บทที่ Á

คำสั่ง (Command)

จุดประสงค์เชิงพฤติกรรม

1. เข้าใจพื้นฐานการคำสั่งใช้งาน
2. ใช้อ้างอิงเมื่อมีความต้องการ แต่นึกรูปแบบคำสั่งไม่ออก
3. ใช้อ้างอิงเมื่อต้องการหาคำสั่ง ไปแก้ปัญหาในการเขียนโปรแกรม

หัวข้อในบทเรียน

คำสั่งทั้งหมด 103 คำสั่ง

จาก คำสั่ง ? จนถึง คำสั่ง ZAP

ตัวอย่างโปรแกรมเกี่ยวกับคำสั่ง 223 ตัวอย่าง

หมายเหตุ

มีคำสั่งใช้งานมากมายที่ Clipper มีให้บริการ มีหลายคำสั่งตัวที่มีหน้าที่คล้ายฟังก์ชัน ก็เพื่อสร้างทางเลือกให้กับผู้พัฒนาสามารถเลือกใช้ได้โดยง่าย บทนี้ต้องการเป็นแหล่งอ้างอิง เพราะมีนักเรียน หรือนักพัฒนาน้อยคน ที่จะจำทั้งหมดได้ครบ หรือจำรูปแบบการใช้ได้ทั้งหมด .. แหล่งอ้างอิงจึงมีความสำคัญ

บทที่ 2 คำสั่ง(Command)

คำสั่งคือ ข้อความที่สั่งให้คอมพิวเตอร์ดกระทำการอย่างใดอย่างหนึ่ง เช่น สั่งให้พิมพ์ข้อความออกทางอุปกรณ์แสดงผล สั่งให้รอรับค่าจากแป้นพิมพ์ หรือสั่งให้ประมวลผลอย่างใดกับข้อมูลหรืออุปกรณ์เป็นต้น

& 2.1 ?|?? [8]

! โครงสร้างไวยากรณ์

?|?? [<รายการนิพจน์>]

P วัตถุประสงค์

คำสั่งนี้ ใช้พิมพ์ข้อความทางอุปกรณ์แสดงผล เช่นเครื่องพิมพ์ หรือจอภาพ

? ให้ผลเหมือนฟังก์ชัน QOUT คือเลื่อน 1 บรรทัดก่อนพิมพ์

?? ให้ผลเหมือนฟังก์ชัน QQOUT คือพิมพ์ต่อจากตำแหน่งปัจจุบัน โดยไม่เลื่อนบรรทัด

การเลื่อนบรรทัด หมายถึงการส่งรหัสแอสกี 13 และ 10 ไปที่อุปกรณ์แสดงผล

แอสกี 13(0D) หมายถึง การเลื่อนตัวกระพริบมาที่ต้นบรรทัด(CARRIAGE RETURN)

แอสกี 10(0A) หมายถึง การเลื่อนบรรทัดลง 1 บรรทัด(LINE FEED)

: ตัวอย่างที่ 2.1

```
USE FILEA
SET PRINT ON
SET CONSOLE OFF
WHILE .NOT. EOF()
  ? FIELD1,FIELD2,FIELD3
  SKIP
END
EJECT
SET CONSOLE ON
SET PRINT OFF
```

: ตัวอย่างที่ 2.2

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

---

[8] ดวง บงกชเกตุสกุล, อ้างแล้วในเชิงอรรถที่ (6), หน้าที่ 86

: ตัวอย่างที่ 2.3

```
USE FILEA
PG = 0 ;   I = 1
DO WHILE .NOT. EOF()
   IF MOD(I,20) = 1
     PG++
     ? 'HEADING   PAGE.'  ;    ?? PG  ;     ? '-------'
   ENDIF
   ?  I,FIELD1,FIELD2
   ?? FIELD3
   SKIP  ;      I++
ENDDO
```

& 2.2 @ ... BOX

! โครงสร้างไวยากรณ์

@ <มุมบน>, <มุมซ้าย>, <มุมล่าง>, <มุมขวา>,
BOX <ระบุแบบของกรอบ> [COLOR <สีของกรอบ>]

P วัตถุประสงค์

คำสั่งนี้ ใช้วาดสี่เหลี่ยมบนจอภาพ

: ตัวอย่างที่ 2.4

```
CLS                                          // 1223
@ 5,5,7,8 BOX '123456789'                    // 8994
INKEY(0)                                     // 7665
R1 = 10
C1 = 10
R2 = 13
C2 = 15
@ R1+1,C1+1,R2+1,C2+1 BOX '.........'        // ส่วนของเงา
@ R1,C1,R2,C2       BOX '+-+|+-+|*'   // ส่วนของพื้น
//  +----+          ผลลัพธ์ของการสร้างพื้นและเงาด้วยตัวอักษร
//  |****|.
//  |****|.
//  +----+.
//   ......
```

: ตัวอย่างที่ 2.5

```
#INCLUDE "BOX.CH"                    // B_DOUBLE_SINGLE ตัวเป็นตัวอักษรใหญ่
SETCOLOR("/B")                       // ระบุสีของพื้นทั้งหมดก่อนใช้คำสั่ง CLS
CLS
@ 6,12,21,32 BOX "" COLOR "N"
@ 5,10,20,30 BOX B_DOUBLE_SINGLE+" " COLOR "W/B"
```

