROW() , COL() SAY "GET DATE :" GET GDATE ;
          RANGE Y , DATE()
READ
```

& 3.30 CURDIR()

! รูปแบบ

CURDIR([<ชื่อไดรฟ์>]) --> ชื่อไดเร็คทรอรี่

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ส่งค่า DIRECTORY ปัจจุบัน

: ตัวอย่างที่ 3.41

```
? CURDIR("C")                  // LANGUAGE\CLIPPER5
? CURDIR("C:")                 // LANGUAGE\CLIPPER5
? CURDIR("CAT")                // LANGUAGE\CLIPPER5
```

```
RUN CD\WINDOWS
? CURDIR("C")                    // WINDOWS
RUN DIR
RUN CD\
? CURDIR("C")+"*"                // *
RUN CD\LANGUAGE\CLIPPER5
? CURDIR("A")                    // DATA
```

& 3.31 DATE()

! รูปแบบ

DATE() --> วันที่ของเครื่อง

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ส่งค่าวันที่ปัจจุบันของเครื่อง

: ตัวอย่างที่ 3.42

```
? DATE()                         // 12/31/02
SET CENTURY ON
? DATE()                         // 12/31/2002
```

& 3.32 DAY()

! รูปแบบ

DAY(<นิพจน์วันที่>) --> วันที่ของเดือน

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ส่งค่าวันที่ของเดือน

: ตัวอย่างที่ 3.43

```
? DATE()                         // 12/31/02
? DAY(DATE())                    // 31
? DAY(CTOD(""))                  // 0
? DAY(DATE())+1                  // 1
```

& 3.33 DBAPPEND()

! รูปแบบ

DBAPPEND() --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ สั่งเพิ่มเรคอร์ดใหม่ (เหมือน APPEND BLANK)

: ตัวอย่างที่ 3.44

```
USE FILEA
IF RECCOUNT() > 20
  DBAPPEND()
ELSE
  APPEND BLANK
ENDIF
```

& 3.34 DBCLEARFIL()

! รูปแบบ

DBCLEARFIL() --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ สั่งยกเลิกเงื่อนไขปัจจุบันของฟิวเตอร์ (เหมือน SET FILTER TO)

: ตัวอย่างที่ 3.45

```
USE FILEA
SET FILTER TO FIELD2 >= 800
COUNT TO X
? RECCOUNT(),X                    //    4     3
IF RECCOUNT() < 20
  SET FILTER TO
ELSE
  DBCLEARFIL()
ENDIF
COUNT TO X
? RECCOUNT(),X                    //    4     4
```

& 3.35 DBCLEARIND()

! รูปแบบ

DBCLEARIND() --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ สั่งยกเลิกการใช้แฟ้มดรรชนี INDEX (เหมือน SET INDEX TO)

: ตัวอย่างที่ 3.46

```
USE FILEA
INDEX ON FIELD1 TO IFIELD1
INDEX ON FIELD2 TO IFIELD2
USE FILEA INDEX IFIELD1,IFIELD2
```

```
LIST FIELD1,FIELD2,FIELD3
INKEY(0)
SET ORDER TO 2
LIST FIELD1,FIELD2,FIELD3
INKEY(0)
SET INDEX TO                    // ยกเลิกแฟ้มแบบ INDEX
LIST FIELD1,FIELD2,FIELD3
INKEY(0)
SET ORDER TO 2                  // ไม่มีผลอีก เพราะ INDEX ถูกยกเลิกไป
LIST FIELD1,FIELD2,FIELD3
INKEY(0)
DBCLEARIND()                    // ยกเลิกแฟ้มแบบ INDEX
LIST FIELD1,FIELD2,FIELD3
```

& 3.36 DBCLEARREL()

! รูปแบบ

DBCLEARREL() --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ สั่งยกเลิก RELATIONS ปัจจุบัน (เหมือน SET RELATION TO)

: ตัวอย่างที่ 3.47

```
USE FILEN INDEX IDN             // ID NAME
USE FILES INDEX IDS NEW         // ID SALARY
SET RELATION TO ID INTO FILEN
LIST  ID , NAME , SALARY
DBCLEARREL()
LIST  ID , SALARY
```

& 3.37 DBCLOSEALL()

! รูปแบบ

DBCLOSEALL() --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ สั่งปิดพื้นที่ทำงานทั้งหมด (เหมือน CLOSE ALL)

: ตัวอย่างที่ 3.48

