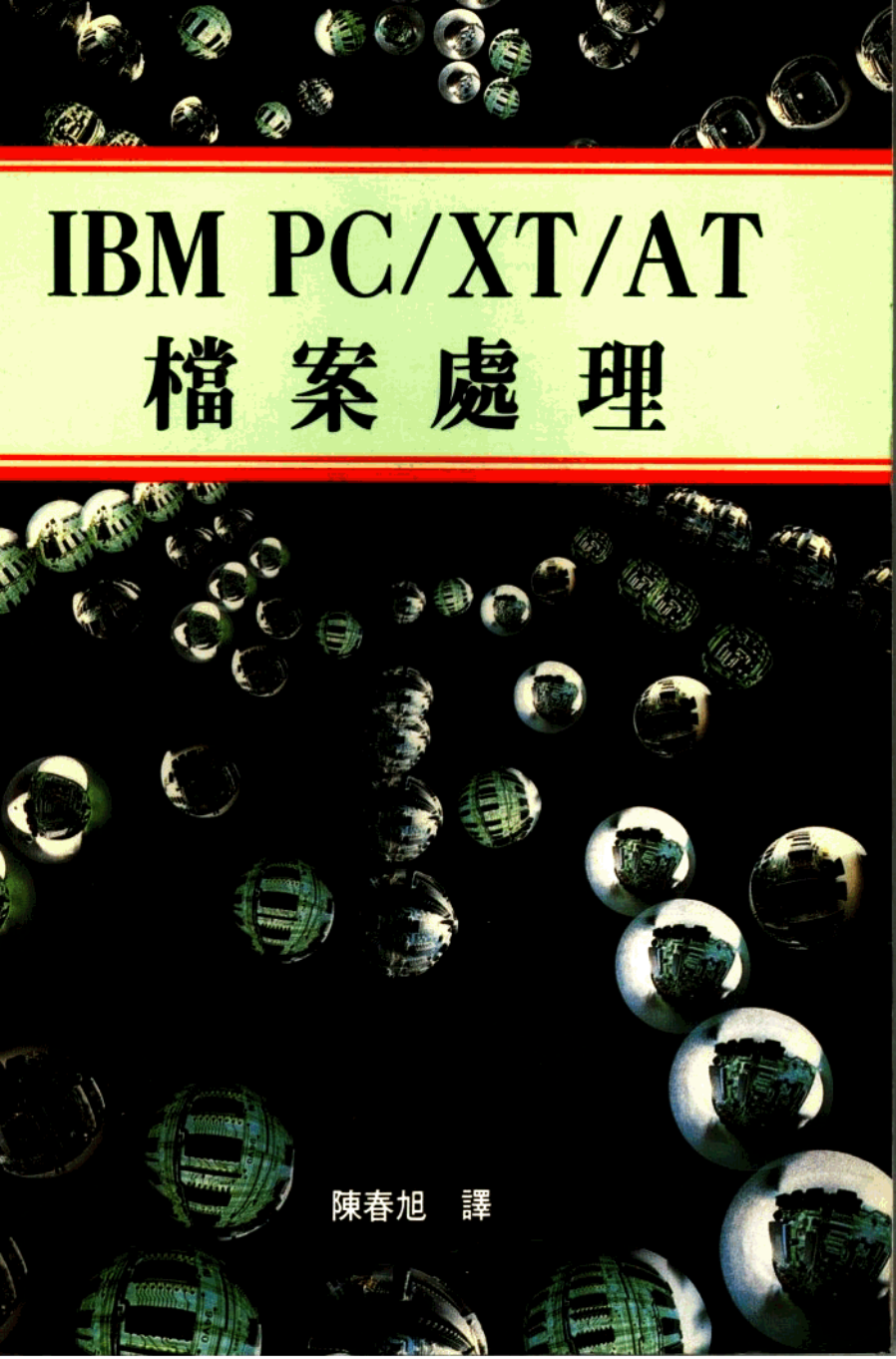
IBM PC/XT/AT
檔案處理



陳春旭　譯

序

收集及維護資訊對電腦至微電腦而言，是最普通也是最熟悉的應用之一。數以百計的通用性及特殊用途之“檔案管理程式”（file manager）及“資料庫管理程式”（data base manager）都已邁入商業化。