& 2.3 @...CLEAR

! โครงสร้างไวยากรณ์

@ <มุมบน>, <มุมซ้าย> [CLEAR [TO <มุมล่าง>, <มุมขวา>]]

P วัตถุประสงค์

คำสั่งนี้ ใช้ลบหน้าจอตามตำแหน่งที่กำหนดได้ ซึ่งใช้ได้หลาย ๆ แบบ

1. ลบจากตำแหน่งที่กำหนดไปจนจบบรรทัด เช่น @ R1,C1
2. ลบจากตำแหน่งที่กำหนดไปถึงตำแหน่งที่ระบุ เช่น @ R1,C1 CLEAR TO R2,C2
3. ลบจากตำแหน่งที่กำหนดไปจนจบบรรทัด และลบบรรทัดลงมาจนสุดจอตามสดมภ์
   เช่น @ R1,C1 CLEAR

: ตัวอย่างที่ 2.6

```
@ 0,0,24,79 BOX "........."
@ 5,5 CLEAR TO 15,75                 // การลบพื้นที่ตามขอบเขตที่ระบุ
@ 20,10                              // ลบบรรทัดเดียว
@ 22,10 CLEAR                        // ลบจากตำแหน่งที่กำหนดลงไปจนหมดจอภาพ
INKEY(0)
```

& 2.4 @...GET

! โครงสร้างไวยากรณ์

@ <แถว>, <หลัก>
 [SAY <นิพจน์>
  [PICTURE <รูปแบบ>]
  [COLOR <ระบุสี>]]
 GET <ตัวแปร>
  [PICTURE <รูปแบบ>]
  [COLOR <ระบุสี>]
  [WHEN <เงื่อนไขก่อนใส่ข้อมูล>]
  [RANGE <ค่าต่ำสุด>, <ค่าสูงสุด>] | [VALID <เงื่อนไขหลังใส่ข้อมูล>]

P วัตถุประสงค์

คำสั่งนี้ รับค่าเข้าไปเก็บในฟิลด์ หรือตัวแปร ต้องใช้คู่กับ READ

ชื่อฟังก์ชัน	แบบของข้อมูล	คำอธิบายฟังก์ชันที่ใช้ควบคุมการรับค่า
A	C	รับเฉพาะตัวอักษรเท่านั้น โดยไม่รับตัวเลข
B	N	แสดงตัวเลขชิดซ้าย
C	N	แสดงตัวอักษร CR หลังตัวเลขค่าบวก
D	D,N	แสดงวันที่ตามรูปแบบที่เคยเซตไว้ในคำสั่ง SET DATE
X	N	แสดงตัวอักษร DB หลังตัวเลขค่าลบ
Z	N	แสดงช่องว่างแทนเลข 0 ที่ไม่มีค่า
(	N	แสดงวงเล็บล้อมรอบตัวเลขค่าลบ และไม่แสดงเครื่องหมายลบ
!	C	แสดงตัวอักษรทั้งหมดเป็นตัวใหญ่

: ตัวอย่างที่ 2.7

```
CLS
CH1 := "ABC"  ;CH2 := "DEF" ;DA := DATE()
NU1 := 15     ;NU2 := 25   ;NU3 := 35 ;NU4 := 45 ;NU5:=55
@ 5,5 SAY "TEST FUNCTION A : " GET CH1 PICT "@A"
READ
@ 6,5 SAY "TEST FUNCTION B : " GET NU1 PICT "@B"
READ
@ 7,5 SAY "TEST FUNCTION C : " GET NU2 PICT "@C"
@ 8,5 SAY "TEST FUNCTION D : " GET DA  PICT "@D"
READ
@ 9,5 SAY "TEST FUNCTION X : " GET NU3 PICT "@X"
@ 10,5 SAY "TEST FUNCTION Z : " GET NU4 PICT "@Z"
@ 11,5 SAY "TEST FUNCTION ( : " GET NU5 PICT "@("
@ 12,5 SAY "TEST FUNCTION ! : " GET CH2 PICT "@!"
READ
? CH1,NU1,NU2,DA,NU3,NU4,NU5,CH2
```

เปรียบเทียบตัวอย่างที่เขียนด้วย FOXPRO จะเห็นว่าแนวการเขียนเหมือนกัน

```
@  7, 10 SAY “WHAT’s YOUR  NAME? “  GET  NAME
```

[9]

[9] Granillo, Robert., Illustrated FOXPRO, Wordware Publishing, Inc., Texas, 1991, p.59

& 2.5 @...PROMPT

! โครงสร้างไวยากรณ์

@ <แถว>, <หลัก> PROMPT <ข้อความ> [MESSAGE <ข้อความ>]