```
USE FILEA
USE FILEB NEW
LIST FIELD1,FIELD2,FIELD3
SELE 1
LIST FIELD1,FIELD2,FIELD3
DBCLOSEALL()
```

& 3.38 DBCLOSEAREA()

! รูปแบบ

DBCLOSEAREA() --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ สั่งปิดพื้นที่ทำงานปัจจุบัน (เหมือน CLOSE หรือ USE)

: ตัวอย่างที่ 3.49

```
USE FILEA
USE FILEB NEW
LIST FIELD1
DBCLOSEAREA()
SELE 1
LIST FIELD1
DBCLOSEAREA()
```

& 3.39 DBCOMMIT()

! รูปแบบ

DBCOMMIT() --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ สั่งให้เขียนแฟ้มลงสื่อ มีผลอย่างมากต่อระบบเครือข่าย (เหมือน COMMIT)

: ตัวอย่างที่ 3.50

```
USE FILEA
APPEND BLANK
REPLACE FIELD1 WITH '105',FIELD2 WITH 1250,FIELD3 WITH 100
DBCOMMIT()
```

& 3.40 DBCOMMITALL()

! รูปแบบ

DBCOMMITALL() --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ สั่งให้เขียนแฟ้มในทุกพื้นที่ลงสื่อ มีผลอย่างมากต่อระบบเครือข่าย

: ตัวอย่างที่ 3.51

```
USE FILEA
APPEND BLANK
REPLACE FIELD1 WITH '105',FIELD2 WITH 1250,FIELD3 WITH 100
USE FILEB NEW
APPEND BLANK
REPLACE FIELD1 WITH '105',FIELD2 WITH 1250,FIELD3 WITH 100
DBCOMMITALL()
```

& 3.41 DBCREATE()

! รูปแบบ

DBCREATE(<ชื่อแฟ้มใหม่>, <ชุดอาเรย์เก็บโครงสร้าง>) --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ สั่งสร้างแฟ้มจากโครงสร้างอาเรย์ที่มีอยู่

: ตัวอย่างที่ 3.52

```
FIELDAR := ARRAY(3,4)                  // เตรียมให้แฟ้มนี้มี 3 ฟิลด์
FIELDAR[1] := {'FIELD1','C',3,0}
FIELDAR[2] := {'FIELD2','C',3,0}   ;    FIELDAR[3] := {'FIELD3','C',3,0}
DBCREATE('TESTCRT',FIELDAR)
USE TESTCRT
FLDAR := ARRAY(AFIELDS())
AFIELDS(FLDAR)
AEVAL(FLDAR,{|X|QOUT(X)})
```

& 3.42 DBDELETE()

! รูปแบบ

DBDELETE() --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ สั่งลบเรคอร์ดปัจจุบัน (เหมือน DELETE)

: ตัวอย่างที่ 3.53

```
USE FILEA
DBDELETE()
PACK
```

& 3.43 DBEDIT()

! รูปแบบ

DBEDIT([<มุมบน>], [<มุมซ้าย>], [<มุมล่าง>], [<มุมขวา>],
[<ชุดอาเรย์เก็บชื่อฟิลด์ที่ต้องการแสดง>]) --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ สั่งแสดงเรคอร์ดที่ต้องการ

: ตัวอย่างที่ 3.54

```
USE FILEA
AR := {'FIELD1','FIELD2'}
DBEDIT(5,5,20,70,AR)              // แก้ไขไม่ได้
BROWSE(4,5,15,70)                 // เพิ่มลบ หรือแก้ไขได้
```

& 3.44 DBF()

! รูปแบบ

DBF() --> สมนาม

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ส่งชื่อแฝงของพื้นที่ทำงานปัจจุบัน

: ตัวอย่างที่ 3.55

```
USE FILEA ALIAS NAMEFA
? DBF()                           // NAMEFA
USE FILEA
? DBF()                           // FILEA
```

& 3.45 DBFILTER()

! รูปแบบ

DBFILTER() --> ข้อความแสดงเงื่อนไขของตัวกรอง (FILTER)

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ส่งเงื่อนไขของ FILTER

: ตัวอย่างที่ 3.56

```
USE FILEA
FIL := 'FIELD2 >= 800'
SET FILTER TO &FIL
? DBFILTER()                          // FIELD2 > = 800
```

& 3.46 DBGOBOTTOM()

! รูปแบบ

DBGOBOTTOM() --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ไปยังเรคอร์ดสุดท้าย (เหมือน GO BOTTOM)

: ตัวอย่างที่ 3.57

```
USE FILEA
DBGOBOTTOM()
WHILE !BOF()
  ? FIELD1,FIELD2,FIELD3
  SKIP -1
END
```

& 3.47 DBGOTO()

! รูปแบบ

DBGOTO(<เลขที่เรคอร์ดที่ต้องการไปชี้>) --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ไปยังเรคอร์ดที่ต้องการ (เหมือน GO หรือ GOTO)

: ตัวอย่างที่ 3.58