使用IBM PC撰寫BASIC程式的使用者，都想要在他們的程式中，加入能在磁碟上儲存及擷取資料的能力。但是，目前所獲得可資取用的許多文件——由電腦所供應，或是有關BASIC書籍所提供的資料，所帶給我們的困擾比幫助還多。

有時候，這些文件假設你知道“記錄”（records）、“欄位”（fields）、“緩衝區”（buffers）以及“隨機存取”（random access），而不加以說明這些名詞。又有些時候，書本認爲這些概念對讀者而言太難了解。

但我絕不做這種假設。如果你讀這些名詞而仍對IBM PC BASIC或資料檔有所疑惑，則本書正適以滿足你的疑點，並且給你一個徹底的概念，如以一個BASIC的程式師的身份而言，你能對資料檔做些什麼樣的工作。我假定你已擁有寫作BASIC程式的經驗，但不限於PC BASIC上的經驗。

本書中除了資料檔的處理外，還提供其他的知識。優良的資料處理程式的基本要求之一是容易使用，甚至對不懂電腦或程式的人亦復如是，同時要儘量確保儲存在檔案中資料的正確性。

所以，本書同時也強調說明，確定資料正確輸入的技巧，並且探討BASIC INPUT敍述的限制.同時提供技巧來幫助使用者輸入正確的資訊；說明如何排序及列印檔案中的資料。

最後，資料處理程式可以變得相當大且複雜。所以，本書只是談論

BASIC 程式規劃的一般技巧——包括程式結構的控制、有效地使用副常式以及使程式更具“方便使用”的特性。

程式師傾向於使用專有名詞。而撰寫BASIC 手册此種技術文件的人，也經常假設讀者熟悉諸如“聯結”(concatenation)等名詞。所以，本書中在使用到專有名詞或是技術名詞時，皆會加以解釋。因此讀本書的一個附帶效果是你會學到許多新名詞；更有甚者是獲得一些和普通字典中不一樣的定義。

同時本書也包括了一個基本資料處理程式的完整列表，此程式所採取的撰寫方式，你可以非常簡單地將其更新成各式各樣不同的應用。

甚至在你不想使用BASIC 撰寫資料處理程式的狀況時，我們仍將幫助你揭開資料如何儲存於磁碟的“面紗”，同時提供你對程式設計及使用者介面有較深的了解。這應該可以幫助你成爲程式方面較富知識的買者及使用者——特別是在資料處理程式方面。

目　錄

第一章

資料檔導論

IBM個人電腦是一種功能相當多的工具 。在同一個螢幕上，使用同一個鍵盤，你可以：

- 使用試算表程式來計算商業上複雜的“what ifs”問題。
- “駕駛”一架模擬的飛機然後予以安全地迫降，或是玩數百種的遊戲，其中有許多是富教育性的。
- 使用文書處理程式來撰寫、編輯、規劃或列印信件、書籍等各類文件。
- 藉電話線和其他個人或大型電腦通訊。
- 使用多種的程式語言來撰寫、測試及執行你自己的程式。
- 保存庫存、薪水、家庭預算、聖誕卡列表、收集記錄、股票等資訊。

本書主要是針對最後一個領域有興趣的人們而撰寫的，即使用 PC 的軟式磁片或硬式磁碟來儲存、控制及擷取資訊；同時也是爲了想要增進BASIC程式技巧的人所寫，而不論其是否學習過有關資料檔的處理 。

