Apple Ⅱ
資料檔案設計

許志堅 譯

衛星出版社

Apple Ⅱ
資料檔案設計

許志堅 譯

衛星出版社 印行

Apple Ⅱ 資料檔案設計

譯　者：許　志　堅
出版者：衛星出版社
發行者：衛星出版社
地　址：香港英皇道380號二樓
印刷者：達成印刷廠
地　址：香港灣仔利東街10號

目　錄

第一章

編寫清晰、易讀、富有邏輯的 BASIC 程式

目的：當您讀完本章的內容之後，您就可以

1. 利用“由上至下”（top-to-bottom）的方法描述程式的編寫法。
2. 利用REMARK 敍述編寫一個“說明模組”（introductory module）。
3. 利用七種規則來編寫程式以節省記憶體空間。

前 言

本書的目的是教您如何使用APPLESOFT BASIC的資料檔（data file）。學習本書之前，您必須先具備基本的BASIC程式編寫能力。您不但要看得懂BASIC的程式，同時還要會寫一些簡單的BASIC語言。因此本書並不適合BASIC語言的初學者使用，本書適合那些具有BASIC語言基礎但從未用過資料檔的人使用。而本書所列舉的資料檔使用法是針對著APPLE II計算機的BASIC語言使

用的。

其他計算機或是其他版型(version)的資料檔與本書所列舉的資料檔類似，但不完全相同。當您閱讀本書時，最好同時與APPLE II的參考手册和磁碟操作系統(disk operating system)手册對照。雖然本書中所敍述的循序(sequential)和隨機(random)資料檔已經相當完備而深入，但是它仍舊無法取代APPLE II 磁碟操作系統手册。

因爲您已經具備編寫BASIC程式的能力，而且也寫過數個小程式，所以下一個步驟是如何使程式更清晰、更具組織。與資料檔案(data file)有關的程式可能相當大、而且複雜，相對地，程式除錯(debug)的過程——使程式能正確無誤的工作——也可能變得相當複雜。因此您務必要詳讀本章所提供的各種程式組織(program organization)方法，以便在程式編寫時，收事半功倍之效。

BASIC語言

計算機語言BASIC是在1960年代初期由Dartmouth 學院研究發展而成的。這種語言是專爲初學者或略具計算機知識的人設計的。最初的BASIC語法(syntax)只包含一些初學者可能用到的功能而已。但是其他的學校、廠商、機構紛紛將適合自己業務需要的部份加到BASIC之中，因此各種不同版型的BASIC語言相繼出現，例Extended BASIC, Expanded BASIC, SUPERBASIC, X-BASIC, BASIC PLUS 等等。直到1978年，美國國家標準局(American National Standards Institute, 簡稱ANSI)才製訂一套工業標準的BASIC語言。但是這一套標準仍然只是具備必要的、一般的語法，所以各廠商、機構仍然以這個標準爲基礎，再增加各

種不同的特性。因此目前所使用的BASIC語言可說是大同小異。

在微型電腦的應用上，使用最廣的BASIC版型是Microsoft 公司所發展的MICROSOFT BASIC。許多不同的微型電腦都有這一套語言，可惜每一種微型電腦所使用的方法仍然不盡相同，所以在APPLE機器上的MICROSOFT BASIC稱爲APPLESOFT。

本書中所列舉的所有程式均可在使用DOS 3.3型（或DOS 3.2型）的APPLE II和APPLE II PLUS機器上實際操作。所有的程式都是利用APPLESOFT BASIC語言寫成的。如果您所使用的是INTEGER BASIC，則您必須依據參考手册上的方法將APPLESOFT改成INTEGER。同理您必須依據DOS手册所載的方法修改本書中的檔案輸入指令。