```
USE FILEA
FOR I = 1 TO RECCOUNT()
   DBGOTO(I)  ;        ? FIELD1,FIELD2,FIELD3
NEXT
```

& 3.48 DBGOTOP()

! รูปแบบ

DBGOTOP() --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ไปยังเรคอร์ดแรก (เหมือน GO TOP)

: ตัวอย่างที่ 3.59

```
USE FILEA
DBGOTO(3)
? FIELD1  ;      DBGOTOP()
WHILE (!EOF())
  ? FIELD1,FIELD2,FIELD3
  DBSKIP()
END
```

& 3.49 DBRECALL()

! รูปแบบ

DBRECALL() --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ยกเลิกการทำเครื่องหมายลบทั้งหมด (เหมือน RECALL)

: ตัวอย่างที่ 3.60

```
USE FILEA
DELETE
SKIP
DELETE
DBRECALL()
```

& 3.50 DBREINDEX()

! รูปแบบ

DBREINDEX() --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ สั่งปรับปรุงแฟ้มดรรชนีให้ถูกต้อง (เหมือน REINDEX)

: ตัวอย่างที่ 3.61

```
USE FILEA INDEX IFIELD1,IFIELD2
DBREINDEX()
```

& 3.51 DBRELATION()

! รูปแบบ

DBRELATION(<เลขที่แฟ้ม>) --> ชื่อฟิลด์ของแฟ้มที่ระบุ

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ แสดงชื่อตัวแปรที่ใช้เชื่อมแฟ้มให้สัมพันธ์กัน

: ตัวอย่างที่ 3.62

```
USE FILECUS  INDEX ICUS              // ICUS  NAMEC
USE FILEEMP  INDEX IEMP NEW          // IEMP  NAMEE
USE FILESALE INDEX ISAL NEW          // ISAL  ICUS IEMP PRODUCT
SET RELATION ICUS INTO FILECUS,IEMP INTO FILEEMP
? DBRELATION(1)                      // ICUS
? DBRELATION(2)                      // IEMP
```

& 3.52 DBRSELECT()

! รูปแบบ

DBRSELECT(<เลขที่แฟ้ม>) --> เลขที่พื้นที่ทำงาน

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ส่งค่าตัวเลขของพื้นที่ ที่มีการใช้ฟิลด์ไปเชื่อมความสัมพันธ์มา

: ตัวอย่างที่ 3.63

```
USE FILECUS  INDEX ICUS              // ICUS  NAMEC
USE FILEEMP  INDEX IEMP NEW          // IEMP  NAMEE
USE FILESALE INDEX ISAL NEW          // ISAL  ICUS IEMP PRODUCT
SET RELATION ICUS INTO FILECUS,IEMP INTO FILEEMP
? DBRSELECT(1) ,DBRELATION(1)    // 1    ICUS
? DBRSELECT(2) ,DBRELATION(2)    // 2    IEMP
? ALIAS(DBRSELECT(1))            // FILECUS
? ALIAS(DBRSELECT(2))            // FILEEMP
```

& 3.53 DBSEEK()

! รูปแบบ

DBSEEK(<ข้อความที่ต้องการค้นหา>) --> จริงหรือเท็จ

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ค้นหาข้อมูลตามฟิลด์ที่ถูกจัดเรียงในแฟ้มดรรชนี

: ตัวอย่างที่ 3.64

```
USE FILEA INDEX IFIELD1
ACCEPT TO INDF1
IF DBSEEK(INDF1)
  ? 'FOUND' ,FIELD1,FIELD2,FIELD3
ELSE
```

```
    ? 'NOT FOUND'
  ENDIF
```

& 3.54 DBSELECTAR()

! รูปแบบ

DBSELECTAR(<ชื่อพื้นที่> | <สมนาม>) --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ เลือกพื้นที่เป้าหมายตามต้องการ

: ตัวอย่างที่ 3.65

```
USE FILEA
USE FILEB NEW
DBSELECTAR("FILEA")
? ALIAS() , SELECT()                    // FILEA      1
```

& 3.55 DBSETFILTER()

! รูปแบบ

DBSETFILTER(<เงื่อนไขการสร้างตัวกรอง>) --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ระบุเงื่อนไขในการเลือกข้อมูล (เหมือน SET FILTER TO)

: ตัวอย่างที่ 3.66