PC是資料管理系統

如果你看過以 PC 爲取向的雜誌廣告，你會發現有上百的程式設計是用來處理及操作資訊。其中某些是通用性的「資料庫管理程式」(Data Base

Manager），而其他的都是爲特殊需要而設計的，如由養豬管理到醫藥帳單，從會計到聖經的用語索引，無所不包。

既然你幾乎可以確定能買到現成的程式，用以管理對你而言是重要的資訊，那麼爲什麼你還要嘗試著自己寫呢？

第一點，因爲“店裏買的”程式比較昂貴，且愈珍奇的應用，價錢愈高。

第二點，因爲普通可以買到的程式也許不能正好滿足你所要的特殊應用。一般而言，你的電腦應該爲你工作，而不是你改變你的作法來適合它的需要。

第三點，或許是最重要的一點，因爲寫你自己有用的資料處理程式，對自身而言，是一個能得到滿足且值得的（雖然有時候是挫敗的）經驗。如果你需要一個新的功能，只需加進去即可，若它工作的方式不是你想要的，只要修改它即可，甚至如果你決定買WonderBase，而把自己寫的程式放在櫃子中冷凍起來，那麼你也已學到建立資料處理程式的一些方法。這使你成爲較佳的使用者及買者。

我得先提醒讀者一件事並下一個定義。我要提醒的是，本書並不是爲完全沒有BASIC程式經驗的人所寫。我先假設你相當熟悉IBM BASIC編輯程式，曾經使用及執行過一些程式，並且大概知道字串變數（string variable）、FOR-NEXT迴圈、GOTO及GOSUB等等。另一方面，也假設你並非是一個專家，因爲如果你是位專家，你便完全知道資料檔，也不用來讀本書了。

對資料檔的第一個定義

以下定義我所謂的資料檔。基本上，資料檔是儲存在軟式磁片或硬式磁碟上具結構性資料的集合。一般而言，資料檔由一個程式所建立，並且由該程式或其他程式來編輯讀取，同時多半（但不是必要的）在原設計時使用的電腦或其相容（compatible）電腦上處理。

請注意：我所說「具結構性資訊」，這對定義而言是很重要的。例如，本書的每一章都可存在磁片上的檔案中，但這些檔案都不是資料檔，因爲它們所包含的資訊是以單字（word）、句子（sentence）及段落（paragraph）

等單位所組成，而每個單位長度都是可變的。雖然句子有主詞、動詞和受詞之別，但並沒有特殊符號來告訴讀者區別它們的方法。人類知道如何來解譯這些資訊，然而電腦縱使強而有力，却無法“讀取”這些文章。

資料檔可包含字元、數目及他種資訊，但不是以句子或段落來組合，而是在「欄位」(field)及「記錄」(record)的單位結構中。且由於電腦不能和人一樣有能力來解譯資訊，因此資料檔必須有一些明確的規則來指引電腦解譯這些結構。讀者知道句子可以使用句點、問號或驚嘆號來結束，如：

句子

```
These are sentences. As humans, we can read them!
Amazing, no?
```

圖1.1 電腦不像人類，它需要精確的規則來指引它解譯句子、檔案等

電腦需要更正確的線索。它會找尋標點符號，如引號(“ ”)或逗點(,)來區分檔案中一筆一筆的資訊。或是程式師所以提出的每一筆資訊都有某一特定長度的規定——例如12個字元。電腦是智慧型機器，即使它有什麼缺點，但在計數上皆非常擅長，所以當其接受12個字元後，程式便“知道”它已讀完一筆完整的資料項目(假設資料確實由該項目的開始點開始)。

一筆記錄

```
12345, ''housecoat'',37,4.95
```

另一筆記錄

```
anthill      |$49.95     |372    |Nature Products Inc.
```

圖1.2 電腦精於計算並且可分辨特定長度的資料

資料檔結構的兩個不同方法，是“標點”或堅持每一項目有相同長度，此乃IBM PC BASIC在處理資料檔時所用的兩種方式之核心。我們將探

入研究標點方式（循序檔）及計數方式（隨機存取檔），以便在選擇使用方法時可以做正確的判斷。

身爲一個程式師，選擇檔案中儲存資訊的方式是非常重要的，但對於使用程式的人而言，除了程式執行的速度及使用磁碟設備的效率外，其儲存資訊的方式並不那麼重要。從使用者的觀點，我們描述的資料檔可以被想成一個檔案夾中包含許多相同的“表單”（Form），而每一表單都有許多資訊。

