ภาคผนวก n

SEQUENTIAL AND RANDOM FILES

ไฟล์ข้อมูลของดิสเกตต์ (diskette data files) มีอยู่ 2 ชนิด ซึ่งไฟล์ข้อมูลทั้งสองชนิดนี้อาจจะถูกสร้างขึ้นมาหรือถูกเข้าถึงโดยโปรแกรมของ GWBASIC ไฟล์ข้อมูลเหล่านี้คือ :

ไฟล์ข้อมูลแบบเรียงลำดับ (sequential file) และไฟล์ข้อมูลแบบสุ่ม (random file)

ไฟล์ข้อมูลแบบเรียงลำดับ (Sequential Files)

ไฟล์ข้อมูลแบบเรียงลำดับนั้นสร้างขึ้นได้ง่ายกว่าไฟล์ข้อมูลแบบสุ่มแต่จะถูกจำกัดขอบเขตในเรื่องของความยืดหยุ่น(flexibility) และความเร็ว(speed) ทั้งนี้เนื่องจากการดึงข้อมูลออกมาใช้งานข้อมูล (อักขระต่าง ๆ ในรูปแอสกี) ที่ถูกบันทึกไปยังไฟล์แบบเรียงลำดับ (sequential file) ถูกเก็บ 1 item หลังจาก item อื่นตามลำดับที่มันถูกส่งไป และการอ่านข้อมูลมาปฏิบัติการก็ดำเนินในรูปแบบเดียวกัน

คำสั่งและฟังก์ชันต่าง ๆ ที่ถูกใช้กับไฟล์แบบเรียงลำดับต่าง ๆ คือ :

C L O S E	L O F
EOF	OPEN
INPUT #	PRINT #
INPUT$	PRINT # USING
LINE INPUT #	WRITE #
LOC	

<u>การสร้างและการเข้าถึงไฟล์แบบเรียงลำดับ</u> (Creating and Accessing a Sequential Files)

ขั้นตอนต่าง ๆ ของโปรแกรมต่อไปนี้ถูกต้องการในการที่จะสร้างไฟล์และเข้าถึงข้อมูลในไฟล์

1. ใช้ OPEN เพื่อที่จะเปิดไฟล์สำหรับ output หรือ append

2. บันทึกข้อมูลลงสู่ไฟล์โดยการใช้คำสั่ง WRITE #, PRINT # หรือคำสั่ง PRINT # USING

3. เพื่อที่จะเข้าถึงข้อมูลในไฟล์ ก่อนอื่นปิดไฟล์ (โดยการใช้ CLOSE) จากนั้นเปิดแฟ้มข้อมูลใหม่เพื่อใช้เป็น input (โดยการใช้ OPEN)

4. อ่านข้อมูลจากไฟล์ลงสู่โปรแกรมโดยการใช้คำสั่ง INPUT # หรือคำสั่ง LINE INPUT #

ต่อไปนี้เป็นโปรแกรมสั้น ๆ ที่สร้างไฟล์แบบเรียงลำดับชื่อ "DATA" ขึ้นมาจากข้อมูลที่ป้อน(input)เข้าไปยังจอภาพ

```
10 OPEN "O",#1 "DATA"
20 INPUT "NAME";N$
25 IF N$="DONE" THEN END
30 INPUT "DEPARTMENT";D$
40 INPUT "DATE HIRED";H$
50 PRINT #1,N$;",";D$;",";H$
60 PRINT:GOTO 20
RUN
NAME? MICKEY MOUSE
DEPARTMENT? AUDIO/VISUAL AIDS
DATE HIRED? 01/12/72

NAME? SHERLOCK HOLMES
DEPARTMENT? RESEARCH
DATE HIRED? 12/03/65
```

```
NAME? EBENEEZER SCROOGE
DEPARTMENT?    ACCOUNTING
DATE HIRED? 04/27/78

NAME? SUPER MANN
DEPARTMENT? MAINTENANCE
DATE HIRED? 08/16/78

NAME?  ect.
```

โปรแกรมต่อมาเข้าถึงไฟล์ที่ถูกสร้างขึ้นข้างบนและพิมพ์ (display) ชื่อของทุก ๆ คนที่ถูกจ้างในปีค.ศ.1978

```
10 OPEN "I",#1, "DATA"
20 INPUT #1,N$,D$,H$
30 IF RIGHT$(H$,2)="78" THEN PRINT N$
40 GOT0  20
Ok
RUN
EBENEEZER   SCROOGE
SUPERMANN
Input past end in 20
Ok
```