```
USE FILEA
DBSETFILTER({||FIELD2>=800})
? DBFILTER()                            // ไม่มีค่าให้แสดง
LIST FIELD1,FIELD2,FIELD3
INKEY(0)
DBSETFILTER({||FIELD2>=800},"FIELD2>=800")
? DBFILTER()                            // FIELD2>=800
LIST FIELD1,FIELD2,FIELD3
```

& 3.56 DBSETINDEX()

! รูปแบบ

DBSETINDEX(<ชื่อแฟ้มดรรชนี>) --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ สั่งเปิดแฟ้มดรรชนี เพื่อให้ข้อมูลจัดเรียงตามที่ต้องการ (เหมือน SET INDEX TO)

: ตัวอย่างที่ 3.67

```
USE FILEA
LIST FIELD1,FIELD2,FIELD3          // แสดงข้อมูลโดยไม่ใช้แฟ้มดรรชนี
DBSETINDEX("IFIELD2")
LIST FIELD1,FIELD2,FIELD3          // จัดเรียงตามแฟ้มดรรชนี IFIELD2
```

& 3.57 DBSETORDER()

! รูปแบบ

DBSETORDER(<เลขลำดับของแฟ้มดรรชนีที่เลือก>) --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ เลือกแฟ้มดรรชนีที่เคยเปิดไว้หลายแฟ้ม (เหมือน SET ORDER TO)

: ตัวอย่างที่ 3.68

```
USE FILEA INDEX IFIELD2,IFIELD3
LIST FIELD1,FIELD2,FIELD3          // จัดเรียงตามแฟ้มดรรชนี IFIELD2
DBSETORDER(2)
LIST FIELD1,FIELD2,FIELD3          // จัดเรียงตามแฟ้มดรรชนี IFIELD3
```

& 3.58 DBSETRELAT()

! รูปแบบ

DBSETRELAT(<เลขที่พื้นที่> | <สมนาม>, <เงื่อนไขการสร้างความสัมพันธ์>) --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ สั่งเชื่อมพื้นที่ทำงาน 2 พื้นที่ให้สัมพันธ์กัน (เหมือน SET RELATION TO)

: ตัวอย่างที่ 3.69

```
USE FILECUS INDEX ICUS
USE FILEEMP INDEX IEMP NEW
DBSETRELAT("FILECUS",{||FILEEMP->IEMP},"FILEEMP->IEMP")
LIST  IEMP , ICUS , INAMEC , INAMEE
```

& 3.59 DBSKIP()

! รูปแบบ

DBSKIP([<จำนวนเรคอร์ด>]) --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ย้ายตัวชี้เรคอร์ดไปยังเรคอร์ดที่ต้องการ (เหมือน SKIP)

: ตัวอย่างที่ 3.70

```
USE FILEA
DBSKIP()
DISPLAY FIELD1,FIELD2,FIELD3        // 2  102   6500   420
DBSKIP(2)
DISPLAY FIELD1,FIELD2,FIELD3        // 4  104    800   700
DBSKIP(-1)
DISPLAY FIELD1,FIELD2,FIELD3        // 3  103    785   200
DBSKIP(-1)
DISPLAY FIELD1,FIELD2,FIELD3        // 2  102   6500   420
DBSKIP(-1)
DISPLAY FIELD1,FIELD2,FIELD3        // 1  101   1000   150
```

& 3.60 DBSTRUCT()

! รูปแบบ

DBSTRUCT() --> ชุดอาเรย์เก็บโครงสร้างแฟ้ม

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ สร้าง ARRAY ที่เก็บโครงสร้างฐานข้อมูล

: ตัวอย่างที่ 3.71

```
USE FILEA
STRUC := DBSTRUCT()
AEVAL(STRUC,{|ELE| QOUT(ELE[1],ELE[2],ELE[3],ELE[4])})
INKEY(0)
AEVAL(STRUC,{|ELE| QOUT(ELE[1]) ,QQOUT(ELE[2]),;
            QQOUT(ELE[3]) ,QQOUT(ELE[4])})
```

& 3.61 DBUNLOCK()

! รูปแบบ

DBUNLOCK() --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ยกเลิกการจองพื้นที่ทำงานปัจจุบัน (เหมือน UNLOCK)

: ตัวอย่างที่ 3.72

```
USE FILEA SHARED
? FLOCK()                           // .T.
```

```
? FIELD1                    // 101
DBUNLOCK()                  // DBUNLOCK() ต้องใช้ในพื้นที่ที่ LOCK
USE FILEA SHARED NEW
? FLOCK()                   // .T.
SELE 1
? FLOCK()                   // .F.
```

& 3.62 DBUNLOCKALL()

! รูปแบบ

DBUNLOCKALL() --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ยกเลิกการจองพื้นที่ทำงานในทุกพื้นที่ (เหมือน UNLOCK ALL)

: ตัวอย่างที่ 3.73