我們儘可能地使用各種機器版型共通的特性來編寫程式。本書的目的是提供您如何編寫一個易懂的程式，而不是在炫耀APPLESOFT BASIC 的特性。

您應該使用那些BASIC語言

穩當的程式編寫法

因爲從現在起，您將會碰到一些較長，較複雜的程式，所以您必須採用“穩當的程式編寫技巧”，以便隔離錯誤、尋找錯誤、和改正錯誤。（再熟練的程式員也會出錯的）。換句話說，在您完全瞭解並測試該指令之前，不要貿然使用該指令。（尤其是較複雜的指令）。此外，即使您對於某些特殊指令非常熟悉，您非到萬不得已，最好不要使用這些特殊指令。因爲這些特殊指令常常是某一機器的特有功能

，例如列表或圖形（graphic）輸出，而其他的機器可能沒有這種指令。因此您使用愈穩當、愈保守的程式編寫技巧，則您所可能遇到的困擾愈少。本章所敍述的就是“穩當的程式編寫技巧”。

使用穩當的程式編寫技巧理由之一是：您可以很容易地將程式轉換到其他的機器上。但是您可能會問：“我爲什麼要將程式轉換到其他機器上？”，理由很簡單，因爲您很可能會與其他朋友交換學習的心得，彼此交換一些程式，這時您能保證您的朋友都使用與您同一型的機器嗎？如果不能，則您必須再修改那些程式才能適應其他的機器。因此，穩當的程式編寫技巧可以減少“修改程式”的困擾。

再者，您以後所使用的機器可能與您現在使用的機器不同，因此爲了避免日後轉換的困擾，您最好現在就使用穩當的程式編寫技巧。

使用穩當的程式編寫技巧的理由：
- 易於隔離錯誤、尋找錯誤。
- 避免程式的“死結”。
- 易於轉換到其他機器上。

編寫易讀的程式

本書中的所有例題都是用最簡單、最直接的BASIC指令寫成的，所以您很容易就能看懂程式的意義。由於這些程式所用的指令都相當簡單易懂，所以在感覺上，它們並不像是“計算機程式”，而像是“簡單的課外讀物”。

如果您也想編寫易懂的BASIC程式，則您必須事先審愼考慮，計劃程式的流程（flow），設計美觀的輸出報表格式（format）。如

果您的職業是程式員，您可能必須率就公司所規定的程式編寫風格。但是如果您是業餘的電腦迷，則您使用易懂的程式編寫法將有助於日後的除錯和修正工作。

“易懂的程式編寫風格”本身具有“文件記錄”（documentation）的能力，因此讀者或使用者很快就能瞭解整個程式的眞正工作原理。雖然我們借用了許多“結構化程式編寫”（structured programming）的特性，但嚴格說起來，本書所使用的方法還不能算是結構化的程式編寫法。我們是用許多“模組”（modules）組織成一個完整的程式，而每個模組都有一個“中心任務”或“主要功能”。本書所提供的方法不見得能節省記憶體空間，也不見得能增快程式執行的速度。相對地，我們是以這些代價來達到“程式易讀性”的要求。但是在本章結尾，我們仍將提供一些用來縮短程式長度、縮減程式執行時間的程序、方法。一般而言，模組式的程式有助於程式本身的執行，也有助於您與其他讀者之間的思想交流。

由上而下的組織

當您設計程式時，最好能把握重點，由程式的主功能（major functions）下手。這些主功能包括：

資料登錄（data entry）
資料分析（data analysis）
數值計算（computation）
檔案更新（file update）
資料編校（editing）
報表產生（report generation）

使用我們所建議的模組處理方式將程式分割成許多模組，每個模組包含上述的主功能之一。而程式的流程必須由一個模組到下一個模組。這種由上而下的組織（top-to-down organization)使您的程式更易於處理。您可以將程式模組再細分成更小的程式塊，每個程式塊執行一個簡單的程序（procedure）或計算。程式塊的大小可依程式的大小、內容、程式員的風格而定。所以程式塊的式樣依人、依程式而不同。