我們都曾經填過許多這種表單，其中“姓”填在特定的空位，名字填在第二格，第三格填地址等等。如果你想用人工把好幾百張這種表單按照郵遞區號來排列。則你的眼睛就要快速掃描表單中包含郵遞區號的地方，而不必花精神在別的資料上。有時候表單上的項目會有一條底線或部份的底線用以指導使用者填入資料：

Last: _____ First: _____

而其他時候，項目的直線會把每一字元分割成一格一格的空間：

Last: |_|_|_|_|_|_|_|_|_|_| First: |_|_|_|_|_|_|_|_|_|_|

圖1.3 表單中的項目可以用直線、部份直線或小盒子來標示

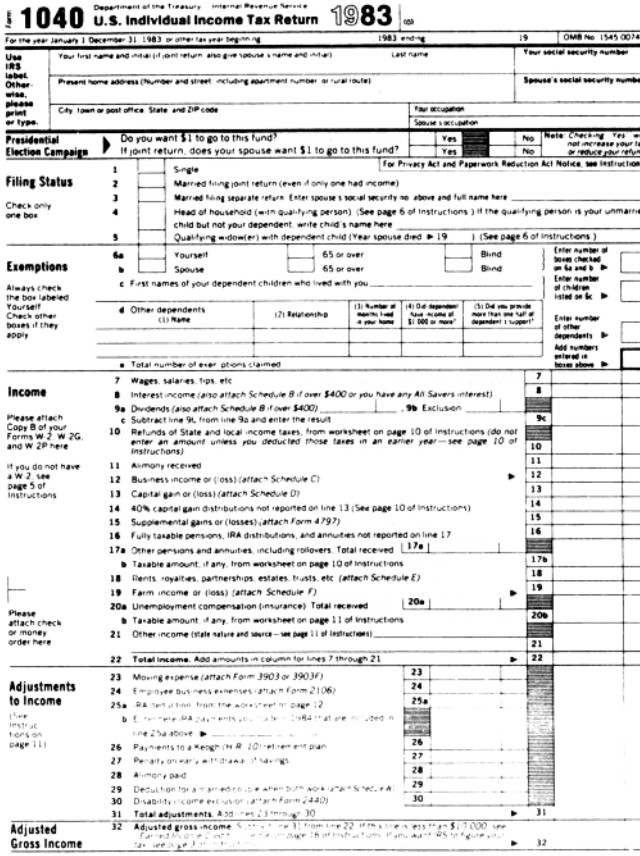
你可以看出，第二種格式特別適合於依賴計算字元來建立資訊結構的資料儲存技巧。

從本書中你也可以了解，電腦迅速地找尋資訊及比較各筆資訊間關係的能力，大部份有賴於程式師將所有的資訊以有次序的方式組合。如果想要我們的程式找出所有某一特定郵遞區號的“表單”時，最好有一個方法能使電腦迅速且無誤地找到該資訊。使用文書處理程式，你可以寫一個包含名字、地址及郵遞區號的檔案。除非你在輸入的資訊上加上嚴格的規定以利管理，否則很難完成任務，諸如以郵遞區號來排列資訊或使用檔案來列印郵寄標籤。

如同列印的表單一樣，對電腦組織的方式最有幫助的莫過於其位置。在列印的表單上，你的眼睛只要看郵遞區號的地方；如果你要以郵遞區號來排

列信件，只要注視郵件的右下角便可。相同地，電腦需要有確定地址的起始點——就如同，也需確定前往何方及往那個方向擷取多少個字元的特定指令。如果你在文書處理程式上建一個郵寄名單（mailing list），則每一項目以一新列開始（舉例說），或者堅持每一地址只能佔一列，這些方式對工作都會有所助益，因爲當程式看到檔案中的新列時，它便知道下一個字元就是記錄中新欄位的開始。

當然，使用列印的表單其主要理由是收集資訊，並確定資訊是完整的且儘可能地正確。以下是一個熟悉的表單，其中有設定好位置的各種不同資訊:



Form 1040 Department of the Treasury—Internal Revenue Service
U.S. Individual Income Tax Return 1983 (99)

For the year January 1–December 31, 1983, or other tax year beginning 1983, ending 19, OMB No. 1545-0074

Use IRS label. Otherwise, please print or type.
Your first name and initial (if joint return, also give spouse's name and initial) Last name — Your social security number
Present home address (Number and street, including apartment number, or rural route) — Spouse's social security number
City, town or post office, State, and ZIP code — Your occupation; Spouse's occupation

Presidential Election Campaign — Do you want $1 to go to this fund? Yes / No. If joint return, does your spouse want $1 to go to this fund? Yes / No. Note: Checking "Yes" will not increase your tax or reduce your refund.