P วัตถุประสงค์

คำสั่งนี้ สร้างเมนูแบบดึงขึ้นลง (PULLDOWN MENU)
เมนูแบบนี้สามารถกดตัวอักษรตัวแรกของตัวเลือก เพื่อเลือกตัวเลือกได้
การเลื่อนตัวเลือกไปจนบนสุด หากต้องการให้กลับมาล่างสุด หรือตัวเลือกอยู่
ที่ตัวเลือกสุดท้ายหากกดลูกศรลงแล้วให้อยู่ที่ตัวเลือกแรกต้อง SET WRAP ON

: ตัวอย่างที่ 2.8

```
@ 5,5 PROMPT "1. PRINT 1 TO 10"
@ 6,5 PROMPT "2. READ SALARY.DBF AND PRINT"
@ 7,5 PROMPT "3. QUIT"
MENU TO OPT
? OPT
```

& 2.6 @...SAY

! โครงสร้างไวยากรณ์

@ <แถว>, <หลัก> SAY <นิพจน์>
[PICTURE <รูปแบบ>] [COLOR <ระบุสี>]

P วัตถุประสงค์

คำสั่งนี้ ใช้แสดงค่าคงที่ ข้อความ หรือตัวแปรออกทางสื่อที่ต้องการ
ใช้คู่กับคำสั่ง GET ได้ เช่น @ 4,5 SAY "GET OPT " GET OPT ; READ

: ตัวอย่างที่ 2.9

```
CLS
@ 5,5 SAY 5  PICT "@C"
@ 6,5 SAY "ABC"+LTRIM(STR(5))+DTOC(DATE())
@ 7,5 SAY "COMPUTER" PICT "@!"
@ 8,5 SAY "GOPHER"   COLOR "B/W"
```

& 2.7 @...TO

! โครงสร้างไวยากรณ์

@ <มุมบน>, <มุมซ้าย> TO <มุมล่าง>, <มุมขวา>
[DOUBLE] [COLOR <ระบุสี>]

P วัตถุประสงค์

คำสั่งนี้ ใช้วาดสี่เหลี่ยมได้ 2 แบบ คือเส้นเดี่ยว และ เส้นคู่

: ตัวอย่างที่ 2.10

```
CLS
@ 5,5 TO 10,10                  // วาดเส้นเดี่ยว
@ 2,3 TO 12,30 DOUBLE           // วาดเส้นคู่
```

& 2.8 ACCEPT

! โครงสร้างไวยากรณ์

ACCEPT [<ข้อความ>] TO <ตัวแปร>

P วัตถุประสงค์

คำสั่งนี้ รับตัวอักษรเก็บในตัวแปร

ใช้ปุ่ม ESC ไม่ได้ และไม่สามารถส่งค่าปริยาย

: ตัวอย่างที่ 2.11

```
ACCEPT "TEST VAR1 : " TO VAR1
X = "AAA"
ACCEPT "TEST X : " TO X         // ไม่มีค่าของ X มารอให้แก้ไข
? VAR1,X
```

& 2.9 APPEND BLANK

! โครงสร้างไวยากรณ์

APPEND BLANK

P วัตถุประสงค์

คำสั่งนี้ เพิ่มเรคอร์ดใหม่ที่ว่างเปล่าเข้าไปในแฟ้ม

: ตัวอย่างที่ 2.12

```
USE FILEA
? RECCOUNT()                    // 4
APPEND BLANK
? RECCOUNT()                    // 5
DELETE ; PACK
? RECCOUNT()                    // 4
```

: ตัวอย่างที่ 2.13

```
USE FILEA
APPEND BLANK
```

```
@ ROW(),10 GET FIELD->FIELD1
@ ROW()+1,10 GET FIELD->FIELD2
@ ROW()+1,10 GET FIELD->FIELD3
READ
LIST FIELD1,FIELD2,FIELD3
GO BOTTOM
DELETE ; PACK
```

: ตัวอย่างที่ 2.14

```
USE FILEA
GO BOTTOM
X1 := LTRIM(STR(VAL(FIELD1)+1))
X2 := 0
X3 := 0
@ ROW(),10 GET X1
@ ROW()+1,10 GET X2
@ ROW()+1,10 GET X3
READ
APPEND BLANK
REPLACE FIELD1 WITH X1, FIELD2 WITH X2,FIELD3 WITH X3
LIST FIELD1,FIELD2,FIELD3
```

& 2.10 APPEND FROM

! โครงสร้างไวยากรณ์

APPEND FROM <ชื่อแฟ้ม>
[FIELDS <รายการชื่อฟิลด์>]
[<ช่วงที่ต้องการ>] [WHILE <เงื่อนไข] [FOR <เงื่อนไข>]
[SDF | DELIMITED [WITH BLANK | <ตัวอักษรที่ใช้แยก>]]

P วัตถุประสงค์

คำสั่งนี้ นำข้อมูลจากแฟ้มหนึ่งเพิ่มเข้าไปในแฟ้มที่มีอยู่

: ตัวอย่างที่ 2.15

```
USE FILEA
APPEND FROM FILEB
LIST FIELD1,FIELD2,FIELD3
```

& 2.11 AVERAGE

! โครงสร้างไวยากรณ์

AVERAGE <NEXP LIST> TO <IDVAR LIST>
[<ช่วงที่ต้องการ>] [WHILE <เงื่อนไข] [FOR <เงื่อนไข>]