โปรแกรมข้างบนนี้อ่าน item ทุก ๆ item ในไฟล์เรียงตามลำดับ เมื่อข้อมูลทุก ๆ ข้อมูลถูกอ่าน ไลน์ 20 จะเป็นเหตุให้เกิด "Input past end" error ขึ้น เพื่อที่จะหลีกเลี่ยงการที่จะได้รับข้อผิดพลาด (error) นี้ ใส่ไลน์ 15 ซึ่งใช้ฟังก์ชัน EOF เพื่อที่จะตรวจสอบสำหรับจุดจบของไฟล์

(end-of-file) เข้าไป

```
15 IF EOF(1) THEN END
```

จากนั้นเปลี่ยนแปลงไลน์ 40 เป็น GOTO 15 เนื่องจากจุดจบของไฟล์ถูกบ่งชี้โดยอักขระพิเศษในไฟล์ซึ่งมีรหัสแอสกีเป็น 26 (hex 1A) ท่านไม่ควรใส่ CHR$(26) ในไฟล์แบบเรียงลำดับ โปรแกรมซึ่งสร้างไฟล์แบบเรียงลำดับนั้นสามารถจะบันทึกข้อมูลที่ถูกกำหนดรูปแบบ (formatted data) ไปยังดิสเกตต์โดยคำสั่ง PRINT # USING ได้ด้วย ตัวอย่างเช่น คำสั่ง

```
PRINT #1,USING "####.##,";A,B,C,D
```

ควรจะถูกใช้เพื่อที่จะบันทึกข้อมูลจำนวนเลขไปยังดิสเกตต์โดยปราศจากตัวจำกัดขอบเขตต่าง ๆ ภายนอก (explicit delemiters) เครื่องหมายจุลภาคที่ตอนท้ายของสตริงก์กำหนดรูปแบบสงวนไว้เพื่อที่จะแบ่งแยก items ในดิสเกตต์ไฟล์

สังเกตว่า ฟังก์ชัน LOC เมื่อถูกใช้กับไฟล์แบบเรียงลำดับ จะส่งจำนวนของเร็คคอร์ดซึ่งถูกบันทึกไปยังหรือถูกอ่านจากไฟล์ตั้งแต่เมื่อมันถูกเปิด (เร็คคอร์ดเป็นบล็อคของข้อมูลขนาด 128 ไบท์) ฟังก์ชัน LOC ส่งค่าเป็นจำนวนของไบท์ต่าง ๆ ที่ถูกจัดสรรไปยังไฟล์ ตัวเลขนี้โดยปกติเป็นพหุคูณของ 128 โดยปัดเศษขึ้นตามความเหมาะสม ตัวอย่างเช่น

```
100 IF LOC(1)>50 THEN STOP
```

จะสิ้นสุดการประมวลผลโปรแกรมถ้าเซกเตอร์ (sector) มากกว่า 50 เซกเตอร์ได้ถูกบันทึกไปยังหรือได้ถูกอ่านจากไฟล์หมายเลข #1 ตั้งแต่เปิดแฟ้มข้อมูลนั้น

<u>การเพิ่มข้อมูลไปยังไฟล์แบบเรียงลำดับ</u> (Adding Data to a Sequential File)

ถ้าท่านมีไฟล์แบบเรียงลำดับบันทึกอยู่บนดิสเกตต์และต้องการที่จะเพิ่มข้อมูลเข้าไปที่ตอนท้ายของไฟล์แล้วท่านไม่สามารถที่จะเปิดไฟล์สำหรับ output mode และเริ่มต้นทำการบันทึกข้อมูลได้โดยง่าย เนื่องจากว่าถ้าเราดำเนินการเช่นนี้แล้ว จะเป็นการทำลายเนื้อหาของไฟล์ (file's content) ซึ่งวิธีการที่จะป้องกันเรื่องนี้ให้ทำโดยการเปิดไฟล์สำหรับ APPEND แทนที่จะเปิดไฟล์สำหรับ output mode (ดูคำสั่ง OPEN สำหรับการเพิ่มข้อมูล)

ไฟล์ข้อมูลแบบสุ่ม (Random Files)

การสร้างและการดึงข้อมูลในแฟ้มแบบสุ่มมาปฏิบัติงานนั้นต้องอาศัยขั้นตอนต่าง ๆ ของโปรแกรมมากกว่าการสร้างและการเข้าถึงไฟล์แบบเรียงลำดับ แต่การใช้ไฟล์แบบสุ่มจะมีข้อดีคือ ไฟล์ต่าง ๆ แบบสุ่มต้องการเนื้อที่บนดิสค์น้อยกว่าเนื่องจาก GWBASIC เก็บข้อมูลในรูปแบบของ packed binary format (อย่าลืมว่าไฟล์แบบเรียงลำดับจะถูกเก็บในสถานะของเป็นชุดลำดับของอักขระแอสกีต่าง ๆ)