```
USE FILEA SHARED
 ? FLOCK()                  // .T.
USE FILEB SHARED NEW
 ? FLOCK()                  // .T.
 DBUNLOCKALL()
USE FILEA SHARED NEW
 ? FLOCK()                  // .T.
USE FILEB SHARED NEW
 ? FLOCK()                  // .T.
SELE 1
 ? FLOCK()                  // .F.
SELE 2
 ? FLOCK()                  // .F.
```

& 3.63 DELETED()

! รูปแบบ

DELETED() --> ผลการตรวจสอบสถานะของเรคอร์ด

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ บอกให้ทราบว่าเรคอร์ดปัจจุบัน ถูกทำเครื่องหมายให้ถูกลบหรือไม่

: ตัวอย่างที่ 3.74

```
USE FILEA
? DELETE()                    // .F.
DELETE
? DELETE()                    // .T.
DBRECALL()
? DELETE()                    // .F.
```

& 3.64 DESCEND()

! รูปแบบ

DESCEND(<นิพจน์>) --> ค่านิพจน์เดิม

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ สั่งเรียงจากมากไปน้อยในแฟ้มดรรชนี

: ตัวอย่างที่ 3.75

```
USE FILEA
INDEX ON DESCEND(FIELD3) TO IFIELD3
LIST FIELD1,FIELD2,FIELD3
SEEK DESCEND(200)
IF FOUND()
  DISPLAY FIELD1,FIELD2,FIELD3     // 3   103   785  200
ENDIF
```

& 3.65 DEVOUT()

! รูปแบบ

DEVOUT(<นิพจน์>, [<ข้อความระบุสี>]) --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ส่งค่าออกทางอุปกรณ์ปัจจุบัน

: ตัวอย่างที่ 3.76

```
USE FILEA
WHILE !EOF()
   DEVPOS(ROW()+1,10)
   DEVOUT(FIELD1+STR(FIELD2)+STR(FIELD3),"BR+/B")
   DBSKIP()
END
```

& 3.66 DEVPOS()

! รูปแบบ

DEVPOS(<แถวที่>, <หลักที่>) --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ย้ายตำแหน่งจากจุดหนึ่งไปอีกจุดหนึ่ง

: ตัวอย่างที่ 3.77

```
USE FILEA
I = 1
WHILE !EOF()
   DEVPOS(I,10)
   DEVOUT(STR(I)+FIELD1+STR(FIELD2)+STR(FIELD3),"GR+/B")
   DBSKIP()
   I++
END
```

: ตัวอย่างที่ 3.78

```
USE FILEA
SET DEVICE TO PRINTER                  // ไม่เห็นผลใด ๆ ทางจอภาพ
WHILE !EOF()
   DEVPOS(PROW()+1,10)
   DEVOUT(FIELD1+STR(FIELD2)+STR(FIELD3),"W+")
   DBSKIP()
END
EJECT
SET DEVICE TO SCREEN
```

& 3.67 DIRECTORY()

! รูปแบบ

DIRECTORY(<กลุ่มแฟ้ม>, [<ลักษณะเฉพาะ>]) --> ชุดอาเรย์เก็บลักษณะแฟ้ม

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ สร้างอาเรย์เก็บชื่อแฟ้ม วันที่ เวลา หรือขนาด

ตัวเลือกของลักษณะเฉพาะ (ATTRIBUTES)

H ชื่อแฟ้ม รวมทั้งที่เป็นแฟ้มซ่อน

S ชื่อแฟ้ม รวมทั้งที่เป็นแฟ้มระบบ

D ชื่อแฟ้ม และชื่อ DIRECTORY ทั้งหมด

V ชื่ออุปกรณ์เก็บข้อมูล ซึ่งเป็นชื่อ VOLUMN

ข้อมูลที่ได้มา จะแบ่งเป็น 5 ส่วนคือ

ชื่อแฟ้ม ขนาด วันที่ เวลา ประเภทแฟ้ม

: ตัวอย่างที่ 3.79

```
OFATTRD := DIRECTORY("C:\*.*","D")
AEVAL(OFATTRD,{|E|QOUT  (E[1],E[2],E[3],E[4],E[5])})
? '--------------------------'
```

& 3.68 DISKSPACE()

! รูปแบบ

DISKSPACE([<เลขไดรฟ์>]) --> ขนาดไดรฟ์

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ตรวจสอบเนื้อที่ที่เหลือในสื่อเก็บข้อมูล

: ตัวอย่างที่ 3.80