Filing Status (Check only one box) — For Privacy Act and Paperwork Reduction Act Notice, see Instructions.
1 Single
2 Married filing joint return (even if only one had income)
3 Married filing separate return. Enter spouse's social security no. above and full name here
4 Head of household (with qualifying person). (See page 6 of Instructions.) If the qualifying person is your unmarried child but not your dependent, write child's name here
5 Qualifying widow(er) with dependent child (Year spouse died ▶ 19). (See page 6 of Instructions.)

Exemptions (Always check the box labeled Yourself. Check other boxes if they apply)
6a Yourself — 65 or over — Blind
b Spouse — 65 or over — Blind
Enter number of boxes checked on 6a and b ▶
c First names of your dependent children who lived with you
Enter number of children listed on 6c ▶
d Other dependents (1) Name | (2) Relationship | (3) Number of months lived in your home | (4) Did dependent have income of $1,000 or more? | (5) Did you provide more than one-half of dependent's support?
Enter number of other dependents ▶
Add numbers entered in boxes above ▶
e Total number of exemptions claimed

Income (Please attach Copy B of your Forms W-2, W-2G, and W-2P here. If you do not have a W-2, see page 5 of Instructions.)
7 Wages, salaries, tips, etc.
8 Interest income (also attach Schedule B if over $400 or you have any All Savers interest)
9a Dividends (also attach Schedule B if over $400) . 9b Exclusion . c Subtract line 9b from line 9a and enter the result
10 Refunds of State and local income taxes, from worksheet on page 10 of Instructions (do not enter an amount unless you deducted those taxes in an earlier year—see page 10 of Instructions)
11 Alimony received
12 Business income or (loss) (attach Schedule C)
13 Capital gain or (loss) (attach Schedule D)
14 40% capital gain distributions not reported on line 13 (See page 10 of Instructions)
15 Supplemental gains or (losses) (attach Form 4797)
16 Fully taxable pensions, IRA distributions, and annuities not reported on line 17
17a Other pensions and annuities, including rollovers. Total received
b Taxable amount, if any, from worksheet on page 10 of Instructions
18 Rents, royalties, partnerships, estates, trusts, etc. (attach Schedule E)
19 Farm income or (loss) (attach Schedule F)
20a Unemployment compensation (insurance). Total received
b Taxable amount, if any, from worksheet on page 11 of Instructions
21 Other income (state nature and source—see page 11 of Instructions)
22 Total income. Add amounts in column for lines 7 through 21

Adjustments to Income (See Instructions on page 11)
23 Moving expense (attach Form 3903 or 3903F)
24 Employee business expenses (attach Form 2106)
25a IRA deduction, from the worksheet on page 12
b Enter "IRA" payments you made in 1984 that are included in line 25a above
26 Payments to a Keogh (H.R. 10) retirement plan
27 Penalty on early withdrawal of savings
28 Alimony paid
29 Deduction for a married couple when both work (attach Sch. W)
30 Disability income exclusion (attach Form 2440)
31 Total adjustments. Add lines 23 through 30

Adjusted Gross Income
32 Adjusted gross income. Subtract line 31 from line 22. If this line is less than $17,000, see "Earned Income Credit" on page 16 of Instructions. If you want IRS to figure your tax, see page 3 of Instructions.