ข้อได้เปรียบที่สำคัญของไฟล์แบบสุ่มก็คือว่า ข้อมูลสามารถถูกดึงออกมาดำเนินงานอย่างสุ่ม นั่นคือ ณ ที่แห่งใดบนดิสเกตต์ เราไม่จำเป็นที่จะต้องอ่านไปตลอดทุกข้อมูลเหมือนกับไฟล์แบบเรียงลำดับ ทั้งนี้จากข่าวสารถูกเก็บและถูกดึงเร็คคอร์ดมาปฏิบัติงานโดยใช้ตำแหน่งที่ตั้งของเร็คคอร์ดในแฟ้มข้อมูลได้โดยอัตโนมัติ โดยที่ GWBASIC จะใช้ทุก ๆ 512 ไบท์ใน 1 เซกเตอร์สำหรับเนื้อที่เก็บข่าวสาร (information storage) โดยทั้ง blocking records และ spanning sector boundaries ดังนั้น ขนาดของเร็คคอร์ดจะไม่สัมพันธ์กับขนาดของเซกเตอร์ ดังนั้นส่วนหนึ่งของเร็คคอร์ดอาจจะอยู่ที่ตอนปลายของเซกเตอร์หนึ่งและส่วนอื่น ๆ อาจจะอยู่ที่จุดเริ่มต้นของเซกเตอร์ถัดไปก็ได้

คำสั่งและฟังก์ชันต่าง ๆ ที่ถูกใช้กับไฟล์แบบสุ่มคือ :

CLOSE	LOC
CVD	LOF
CVI	MKD$
CVS	MKI$
FIELD	MKS$
GET	OPEN
LSET/RSET	PUT

<u>การสร้างไฟล์แบบสุ่ม</u> (Creating a Random File)

ใช้ขั้นตอนของโปรแกรมต่าง ๆ เพื่อที่จะสร้างไฟล์แบบสุ่ม

1. เปิดไฟล์สำหรับการเข้าถึงแบบสุ่ม (random access)
2. ใช้คำสั่ง FIELD เพื่อที่จะจัดสรรที่ว่าง (space) ในบัฟเฟอร์แบบสุ่ม (random buffer) ให้กับตัวแปรต่าง ๆ ซึ่งจะถูกบันทึกไปยังไฟล์
3. ใช้คำสั่ง LSET หรือคำสั่ง RSET เพื่อที่จะย้ายข้อมูลไปยังบัฟเฟอร์แบบสุ่ม มูลค่าจำนวนเลขต่าง ๆ ต้องถูกแปลงค่าเป็นมูลค่าต่าง ๆ ในรูปสตริงก์เมื่อถูกแทนที่ในบัฟเฟอร์ โดยการใช้ฟังก์ชันต่าง ๆ ดังต่อไปนี้ : MKI$ สำหรับมูลค่าจำนวนเต็ม, MKS$ สำหรับมูลค่าตาง ๆ ในรป single-precision และ MKD$ สำหรับมูลค่าต่าง ๆ ในรูป double-precision
4. ใช้คำสั่ง PUT เพื่อที่จะบันทึกข้อมูลจากบัฟเฟอร์ไปยังดิสเกตต์

โปรแกรมต่อไปนี้บันทึกข่าวสารที่ถูกป้อนเข้าไปจากแป้นคีย์ไปยังไฟล์แบบสุ่ม แต่ละครั้งที่คำสั่ง PUT ถูกปฏิบัติ เร็คคอร์ดจะถูกบันทึกไปยังไฟล์ รหัสที่เป็นตัวเลข 2 หลักซึ่งเป็นข้อมูลเข้า (input) ในไลน์ 30 จะกลายเป็นหมายเลขเร็คคอร์ด (record number)

```
10 OPEN "R",#1,"FILE",32
20 FIELD #1,20 AS N$, 4 AS A$, 8 AS P$
30 INPUT "Z-DIGIT CODE";CODE%
40 INPUT "NAME";X$
50 INPUT "AMOUNT";AMT
60 INPUT "PHONE";TEL$:PRINT
70 LSET N$=X$
80 LSET A$=MKS$(AMT)
90 LSET P$=TEL$
100 PUT #1,CODE%
110 GOTO 30
```

อย่าใช้ตัวแปรสตริงก์ที่ถูกบ่งบอกในคำสั่ง FIELD ในคำสั่งการนำข้อมูลเข้า (input statement)