```
? DISKSPACE()                  // 81010122  CURRENT DRIVE
? DISKSPACE(3)                 // 81010122  DRIVE C
? DISKSPACE(1)                 //  1153555  DRIVE A
? DISKSPACE(2)                 //  1153555  DRIVE B ถ้ามี DRIVE เดียว
```

& 3.69 DISPBOX()

! รูปแบบ

DISPBOX(<มุมบน>, <มุมซ้าย>, <มุมล่าง>, <มุมขวา>,
[<แบบของเส้น>], [<สีขอบ>]) --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ แสดงกรอบสี่เหลี่ยมบนจอภาพ

ตัวเลขระบุประเภทกรอบ

1 เส้นเดี่ยว

2 เส้นคู่

: ตัวอย่างที่ 3.81

```
DISPBOX(5,10,10,30,1,"BG+/B")
DISPBOX(15,5,20,40,2,"BG+/B")
```

& 3.70 DISPOUT()

! รูปแบบ

DISPOUT(<นิพจน์>, [<สีของตัวอักษร>]) --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ แสดงสิ่งที่ต้องการทางจอภาพ โดยคำสั่ง SET DEVICE จะไม่มีผลกับคำสั่งนี้

: ตัวอย่างที่ 3.82

```
USE FILEA
I = 1
WHILE !EOF()
   SETPOS(I,10)
   DISPOUT(STR(I)+FIELD1+STR(FIELD2)+STR(FIELD3),"GR+/B")
   DBSKIP()  ;       I++
END
```

& 3.71 DOW()

! รูปแบบ

DOW(<นิพจน์วันที่>) --> วันของสัปดาห์

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ แปลงวันที่เป็นวันของสัปดาห์

: ตัวอย่างที่ 3.83

```
? DATE()                          // 12/15/96
? DOW(DATE())                     //  2  (MONDAY)
? DOW(DATE()-3)                   //  6
? DOW(CTOD("06/04/69"))           //  4
FOR I = DOW(DATE()) TO DOW(DATE())+2
   ? CDOW(DATE()+I)               //  TUESDAY WEDNESDAY THURSDAY
NEXT
```

& 3.72 DTOC()

! รูปแบบ

DTOC(<นิพจน์วันที่>) --> วันที่เป็นตัวอักษร

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ แปลงวันที่เป็นตัวอักษร DD/MM/YY

: ตัวอย่างที่ 3.84

```
? DTOC(DATE())                  // 03/05/96
X = DTOC(DATE())                // 03/01/96
Y = VAL(SUBSTR(X,4,2))          // 03/02/96
FOR I = 1 TO Y                  // 03/03/96
   ? DATE()-Y+I                 // 03/04/96
NEXT                            // 03/05/96
```

& 3.73 DTOS()

! รูปแบบ

DTOS(<นิพจน์วันที่>) --> วันที่เป็นตัวอักษร

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ แปลงวันที่เป็นตัวอักษรในรูปแบบ YYYYMMDD

: ตัวอย่างที่ 3.85

```
? DTOS(DATE())                  // 19960309
X = DTOS(DATE())
? LEN(X)                        //        8
? "YEAR  :"+(SUBSTR(X,1,4))     // YEAR  :1996
? "MONTH :"+(SUBSTR(X,5,2))     // MONTH :03
? "DAY   :"+(SUBSTR(X,7,2))     // DAY   :09
```

& 3.74 EMPTY()

! รูปแบบ

EMPTY(<นิพจน์>) --> ผลการตรวจสอบค่าของนิพจน์

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ตรวจสอบค่าของตัวแปรว่ามีค่าหรือไม่

เงื่อนไขการตรวจสอบสำหรับประเภทของตัวแปร

```
ARRAY       : กว้างเป็น 0
CHARACTER   : ""
NUMERIC     : 0
DATE        : NULL(CTOD(""))
LOGICAL     : FALSE (.F.)
MEMO        : ""
NIL         : NIL
```

: ตัวอย่างที่ 3.86

```
A := {}
B := {""}
C := " "
D := ""
E := 0
F := .F.
? EMPTY(A)                    // .T.
IF EMPTY(A)
  ? EMPTY(B)                  // .F.
  ? EMPTY(C)                  // .T.
  ? EMPTY(D)                  // .T.
  ? EMPTY(E)                  // .T.
  ? EMPTY(F)                  // .T.
ENDIF
```

& 3.75 EOF()

! รูปแบบ

EOF() --> ผลการตรวจสอบเรคอร์ดสุดท้ายของแฟ้ม

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ถ้าหมดเรคอร์ดสุดท้ายจะส่งค่าจริงมาให้

: ตัวอย่างที่ 3.87