使用模組的格式和由上而下的組織

REMARK敍述

在程式中，我們可以利用REMARK 敍述或空白來分隔各個程式模組和程式塊。通常我們利用這種方法來增加程式的明晰度和可讀性。但是在一個程式中也不應該使用太多的REMARK 敍述。如果您打完程式的敍述編號之後緊跟著一個冒號（150:），則程式就會出現一行幾近空白的敍述。當然，在敍述編號後頭接著REM（150 REM）也會產生類似的效果

```
100  REM    DATA ENTRY MODULE
110  REM  **** READ DATA FROM DATA STATEMENTS 9000-9090
120
130  REM
200  REM    COMPUTATION MODULE
210  REM ARK
```

（注意：APPLE 機器會將REMARK 這個字分成兩部份，例如編號210的敍述。因此我們建議您使用REM，而不要寫成REMARK）。

上例中，我們在程式模組、程式塊、或副常規（subroutine）的一開頭就使用來解說程式的REM 敍述（編號100和110的敍述

）。然後利用一個幾近空白的敍述（編號120）或空白的REM敍述（編號130）表示結束。

如果您有系統地使用REM敍述，則可增加程式的可讀性。有些人在說明之前加一些星號（編號110），有些人則先空數個空白字元之後才開始解說的敍述（編號200）。這兩種方法都可用來隔離解說的敍述和眞正的BASIC指令。

您也可以利用一行中有數個敍述的方法，將BASIC指令與REM解說寫在同一行之中。這時，您的REM解說必須是最後一個敍述才行。這種“立即”解說可用來解說每一個敍述的作用。通常，我們都在BASIC指令與解說之間保留一個相當的空白。如下例：

```
220  LET C(X) = C(X) + U: REM ***COUNT UNITS IN C ARRAY
240  LET T(X) = T(X) + C(X): REM ***INCREASE TOTALS ARRAY
```

我們有必要利用REM來解說程式的功用，但是我們不可濫用。例LET C=A+B 根本不須再解說了。REM是用來提供更多的訊息，而不只是用來說明一些簡單易懂的步驟而已。

就如同其他事件一樣，本章所敍述的也有許多例外。但是我們的目的只是想讓您再考慮一下您所慣用的程式編寫技巧與格式是否合適罷了。因此下述的各項“規則”只不過是一種建議罷了，（可能會有例外的情況）。

GOTO敍述

在BASIC語言中最引人爭議的敍述要算是無條件的GOTO敍述了。有些人主張根本不要使用無條件的GOTO敍述（例GOTO 100）。另一些人則主張合理的使用：GOTO敍述或GOSUB敍述可以跳到編

號比自己大的敍述上。這與由上而下的程式組織並不衝突。同理，同樣的法則也可用到 IF…THEN 敍述上。
（這個敍述可能存在着許多例外）。

```
140  GOTO 210
150  IF X < Y THEN 800
160  GOSUB 8000
```

最後的建議：GOTO, GOSUB, 或 IF…THEN敍述不可跳到一個只有 REM 指令的敍述上。因爲如果記憶體位置不敷使用時，您可能會刪除一些 REM 敍述，這時這些GOTO指令也可能跟著變動。某些BASIC 語言根本就禁止程式跳到一個 REM 敍述上。

劣

```
150  GOTO 300

300  REM    DATA ENTRY
310  INPUT "ENTER NAME:";N$
```

優

```
150  GOTO 300

290  REM    DATA ENTRY
300  INPUT "ENTER NAME:";N$
```

程式說明模組的格式

您應該將BASIC 程式中的第一個模組（編號100至199或編號1000 到1999 ）保留用來說明整個程式的功能，使用指南，程式中所用變數的意義，各變數、常數、數列的初值等等。