หรือบนด้านซ้ายของคำสั่งการกล่าวอ้าง (assignment statement) (คำสั่ง LET) สิ่งนี้เป็นเหตุให้ตัวบ่งชี้ (pointer) สำหรับตัวแปรนั้นชี้ไปยัง string space แทนที่จะชี้ไปยังบัฟเฟอร์ของไฟล์แบบสุ่ม

การเข้าถึงไฟล์แบบสุ่ม (Accessing a Random File)

ใช้ขั้นตอนต่าง ๆ ต่อไปนี้เพื่อที่จะเข้าถึงไฟล์แบบสุ่ม

1. เปิดไฟล์สำหรับการเข้าถึงแบบสุ่ม
2. ใช้คำสั่ง FIELD เพื่อที่จะจัดสรรเนื้อที่ว่างในบัฟเฟอร์แบบสุ่มสำหรับตัวแปรต่าง ๆ ซึ่งจะถูกอ่านจากไฟล์

   เราสามารถใช้กับกรณีของการ OPEN และการกำหนด FIELD เพียง 1 คำสั่งเท่านั้นในโปรแกรม
3. ใช้คำสั่ง GET เพื่อที่จะย้ายเร็คคอร์ดในแฟ้มข้อมูลแบบสุ่มที่ถูกต้องการลงสู่บัฟเฟอร์

ข้อมูลในบัฟเฟอร์อาจจะถูกดึงมาปฏิบัติงานโดยโปรแกรม มูลค่าจำนวนเลขต่าง ๆ ต้องถูกแปลงค่ากลับไปสู่จำนวนเลขต่าง ๆ ขั้นตอนนี้กระทำโดยฟังก์ชันการแปลงค่าต่าง ๆ เช่น CVI สำหรับจำนวนเต็ม, CVS สำหรับมูลค่าต่าง ๆ ในรูป single-precision และ CVD สำหรับมูลค่าต่าง ๆ ในรูป double-precision

โปรแกรมต่อไปนี้เข้าถึงไฟล์แบบสุ่มที่ชื่อ "FILE" ซึ่งถูกสร้างขึ้นโดยที่รหัสตัวเลข 2 หลักถูกป้อนเข้าไปที่แป้นพิมพ์ ข้อมูลที่เกี่ยวข้องสัมพันธ์กันกับรหัสนั้นจะถูกอ่านจากไฟล์และถูกพิมพ์

```
10 OPEN "R",#1,"FILE",32
20 FIELD #1, 20 AS N$, 4 AS A$, 8 AS P$
30  INPUT  "Z-DIGIT  CODE";CODE%
40  GET  #1, CODE%
50  PRINT  N$
60 PRINT USING "$$###.##";CVS(A$)
70  PRINT  P$:PRINT
```

```
80 GOTO 30
```

ฟังก์ชัน LOC สำหรับไฟล์แบบสุ่มต่าง ๆ จะทำหน้าที่ส่งค่า "หมายเลขเร็คคอร์ดปัจจุบัน(current record number)" หมายเลขเร็คคอร์ดปัจจุบันเป็นหมายเลขล่าสุดที่ถูกใช้ในคำสั่ง GET หรือคำสั่ง PUT ตัวอย่างเช่น คำสั่ง

```
IF LOC(1)>50 THEN END
```

สิ้นสุดการประมวลผลโปรแกรมถ้าหมายเลขเร็คคอร์ดปัจจุบันในไฟล์ #1 มีค่ามากกว่า 50

<u>โปรแกรมตัวอย่าง</u>

ต่อไปนี้เป็น inventory program ซึ่งแสดงการเข้าถึงไฟล์แบบสุ่ม ในโปรแกรมนี้หมายเลขเร็คคอร์ดถูกใช้ในฐานะเป็นหมายเลขแผนก (part number) และจะถูกกำหนดเงื่อนไขว่าในแฟ้มข้อมูลจะบรรจุหมายเลขแผนกไม่มากกว่า 100 หมายเลขแผนกโดยที่หมายเลขจะแตกต่างกัน คำสั่งไลน์ 900 ถึงไลน์ 960 ในโปรแกรม จะกำหนดค่าเริ่มต้นไฟล์ข้อมูลโดยการบันทึก CHR$ (255) ในฐานะเป็นอักขระตัวแรกของเร็คคอร์ดแต่ละเร็คคอร์ด ซึ่งจะถูกใช้(ไลน์ 270 และไลน์ 500) เพื่อที่จะเป็นตัวกำหนดว่ารายการเข้า (entry) คือหมายเลขแผนกมีปรากฏอยู่หรือไม่ ไลน์ 140 ถึงไลน์ 210 พิมพ์ฟังก์ชันต่าง ๆ ของ inventory ที่แตกต่างกันซึ่งโปรแกรจะดำเนินการเมื่อท่านพิมพ์หมายเลขฟังก์ชันที่ต้องการเข้าไป ไลน์ 230 จะส่งไปยังสับรูทีนที่ตรงตามเงื่อนไข