P วัตถุประสงค์

คำสั่งนี้ หาค่าเฉลี่ยในฟิลด์ที่ต้องการ

: ตัวอย่างที่ 2.16

```
USE FILEA
AVERAGE FIELD1 TO X1
AVERAGE FIELD1 TO X2 FOR FIELD1 > 800
? X1, X2
```

& 2.12 CANCEL

! โครงสร้างไวยากรณ์

CANCEL | QUIT

P วัตถุประสงค์

คำสั่งนี้ ยกเลิกการทำงานของโปรแกรมทั้งหมด (ให้ผลเหมือน QUIT)

: ตัวอย่างที่ 2.17

```
FOR I = 1 TO 10
   FOR J = 1 TO 10
      ? J                          // ผลของโปรแกรมนี้คือพิมพ์ 1 ถึง 10
   NEXT
   CANCEL
NEXT
```

& 2.13 CLEAR ALL

! โครงสร้างไวยากรณ์

CLEAR ALL

P วัตถุประสงค์

คำสั่งนี้ ปิดแฟ้มทั้งหมด และยกเลิกตัวแปรแบบส่วนตัว (PRIVATE) และสาธารณะ (PUBLIC)
แต่ไม่ยกเลิกตัวแปรแบบ LOCAL และ STATIC

: ตัวอย่างที่ 2.18

```
LOCAL Z
USE FILEA
```

```
PRIVATE X
PUBLIC Y
X = 5
Z = 6
CLEAR ALL
? Z                                    // 6
```

& 2.14 CLEAR MEMORY

! โครงสร้างไวยากรณ์

CLEAR MEMORY

P วัตถุประสงค์

คำสั่งนี้ ยกเลิกตัวแปรแบบส่วนตัว (PRIVATE) และสาธารณะ (PUBLIC)
แต่ไม่ยกเลิกตัวแปรแบบ LOCAL และ STATIC

: ตัวอย่างที่ 2.19

```
LOCAL Z
USE FILEA
PRIVATE X
PUBLIC Y
X = 5
Z = 6
CLEAR MEMORY
LIST FIELD1,FIELD2,FIELD3
? Z
```

& 2.15 CLEAR SCREEN

! โครงสร้างไวยากรณ์

CLEAR [SCREEN] | CLS

P วัตถุประสงค์

คำสั่งนี้ ลบจอภาพ (ให้ผลเหมือน CLS)

: ตัวอย่างที่ 2.20

```
CLS
@ 5,5 TO 10,10
INKEY(0)
CLEAR
```

```
@ 6,5 TO 11,10 DOUBLE
CLEAR SCREEN
@ 7,5,12,10 BOX '/'
```

& 2.16 CLOSE

! โครงสร้างไวยากรณ์

CLOSE [<สมนาม> | ALL | ALTERNATE | DATABASES | FORMAT | INDEXES]

P วัตถุประสงค์

คำสั่งนี้ ปิดแฟ้มข้อมูล ในพื้นที่ทำงาน หรือแฟ้มดรรชนี

: ตัวอย่างที่ 2.21

```
USE FILEA
LIST FIELD1,FIELD2,FIELD3
USE FILEB NEW
LIST FIELD1,FIELD2,FIELD3
CLOSE
CLOSE ALL
```

& 2.17 COMMIT

! โครงสร้างไวยากรณ์

COMMIT

P วัตถุประสงค์

คำสั่งนี้ สั่งเก็บแฟ้มข้อมูลลงสื่อเก็บข้อมูล

: ตัวอย่างที่ 2.22

```
USE FILEA
X3 = 55
APPEND BLANK
REPLACE FIELD1 WITH '109'
REPLACE FIELD2 WITH 1000
REPLACE FIELD3 WITH X3
COMMIT
```

& 2.18 CONTINUE

! โครงสร้างไวยากรณ์

CONTINUE

P วัตถุประสงค์

คำสั่งนี้ สั่งค้นหาต่อเนื่อง ใช้ร่วมกับคำสั่ง LOCATE

: ตัวอย่างที่ 2.23

```
USE FILEA
LOCATE FOR FIELD2 >= 800
DO WHILE FOUND()
  ? FIELD1,FIELD2,FIELD3
  CONTINUE
ENDDO
```

: ตัวอย่างที่ 2.24

```
USE FILEA
LOCATE FOR FIELD2 >= 800
WHILE FOUND()
  ? FIELD1,FIELD2,FIELD3
  SKIP
  LOCATE REST FOR FIELD2 >=800
END
```

& 2.19 COPY FILE

! โครงสร้างไวยากรณ์

COPY FILE <ชื่อแฟ้มต้นแบบ> TO <ชื่อแฟ้มเป้าหมาย>

P วัตถุประสงค์

คำสั่งนี้ คัดลอกแฟ้มไปเก็บไว้อีกแฟ้มหนึ่ง

: ตัวอย่างที่ 2.25

```
? FILE("CL.BAT")                    // ถ้ามีแฟ้มนี้จะให้ผลเป็นจริง .T.
IF FILE("CL.BAT")
  COPY FILE "CL.BAT" TO "CL.BAK"
  ? FILE("CL.BAK")
  RUN DIR CL.BAK
ENDIF
```