```
USE FILEA
? EOF() , RECNO()             // .F.     1
GO BOTTOM
? EOF() , RECNO()             // .F.     4
SKIP
? EOF() , RECNO()             // .T.     5
GO TOP
DO WHILE .NOT. EOF()          //     101   1000    150
  ? FIELD1,FIELD2,FIELD3      //     102   6500    420
  SKIP                        //     103    785    200
ENDDO                         //     104    800    700
```

& 3.76 ERRORLEVEL()

! รูปแบบ

ERRORLEVEL([<ค่าผิดพลาด>]) --> ค่าผิดพลาด

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ระบุค่าที่จะส่งกลับให้โปรแกรม

: ตัวอย่างที่ 3.88

```
IF !FILE("XX.PRG")
  ERRORLEVEL(25)              // TO SET VALUE OF ERRORLEVEL
  ERR = ERRORLEVEL()
ENDIF
IF ERRORLEVEL() = 25
  RUN DIR/S/OD/W
ENDIF
? ERR                         //  25
```

& 3.77 EVAL()

! รูปแบบ

EVAL(<นิพจน์คำนวณ>, [<รายการค่าต่าง ๆ ที่ป้อนให้นิพจน์>]) --> ค่าที่ได้จากนิพจน์

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ตรวจสอบโปรแกรม

: ตัวอย่างที่ 3.89

```
BCODE = {|A|A+1}
? EVAL(BCODE,5)               //  6
? EVAL(BCODE,2)               //  3
? EVAL(BCODE,7)               //  8
? (BCODE,5)                   //  5
? BCODE                       //  NOTHING
```

: ตัวอย่างที่ 3.90

```
BCODE = &("{|A|A*5-2}")
FOR I = 1 TO 3                //  3
  ? EVAL(BCODE,I)             //  8
NEXT                          //  13
```

: ตัวอย่างที่ 3.91

```
GRADE = {|G|IF(G>80,'A',(IF(G>70,'B','F')))}
? EVAL(GRADE,85)                    // A
? EVAL(GRADE,75)                    // B
? EVAL(GRADE,65)                    // F
```

& 3.78 EXP()

! รูปแบบ

EXP(<นิพจน์ตัวเลข>) --> ผลการยกกำลังของค่า E

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ สั่งคำนวณ E ยกกำลัง X

: ตัวอย่างที่ 3.92

```
? EXP(1)                            //   2.72
SET DECIMAL TO 10
? EXP(1)                            //   2.7182818285
? LOG(EXP(5))                       //   5.0000000000
? EXP(LOG(5))                       //   5.0000000000
```

& 3.79 FCLOSE()

! รูปแบบ

FCLOSE(<ชื่อตัวแปรระดับแฟ้ม>) --> ผลการปิดแฟ้ม

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ สั่งปิดแฟ้มที่เปิดแบบ BINARY แล้ว นำค่าใน BUFFER เขียนลง DISK

: ตัวอย่างที่ 3.93

```
F := FCREATE("TEST.TXT",0)          // เปิดแฟ้มแบบอ่านเขียนได้
FWRITE(F,"COMPUTER",10)             // แฟ้มจะมีขนาด 10 ไบต์
FCLOSE(F)
```

& 3.80 FCOUNT()

! รูปแบบ

FCOUNT() --> จำนวนฟิลด์ของแฟ้ม

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ จะแสดงจำนวณฟิลด์ของแฟ้ม

: ตัวอย่างที่ 3.94

```
USE FILEA
AFLDNAME := ARRAY(AFIELDS())
AFLDTYPE := ARRAY(AFIELDS())
AFLDWIDTH:= ARRAY(AFIELDS())
AFLDDEC  := ARRAY(AFIELDS())
AFIELDS(AFLDNAME,AFLDTYPE,AFLDWIDTH,AFLDDEC)
FOR I = 1 TO AFIELDS()              // FIELD1 C  10
   ?  AFLDNAME[I],AFLDTYPE[I]       // FIELD2 N  10  0
   ?? STR(AFLDWIDTH[I],10)          // FIELD3 N  10  0
   ?? IF(AFLDTYPE[I]='N',STR(AFLDDEC[I],10),'')
NEXT
? AFIELDS()                         //    3
? FCOUNT()                          //    3
```

& 3.81 FCREATE()

! รูปแบบ

FCREATE(<ชื่อแฟ้ม>, [<เลขระบุลักษณะเฉพาะ>]) --> ตัวแปรระดับแฟ้ม

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ สร้างแฟ้มแบบ BINARY FILE

การสร้างแฟ้มสามารถระบุแฟ้มได้หลายแบบดังนี้

0 อ่าน/เขียนได้

1 อ่านอย่างเดียว

2 แฟ้มแบบซ่อน

3 แฟ้มระบบ

: ตัวอย่างที่ 3.95