在程式一開頭的 REM 敍述可能是一個程式名稱。您所選的程式名稱應該讓讀者一眼就知道該程式的目的。接著可能是作者或程式員的名字和編寫該程式的日期。爲了讓使用者更加清楚起見，您也可以將機器名稱、編號，軟體系統等資料寫在 REM 敍述中。但是一旦程式有所變動，這些REM敍述也要跟着更新。

```
100   REM     PAYROLL SUBSYSTEM
110   REM     COPYRIGHT CONSUMER PROGRAMMING CORP. 8/81
120   REM
130   REM     HP 2000 BASIC
140   REM     MODIFIED FOR APPLESOFT BASIC BY J. BROWN
150   REM     ON APPLE II, 48K
```

接着您可以用REM指令或PRINT指令，簡略地介紹程式的作用接下來是使用指南，您可以讓使用者自行選擇是否要輸出使用指南。如何使用指南非常冗長，則您可以將敍述部份分離成一個副常規，當使用者要求輸出使用指南時才跳到該副常規去執行。

```
170   REM     THIS PROGRAM WILL COMPUTE PAY AND PRODUCE PRINTED PAYROLL
180   REM     REGISTER USING DATA ENTERED BY OPERATOR
190   REM
200   INPUT "DO YOU NEED INSTRUCTIONS?";R$
210   IF R$ = "YES" THEN  GOSUB 800
220   REM
```

跟在說明和使用指南之後的 REM 敍述是變數（ variable ）、字串（ string ）、陣列（ array ）、常數、和檔案的說明。這些說明可以讓讀者更容易瞭解程式，同時也利於日後更新程式之用。通常我們都是寫完程式之後再加上這一整個模組，以免遺漏任何重要的訊息。

程式中所用到的所有常數最好都賦予它一個變數名稱。因爲在某一次執行時，常數值保持不變，但是在下一次執行時，我們可能想將常數改成另一個值。所以賦予常數一個變數名稱可以使上述工作易於處理，您只須改變一個敍述就可以了。總之，您在編寫程式時最好隨時記錄一些重要的解說（程式變動，解說也隨著更新），等程式完全正常工作之後再將這些解說寫成 REM 敍述。

```
220  REM   VARIABLES USED
230  REM     G=GROSS PAY
240  REM     N=NET PAY
250  REM     T1=FEDERAL INCOME TAX
260  REM     T2=STATE INCOME TAX
270  REM     F=SOC.SEC.TAX
280  REM     D=DISABILITY (SDI) TAX
290  REM     X,Y,Z=FOR-NEXT LOOP CONTROL VARIABLE
300  REM     H(X)=HOURS ARRAY
310  REM     N$=EMPLOYEE NAME (20 CHAR)
320  REM     PN$=EMPLOYEE NO. (5 CHAR)
330  REM
340  REM   CONSTANTS
350  LET FR = .0813: REM    SOC.SEC. RATE
360  LET DR = .01: REM    SDI RATE
370  REM
380  REM    FILES USED
390  REM      ITM=FEDL. TAX MASTER FILE
400  REM      STM=STATE TAX MASTER FILE
410  REM
```

（注意編號310和320的敍述中用來說明字串長度的方法）
（注意編號350和360敍述中的“立即解說”）。

程式說明模組的最後一個部份是定初值（initialization）部份。您可以在這一部份說明所有陣列、字串的大小。即使您所使用的機器不用爲陣列等定DIMENSION，但是爲了清晰起見，您最好加上這些說明。同時您也可以在這一部份將必要的變數初值定爲零。（爲了清晰起見，即使計算機會自動將所有變數的初值設定爲零，您最好再設定一次）。這一部份同時包括一些您自行定義的函數（function）等。

```
410  REM     INITIALIZE
420  :
430  DIM H(7),R(10,13),N$(30)
440  :
450  REM
```

接在說明模組之後的其他模組

程式的其他部份由一些主功能模組和副常規組成（有時還包括一些DATA 敍述）。記得在各個模組之間利用REM敍述（空白的REM