& 2.20 COPY STRUCTURE

! โครงสร้างไวยากรณ์

COPY STRUCTURE [FIELDS <รายการชื่อฟิลด์>] TO <ชื่อแฟ้ม>

P วัตถุประสงค์

คำสั่งนี้ คัดลอกโครงสร้างแฟ้มข้อมูลของแฟ้มหนึ่งไปสร้างแฟ้มใหม่

: ตัวอย่างที่ 2.26

```
USE FILEA
COPY STRUCTURE TO FILEC
USE FILEC NEW
APPEND BLANK
REPLACE FIELD1 WITH "110"
REPLACE FIELD2 WITH 1200
REPLACE FIELD3 WITH 150
CLOSE
```

& 2.21 COPY STRUCTURE EXTENDED

! โครงสร้างไวยากรณ์

COPY STRUCTURE EXTENDED TO <ชื่อแฟ้ม>

P วัตถุประสงค์

คำสั่งนี้ คัดลอกโครงสร้างแฟ้มข้อมูลไปสร้างเป็นเรคอร์ดในแฟ้มใหม่

โดยแฟ้มใหม่จะประกอบด้วย 4 ฟิลด์ ซึ่งอธิบายเกี่ยวกับฟิลด์ของแฟ้มเดิม

: ตัวอย่างที่ 2.27

```
USE FILEA
COPY STRUCTURE EXTENDED TO TEMPFILE
USE TEMPFILE
LIST FIELD_NAME,FIELD_TYPE,FIELD_LEN,FIELD_DEC
```

& 2.22 COPY TO

! โครงสร้างไวยากรณ์

COPY [FIELDS <รายการชื่อฟิลด์>] TO <ชื่อแฟ้ม>
[<ช่วงที่ต้องการ>] [WHILE <เงื่อนไข] [FOR <เงื่อนไข>]
[SDF | DELIMITED [WITH BLANK | <ตัวอักษรที่ใช้แยก>]]

P วัตถุประสงค์

คำสั่งนี้ คัดลอกแฟ้มข้อมูลไปสร้างเป็นแฟ้มข้อมูลใหม่ หรือสร้างเป็นแฟ้มตัวอักษร

SDF หมายถึงสร้างเป็นแฟ้มอักษรที่ทุกฟิลด์มีขนาดเท่ากัน

DELIMITED หมายถึงสร้างแฟ้มข้อมูลที่แบ่งฟิลด์ด้วยตัวอักษร

: ตัวอย่างที่ 2.28

```
USE FILEA
COPY TO FILEC
COPY TO FILETXT1 SDF              // สร้างแฟ้มชื่อ FILETXT1.TXT
COPY TO FILETXT2 DELIMITED        // ใช้เครื่องหมายคำพูดแยกฟิลด์
```

& 2.23 COUNT

! โครงสร้างไวยากรณ์

```
COUNT TO <IDVAR>
 [<ช่วงที่ต้องการ>] [WHILE <เงื่อนไข] [FOR <เงื่อนไข>]
```

P วัตถุประสงค์

คำสั่งนี้ นับข้อมูลตามเงื่อนไขที่ต้องการ

: ตัวอย่างที่ 2.29

```
USE FILEA
COUNT TO X1
COUNT TO X2 FOR FIELD2 >= 800
COUNT TO X3 FOR FIELD2 >= 800 WHILE LEFT(FIELD1,1) = '1'
GO TOP  ;   COUNT TO X4 FOR FIELD2 >= 800 WHILE LEFT(FIELD1,1) = '1'
GO TOP  ;   COUNT TO X5 FOR FIELD2 >= 800 WHILE RIGHT(STR(FIELD3),1)='0'
? X1,X2,X3,X4,X5                  // 4   3   0  3  3
```

& 2.24 CREATE

! โครงสร้างไวยากรณ์

```
CREATE <ชื่อแฟ้ม>
```

P วัตถุประสงค์

คำสั่งนี้ สร้างแฟ้มขึ้นมา เพื่อเตรียมสร้างโครงสร้างให้อีกแฟ้มหนึ่ง

: ตัวอย่างที่ 2.30

```
CREATE TEMPFILE                   // มี 4 ฟิลด์ซึ่งไว้เก็บลักษณะฟิลด์
APPEND BLANK                      // คำต่อไปนี้เปลี่ยนไม่ได้
REPLACE FIELD_NAME WITH "ID",;    // FIELD_NAME
      FIELD_TYPE WITH "C",;       // FIELD_TYPE
      FIELD_LEN WITH 7,;          // FIELD_LEN
      FIELD_DEC WITH 0            // FIELD_DEC
CLOSE
```