圖1.4　這個表單要我們把資訊填在一定的位置上

可以確定的是，這格式上的資訊將會被直接轉換到電腦的資料檔上。如果你把正確的號碼寫在表單上錯誤的位置，那麼幾乎可以肯定的是它會進入電腦中錯誤的位置，當以後電腦想要在這上面做事時，你可能就會發現到其所產生的錯誤。本書的主要工作之一，是發展程式技巧來確定正確的資料會進入檔案中正確的位置；建立一個“使用者介面”(user interface)也許比單純從磁碟中存取資訊較重要，相對的也較困難。

當讀完本書後，你應該：

- 從一個程式師的觀點，要確知程式使用者眞正知道如何使用你所設計的程式；
- 了解如何從檔案中存取你所需的資訊，並且選擇有用的已規劃之資料格式；
- 進一步了解磁碟資料儲存體的基本概念，這包括記錄、欄位及鍵；
- 知道更多關於IBM PC及其他電腦如何工作。

設備的考慮

本書所撰寫的程式及程式片段都是執行於兩部雙面磁碟機、320K 記憶體及IBM單色顯示器的PC上。它們使用BASICA版本2.0（從DOS 2.0來的版本）來撰寫，其中有些DOS從DOS 2.1中修改而來。

對你而言，這有什麼差別呢？在大部份情形下是沒有差別的，但如果你的PC不同，也許需要以不同方法處理某些工作。

磁碟機：如果你只有單面磁碟機，程式可以正常工作，但是你的資料儲存體擁有較少的空間。雖然本書附錄所列的主要程式，皆假設程式磁碟只有一部磁碟機，但擁有兩部磁碟機還是有所幫助的（雖然不是非屬必要）。如果你計劃使用PC當成資料儲存體的工具，而且你只擁有一部磁碟機，則你應該愼重地考慮以取得更多的磁碟儲存體容量。如果你有一部XT或是有“winchester”硬式磁碟的其他電腦，則你可以用很好的方法來處理許多資訊。

顯示器：如果你有彩色／圖形板及一個彩色顯示器（不論是組合式的或是RGB），程式依然可以工作。但是在「螢幕管理程式」（Screen Manager）一章中，談論列在螢幕上各種規劃資料的格式時，你也許會想要嘗試一些包括色彩在內的變化。

讓我在這兒陳述一下個人的喜好，色彩對電腦遊戲及圖形非常有用。但是如果你計劃在螢幕上做許多工作，特別是文字及數字，則IBM單色螢幕或是解析度良好的單色顯示器是唯一的選擇。某些非IBM 的顯像板可以將單色顯示器的字元顯示清晰能力和商業程式的圖形表達能力結合在一起，或是可以在單色及彩色顯示器間轉換使用。

版本：BASIC 2.0一般皆可以和以前的版本相容；我們使用它來發展這些程式的原因是因爲它提供一些有用的新特性，其中包括某些和資料檔處理有直接關係的特性。在碰到最新BASIC 和舊版本不是完全相容的地方時，我會在文中提醒你。版本2.1用在PCjr 上，並且更正DOS 2.0 中的一些錯誤，但它的速度也慢了一點；如果你有一個正統的PC，則使用版本2.0會感覺舒適而流暢。版本3.0以上是應用在PC AT上，對“普通”的PC使用者並不會提供任何額外的優點。

如果你還使用DOS 1.1（或是DOS 1.0），你也許需要考慮提升至2.0或2.0之上。有時候這是相當複雜的工作，然而大部份的複雜面你都可以忽略，如對資料檔處理來說，它的主要優點是，磁碟片中你可存放資料的空間增加12％，因爲記憶體空間由360K取代以往的320K，但缺點是其使用較多的主記憶體。

BASIC對BASICA．當你啓動BASIC時，你可以在“磁碟”及“高階”版本中做選擇。“高階”版本BASICA提供較好的圖形及聲音功能，它也提供“事件陷阱”（event trapping），意謂著你的程式可以在某種事件發生時（如按某一特殊鍵時）做一些檢視的工作，而所付出的代價是BASICA佔較多的記憶體；另一個代價是減少“可移性”（portability），也就是說你的程式可以在別的BASIC電腦上工作的可能性。本書中只有一兩個範例程式需要使用BASICA來執行一些我並不贊同的特性。然而，如果你選