```
F := FCREATE("TEST.TXT",0)          // เปิดแฟ้มแบบอ่านเขียนได้
FWRITE(F,"AA",10)
FWRITE(F,"AA",10)
FWRITE(F,"AA",10)
FWRITE(F,"AA",10)
FCLOSE(F)                           // แฟ้มจะมีขนาด 40 ไบต์
? FERROR()                          // 0 แสดงว่าปิดแฟ้มได้ไม่ผิดพลาด
```

& 3.82 FERASE()

! รูปแบบ

FERASE(<ชื่อแฟ้ม>) --> ผลการลบแฟ้ม

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ สั่งลบแฟ้มจากสื่อเก็บข้อมูล

: ตัวอย่างที่ 3.96

```
IF FERASE("TEST.TXT") = -1
  ? "ERROR ON ERASING"
ENDIF
```

& 3.83 FERROR()

! รูปแบบ

FERROR() --> เลขระบุความผิดพลาด

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ตรวจสอบข้อผิดพลาดของการทำงานกับ BINARY FILE

: ตัวอย่างที่ 3.97

```
F := FCREATE("TEST.TXT",0)  // เปิดแฟ้มแบบอ่านเขียนได้
IF FERROR() = 0
  FWRITE(F,"AA",10)
  FWRITE(F,"AA",10)
  FWRITE(F,"AA",10)
  FWRITE(F,"AA",10)
  FCLOSE(F)                 // แฟ้มจะมีขนาด 40 ไบต์
  ? FERROR()                // 0 แสดงว่าปิดแฟ้มได้ไม่ผิดพลาด
ENDIF
```

& 3.84 FIELDNAME()

! รูปแบบ

FIELDNAME(<ตำแหน่งฟิลด์>) --> ชื่อฟิลด์

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ แสดงชื่อฟิลด์จากแฟ้มปัจจุบัน

: ตัวอย่างที่ 3.98

```
USE FILEA
? FIELDNAME(1)              // FIELD1 แสดงชื่อฟิลด์แรกของแฟ้ม
FOR I = 1 TO FCOUNT()
```

```
    ? FIELDNAME(I)
  NEXT
```

& 3.85 FIELDGET()

! รูปแบบ

FIELDGET(<ตำแหน่งฟิลด์>) --> ค่าของฟิลด์

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ รับค่าจากจากฟิลด์

: ตัวอย่างที่ 3.99

```
USE FILEA
? FIELD1                    // 101
? FIELD2                    //    1000
? FIELDGET(1)               // 101
? FIELDGET(2)               //    1000
X := FIELDNAME(1)
Y := &X
? Y                         // 101
```

& 3.86 FIELDPOS()

! รูปแบบ

FIELDPOS(<ชื่อฟิลด์>) --> ตำแหน่งฟิลด์

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ แสดงตำแหน่งของฟิลด์ปัจจุบัน

: ตัวอย่างที่ 3.100

```
USE FILEA
? FIELDPOS("FIELD1")        //    1
? FIELDPOS("FIELD2")        //    2
? FIELDGET(FIELDPOS("FIELD1"))  // 101
```

& 3.87 FIELDPUT()

! รูปแบบ

FIELDPUT(<ตำแหน่งฟิลด์>, <ค่าใหม่>) --> ค่าใหม่ที่แทนที่ค่าเก่าในฟิลด์

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ปรับปรุงค่าในฟิลด์

: ตัวอย่างที่ 3.101

```
USE FILEA
? FIELDGET(1)                 // 101
FIELDPUT(1,"103")             // ทำให้ค่าในฟิลด์เปลี่ยนไปโดยไม่ต้อง REPLACE
? FIELDGET(1)                 // 103
```

& 3.88 FILE()

! รูปแบบ

FILE(<ชื่อแฟ้ม>) --> ผลการตรวจสอบการเปิดแฟ้ม

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ตรวจสอบการเปิดแฟ้ม

: ตัวอย่างที่ 3.102

```
? FILE("COMMAND.COM")         // .F.
? FILE("\WINDOWS\WIN.COM")    // .T.
SET PATH TO \WINDOWS
? FILE("SYSTEM.INI")          // .T.
SET DEFAULT TO C:\DOS
? FILE("COMMAND.COM")         // .T.
SET PATH TO \WINDOWS
SET DEFAULT TO C:\DOS
? FILE("XCOPY.EXE")           // .T.
? FILE("WIN.INI")             // .T.
? CURDIR("C:")                // LANGUAGE\CLIPPER5
```