& 2.25 CREATE FROM

! โครงสร้างไวยากรณ์

CREATE <ชื่อแฟ้มปลายทาง> FROM <ชื่อแฟ้มต้นแบบ>

P วัตถุประสงค์

คำสั่งนี้ สร้างแฟ้มใหม่โดยใช้โครงสร้างจากแฟ้มที่เคยสร้างด้วยคำสั่ง CREATE

: ตัวอย่างที่ 2.31

```
CREATE TEMPFILE
APPEND BLANK
REPLACE FIELD_NAME WITH "ID",;
        FIELD_TYPE     WITH "C",;
        FIELD_LEN      WITH 7,;
        FIELD_DEC      WITH 0
APPEND BLANK
REPLACE FIELD_NAME WITH "NAME",;
        FIELD_TYPE     WITH "C",;
        FIELD_LEN      WITH 20,;
        FIELD_DEC      WITH 0
CLOSE
CREATE NEWFILE FROM TEMPFILE
ERASE TEMPFILE.DBF
```

& 2.26 DELETE

! โครงสร้างไวยากรณ์

DELETE [<ช่วงที่ต้องการ>] [WHILE <เงื่อนไข>] [FOR <เงื่อนไข>]

P วัตถุประสงค์

คำสั่งนี้ ทำเครื่องหมายบนเรคอร์ดที่ต้องการลบออกจากแฟ้ม

: ตัวอย่างที่ 2.32

```
USE FILEA
DELETE ALL FOR EMPTY(FIELD1)
DELETE ALL FOR FIELD2 > 1000
PACK                                    // การลบเรคอร์ดออกจากแฟ้ม
```

& 2.27 DELETE FILE

! โครงสร้างไวยากรณ์

DELETE FILE | ERASE <ชื่อแฟ้ม>

P วัตถุประสงค์

คำสั่งนี้ ลบแฟ้มออกจากสื่อบันทึกข้อมูล (ให้ผลเหมือน ERASE)

: ตัวอย่างที่ 2.33

```
!DIR >TEST.TXT
? FILE("TEST.TXT")                  // .T.
DELETE FILE TEST.TXT
? FILE("TEST.TXT")                  // .F.
!DIR >TEST.TXT
? FILE("TEST.TXT")                  // .T.
ERASE TEST.TXT
? FILE("TEST.TXT")                  // .F.
```

& 2.28 DIR

! โครงสร้างไวยากรณ์

DIR [<ชื่อแฟ้ม>]

P วัตถุประสงค์

คำสั่งนี้ แสดงชื่อแฟ้มในระบบ พร้อมรายละเอียด

: ตัวอย่างที่ 2.34

```
DIR                                 // ดูชื่อแฟ้มที่นามสกุล DBF
INKEY(0)
DIR *.PRG                           // ดูชื่อแฟ้มที่นามสกุล PRG
INKEY(0)
X = "*.*"
DIR (X)                             // ต้องมีเครื่องหมายวงเล็บล้อมรอบตัวแปร
INKEY(0)
DIR X                               // คำสั่งนี้ใช้ไม่ได้ผล
DIR &X                              // & ทำหน้าที่ถอดเครื่องหมายคำพูด
INKEY(0)
```

& 2.29 DISPLAY

! โครงสร้างไวยากรณ์

DISPLAY <รายการนิพจน์>
[TO PRINTER] [TO FILE <ชื่อแฟ้ม>]
[<ช่วงที่ต้องการ>] [WHILE <เงื่อนไข] [FOR <เงื่อนไข>] [OFF]

P วัตถุประสงค์

คำสั่งนี้ แสดงฟิลด์ข้อมูลออกทางเครื่องพิมพ์ หรือแฟ้ม

: ตัวอย่างที่ 2.35

```
USE FILEA
DISPLAY DATE(),TIME(),RECCOUNT() OFF
WHILE .NOT. EOF()
  DISPLAY FIELD1,FIELD2,FIELD3
  SKIP
END
```

: ตัวอย่างที่ 2.36

```
USE FILEA
DISPLAY DATE(),TIME(),RECCOUNT() OFF TO PRINTER
DO WHILE .NOT. EOF()
  DISPLAY FIELD1,FIELD2,FIELD3 TO PRINTER
  SKIP
ENDDO
EJECT  // ผลลัพธ์จะออกทั้งจอภาพ และเครื่องพิมพ์
```

: ตัวอย่างที่ 2.37

```
USE FILEA
DISPLAY DATE(),TIME(),RECCOUNT() OFF
DISPLAY FIELD1,FIELD2,FIELD3 ALL TO FILE "TEST.TXT"
? "FINISH"                          // ผลลัพธ์จะออกทั้งจอภาพ และแฟ้ม
```

& 2.30 EJECT

! โครงสร้างไวยากรณ์

EJECT

P วัตถุประสงค์

คำสั่งนี้ สั่งเลื่อนการดาษออกจากเครื่องพิมพ์ 1 หน้า

: ตัวอย่างที่ 2.38

```
USE FILEA
LIST FIELD1,FIELD2,FIELD3 TO PRINTER
EJECT
DISPLAY ALL OFF FIELD1,FIELD2,FIELD3 TO PRINTER
EJECT
```

& 2.31 ERASE

! โครงสร้างไวยากรณ์

ERASE | DELETE FILE <ชื่อแฟ้ม>

P วัตถุประสงค์

คำสั่งนี้ ลบแฟ้มออกจากสื่อบันทึกข้อมูล (ให้ผลเหมือน DELETE FILE)