inventory Program

```
120 OPEN"R",#1,"INVEN.DAT",39
130 FIELD#1,1 AS F$,30 AS D$,2 AS Q$,2 AS R$,4 AS P$
140 PRINT:PRINT "FUNCTIONS:":PRINT
150 PRINT 1,"INITIALIZE FILE"
160 PRINT 2,"CREATE A NEW ENTRY"
170 PRINT 3,"DISPLAY INVENTORY FOR ONE PART"
180 PRINT 4,"ADD TO STOCK"
190 PRINT 5,"SUBTRACT FROM STOCK"
```

```
200 PRINT 6,"DISPLAY ALL ITEMS BELOW REORDER LEVEL"
210 PRINT:PRINT:INPUT"FUNCTION";FUNCTION
220 IF (FUNCTION<1)OR(FUNCTION>6) THEN PRINT
       "BAD FUNCTION NUMBER":GOTO 140
230 ON FUNCTION GOSUB 900,250,390,480,560,680
240 GOT0 210
250 REM BUILD NEW ENTRY
260 GOSUB 840
270 IF ASC(F$)<>255 THEN INPUT"OVERWRITE";A$:
       IF A$<>"Y" THEN RETURN
280 LSET F$=CHR$(0)
290 INPUT "DESCRIPTION";DESC$
300 LSET D$=DESC$
310 INPUT "QUANTITY IN STOCK";Q%
320 LSET Q$=MKI$(Q%)
330 INPUT "REORDER LEVEL";R%
340 LSET R$=MKI$(R%)
350 INPUT "UNIT PRICE";P
360 LSET P$=MKS$(P)
370 PUT#1,PART%
380 RETURN
390 REM DISPLAY ENTRY
400 GOSUB 840
410 IF ASC(F$)=255 THEN PRINT "NULL ENTRY":RETURN
```

```
420 PRINT USING "PART NUMBER ###";PART%
430 PRINT D$
440 PRINT USING "QUANTITY ON RAND #####";CVI(Q$)
450 PRINT USING "REORDER LEVEL #####";CVI(R$)
460 PRINT USING "UNIT PRICE $$###.##";CVS(P$)
470 RETURN
480 REM ADD TO STOCK
490 GOSUB 840
500 IF ASC(F$)=255 THEN PRINT "NULL ENTRY":RETURN
510 PRINT D$:INPUT "QUANTITY TO ADD ";A%
520 Q%=CVI(Q$)+A%
530 LSET Q$=MKI$(Q%)
540 PUT#1,PART%
550 RETURN
560 REM REMOVE FROM STOCK
570 GOSUB 840
580 IF ASC(F$)=255 THEN PRINT "NULL ENTRY":RETURN
590 PRINT D$
600 INPUT "QUANTITY TO SUBTRACT";S%
610 Q%=CVI(Q$)
620 IF (Q%-S%)<0 THEN PRINT "ONLY";Q%;" IN STOCK":GOTO 600
630 Q%=Q%-S%
640 IF Q%=<CVI(R$) THEN PRINT "QUANTITY NOW";Q%;
      " REORDER LEVEL";CVI(R$)
```

```
650 LSET Q$=MKI$(Q%)
660 PUT#1,PART%
670  RETURN
680 REM DISPLAY ITEMS BELOW REORDER LEVEL
690 FOR I=1 TO 100
710    GET#1,I
720 IF CVI(Q$)<CVI(R$) THEN PRINT D$;" QUANTITY";
       CVI(Q$) TAB(50) "REORDER LEVEL";CVI(R$)
730 NEXT I
740  RETURN
840 INPUT "PART' NUMBER";PART%
850 IF(PART%<1)OR(PART%>100) THEN PRINT "BAD PART NUMBER":
       GOT0 840 ELAE GET#1,PART%:RETURN
890 END
900  REM INITIALIZE FILE
910 INPUT "ARE YOU SURE";B$:IF B$<>"Y" THEN RETURN
920 LSET F$=CHR$(255)
930 FOR I=1 TO 100
940 PUT#1,I
950 NEXT I
960 RET
```