: ตัวอย่างที่ 2.39

```
USE FILEA
LIST FIELD1,FIELD2,FIELD3 TO FILE "TEST.TXT"
? FILE("TEST.TXT")                    // .T.
ERASE TEST.TXT
? FILE("TEST.TXT")                    // .F.
DISPLAY FIELD1,FIELD2 TO FILE "TEST.TXT" ALL
? FILE("TEST.TXT")                    // .T.
ERASE TEST.TXT
? FILE("TEST.TXT")                    // .F.
```

& 2.32 FIND

! โครงสร้างไวยากรณ์

FIND <ข้อความที่ต้องการหา>

P วัตถุประสงค์

คำสั่งนี้ ค้นหาจากข้อมูลที่ได้รับการจัดเรียงแบบดรรชนี

: ตัวอย่างที่ 2.40

```
USE FILEA INDEX IFIELD1
FIND "104"
? FOUND(),RECNO()                     // .T.  4
GO TOP
FIND 10
```

```
? FOUND(),RECNO()          // .T.  1
GO TOP
X := "103"
FIND X                     // ไม่สามารถใช้ตัวแปรในการค้นหาได้
? FOUND(),RECNO()          // .F.  5  แต่แฟ้มมีเพียง 4 เรคอร์ด
GO TOP
X := "103"
FIND (X)                   // ให้ใช้ () ล้อมรอบ X หรือใช้ &
? FOUND(),RECNO()          // .T.  3  แต่แฟ้มมีเพียง 4 เรคอร์ด
```

& 2.33 GO

! โครงสร้างไวยากรณ์

GO[TO] <เรคอร์ดที่ต้องการไปชี้> | BOTTOM | TOP

P วัตถุประสงค์

คำสั่งนี้ เลื่อนตัวชี้ไปชี้ที่เรคอร์ดที่ต้องการ

: ตัวอย่างที่ 2.41

```
USE FILEA
GO BOTTOM
? RECNO()                  // 4
GO TOP
? RECNO()                  // 1
GOTO BOTTOM
? RECNO()                  // 4
GOTO 3
? RECNO()                  // 3
GO 2
? RECNO()                  // 2
```

& 2.34 INDEX

! โครงสร้างไวยากรณ์

INDEX ON <ชื่อฟิลด์> TO <ชื่อแฟ้มดรรชนี> [UNIQUE]

P วัตถุประสงค์

คำสั่งนี้ สั่งสร้างแฟ้มดรรชนี

: ตัวอย่างที่ 2.42

```
USE FILEA
DBEDIT(5,5,10,40)
INDEX ON FIELD2 TO IFIELD2
LIST FIELD1,FIELD2,FIELD3
INDEX ON FIELD3 TO IFIELD3
DISPLAY ALL FIELD1,FIELD2,FIELD3
```

& 2.35 INPUT

! โครงสร้างไวยากรณ์

INPUT [<ข้อความ>] TO <ตัวแปร>

P วัตถุประสงค์

คำสั่งนี้ รับชื่อตัวแปรไปเก็บในตัวแปรตัวใหม่

: ตัวอย่างที่ 2.43

```
A = 2 ;      B = 3
INPUT "TEST : " TO X1          // ถ้าพิมพ์ A จะได้ X2 เป็น 1
? X1                           // ถ้าพิมพ์ B จะได้ X2 เป็น 1.5
X2 := X1 / 2                   // ถ้าพิมพ์ 6 จะได้ X2 เป็น 3
? X2
```

& 2.36 JOIN

! โครงสร้างไวยากรณ์

JOIN WITH <สมนามหรือชื่อแฟ้ม> TO <ชื่อแฟ้ม>
FOR <เงื่อนไข> [FIELDS <รายการชื่อฟิลด์>]

P วัตถุประสงค์

คำสั่งนี้ นำแฟ้มข้อมูล 2 แฟ้มมาเชื่อมกันด้วยเงื่อนไข แล้วสร้างแฟ้มข้อมูลใหม่

: ตัวอย่างที่ 2.44

```
USE FILEA
USE FNAME NEW
JOIN WITH FILEA TO FILEB FOR FLDNAME = FILEA->FIELD1;
   FIELDS FLDNAME,FILEA->FIELD2,FILEA->FIELD3,NAME
USE FILEB
LIST FLDNAME,FIELD2,FIELD3,NAME
CLEAR ALL
```

& 2.37 KEYBOARD

! โครงสร้างไวยากรณ์

KEYBOARD <ข้อความ>

P วัตถุประสงค์

คำสั่งนี้ ส่งค่าเข้าไปในแป้นพิมพ์

: ตัวอย่างที่ 2.45

```
KEYBOARD "A"+CHR(13)+CHR(10)+"5"+CHR(13)+CHR(10)
A = TIME()
INPUT TO X1
? X1                                    // 12:15:10
INPUT TO X2
? X2 * 2                                // 10
```

& 2.38 LIST

! โครงสร้างไวยากรณ์

LIST <รายการนิพจน์>
[TO PRINTER] [TO FILE <ชื่อแฟ้ม>]
[<ช่วงที่ต้องการ>] [WHILE <เงื่อนไข] [FOR <เงื่อนไข>] [OFF]

P วัตถุประสงค์

คำสั่งนี้ แสดงข้อมูลทุกเรคอร์ด หรือตามเงื่อนไข

: ตัวอย่างที่ 2.46

```
USE FILEA
LIST FIELD1,FIELD2 FOR FIELD2 >= 1000
INKEY(0)
GO TOP                                          // ต้องมีบรรทัดนี้ จึงจะใช้ WHILE ได้ถูกต้อง
LIST FIELD1,FIELD2 WHILE FIELD1 != "103"        // พิมพ์ 2 เรคอร์ดแรก
INKEY(0)
LIST FIELD1,FIELD2,FIELD3 WHILE INKEY() != 27
INKEY(0)
LIST FIELD1,FIELD2,FIELD3 TO PRINTER TO FILE TEST.TXT
INKEY(0)
```

& 2.39 LOCATE

! โครงสร้างไวยากรณ์

LOCATE [<ช่วงที่ต้องการ>] [WHILE <เงื่อนไข] [FOR <เงื่อนไข>]

P วัตถุประสงค์

คำสั่งนี้ การค้นหาข้อมูลจากแฟ้มข้อมูล

: ตัวอย่างที่ 2.47

```
USE FILEA
LOCATE FOR FIELD1 >= "102"
IF FOUND()
  ? FIELD1,FIELD2,FIELD3
  ? RECNO()
ELSE
  ? "NOT FOUND"
ENDIF
```

: ตัวอย่างที่ 2.48

```
USE FILEA
LOCATE FOR FIELD2 >= 1000
WHILE FOUND()
  ? FIELD1,FIELD2,FIELD3
  CONTINUE
END
```

: ตัวอย่างที่ 2.49

```
USE FILEA
CRITERIA = "FIELD2 >= 1000"
LOCATE FOR &CRITERIA              // ไม่สามารถใช้ CRITERIA
WHILE FOUND()
  ? FIELD1,FIELD2,FIELD3
  SKIP
  LOCATE REST FOR &CRITERIA
END
```

& 2.40 MENU TO

! โครงสร้างไวยากรณ์

MENU TO <ตัวแปร>

P วัตถุประสงค์

คำสั่งนี้ รับค่าตัวเลือกที่ได้จาก ชุดของตัวเลือกที่สร้างด้วยคำสั่ง PROMPT

: ตัวอย่างที่ 2.50

```
CLS
SET WRAP ON
SET MESSAGE TO 21 CENTER
@ 5,5 PROMPT "A. HOME " MESSAGE "CHOICE OF HOME"
@ 6,5 PROMPT "B. CAR  " MESSAGE "CHOICE OF CAR"
@ 7,5 PROMPT "C. EXIT " MESSAGE "OUT TO DOS"
MENU TO OPT
IF OPT <> 3
  DO X
ENDIF
? "OPTION IS ",OPT                    // ให้สังเกตเมื่อเรียก X หลายครั้ง
```

& 2.41 NOTE

! โครงสร้างไวยากรณ์

NOTE [<ข้อความ>]

P วัตถุประสงค์

คำสั่งนี้ ใช้ระบุเป็นหมายเหตุในโปรแกรม

: ตัวอย่างที่ 2.51

```
USE FILEA
LIST FIELD1,FIELD2,FIELD3
NOTE  LIST FIELD1,FIELD2,FIELD3    หมายเหตุแบบที่ 1
//    LIST FIELD1,FIELD2,FIELD3    หมายเหตุแบบที่ 2
/*    LIST FIELD1,FIELD2,FIELD3    หมายเหตุแบบที่ 3 */
*     LIST FIELD1,FIELD2,FIELD3    หมายเหตุแบบที่ 4
&&  LIST FIELD1,FIELD2,FIELD3      หมายเหตุแบบที่ 5
```

& 2.42 PACK

! โครงสร้างไวยากรณ์

PACK

P วัตถุประสงค์

คำสั่งนี้ ลบเรคอร์ดที่เคยถูกทำเครื่องหมายด้วยคำสั่ง DELETE ออกจากสื่อบันทึกข้อมูล

: ตัวอย่างที่ 2.52

```
USE FILEA
DELETE FOR FIELD2 >= 800
DELETE FOR FIELD2 >= 800
PACK
```

& 2.43 QUIT

! โครงสร้างไวยากรณ์

QUIT | CANCEL

P วัตถุประสงค์

คำสั่งนี้ ยกเลิกการทำงานของโปรแกรมทั้งหมด (ให้ผลเหมือน CANCEL)

: ตัวอย่างที่ 2.53

```
I = 0
DO WHILE .T.
  ? "*"                                  // ผลลัพธ์คือพิมพ์ * ทั้งหมด 4 บรรทัด
  WHILE I != 0
    I++
    IF I > 5
      QUIT
    ENDIF
    EXIT
  END
  I++
ENDDO
```