擇使資料程式一直監視PC 的功能鍵，以當成程式輸入之來源，就需要使用BASICA ，因爲功能鍵的監視是“事件陷阱”領域的一部份。在目錄及資料捕捉的討論中，將談及功能鍵的使用。我使用BASICA是出於習慣的緣故；其使用了將近1500個位元組（bytes）的記憶體。如果你想要知道你用BASICA所寫的程式能不能在BASIC中執行，則只要叫出BASIC並儘您所能地執行；如果有問題，你會得到--個“Advanced feature”的訊息，也就是說程式中含有需要BASIC的敍述。

記憶體：超過某一程度後，增加記憶體對BASIC程式並沒有什麼幫助，因爲程式無法用到它們，128K記憶體也許就夠了。

解譯程式：本書的程式爲“解譯”BASIC而撰寫的——乃指你PC DOS中的BASIC。你也可向IBM買BASIC編譯程式（compiler），它會把你所寫的“原始碼”（source code）轉換成PC直接了解的機器語言。因爲在解譯BASIC中，轉換的處理發生在程式執行時，所以非常慢且無效率可言，而編譯的程式執行起來總是比解譯過的程式快很多，如果你的BASIC程式使用很頻繁，則可考慮買一個編譯程式。

一般而言，你寫的程式會以解譯格式或是編譯格式來執行，但如果你有編譯程式，則有一些程式結構的特殊規則需要你花點精神來學習。在本書內文中，當我用到會引起編譯程式產生問題的命令或程式結構時，我會指明出來告訴你。

關於程式結構的一些“規則”

BASIC是由Microsoft爲IBM PC而完成的，並不是一個結構化程式語言。所謂結構化語言乃是對於程式中你可以做的事有硬性規定；這種規定的用意是有效利用電腦資源（記憶體、磁碟空間、處理時間等等），並且容易了解程式在做什麼。

但是BASIC讓你高高興興GOTO到副常式中或從其中出來（雖然你可能會造成“沒有GOSUB的RETURN”這種錯誤，以及其他可怕的內在

錯誤）。隨便地使用GOTO容易造成設計上的混淆，之所以稱呼這種效果爲“spaghetti programming”是因爲這種程式的流程圖看起來就像一盤義大利圓麵一樣地混亂。

更甚者，在解譯BASIC中，你可以在任何所要的地方建立新變數，而不管程式其他的地方是否曾經爲了不同目的使用過相同的變數名稱。你甚至可以寫一個程式，在執行時來修改程式本身。

BASIC這個缺乏強制結構的缺點令許多電腦學家大爲失望，他們宣稱任何初學BASIC 程式的人絕不易更改其漫不經心的習慣。

然而，仍然可以用結構化的方式來寫BASIC程式，而且我也試著教導你往那個方向學習。

以下是我所採用的結構化原則。

1. 程式由一些較小、定義清楚的工作所組成：如啓始螢幕、開啓一個檔案、取出記錄中有價值的資料、檢查輸入資料的有效性等等。每一個這種工作都是用副常式來完成。
2. 所有副常式第一列的號碼都要被100整除，例如1100，或24300。這對程式沒有什麼特殊的效果；只要讓所有GOSUB的敍述都有明顯的兩個0在目標列的末尾，如同：

```
170 GOSUB 1100
```

這對我們強調副常式是一個標準單位的觀念有所幫助。

3. 副常式的第一列及最後一列至少包括一個可看得見的註解REM，如下：

```
600 REM**This subroutine does important things***
=
=
690 RETURN '*********************************************
```

4. 不可以GOTO進入副常式或從副常式中直接離開，而且在副常式中也只有一個RETURN敍述。類似如下的敍述：

```
670 IF A$="DONE" THEN RETURN ELSE ....
```

應該被重寫成

```
670 IF A$="DONE" THEN GOTO 690
IF A$ = "DONE" THEN GOTO 690
```

其中 690 列包含副常式的一個也是唯一的一個RETURN 。

5. 如果副常式包含多於15-20 個程式列，它也許做比其所需要做的工作還多的工作，那麼你就應該檢視一下是否其執行好幾個不同的工作，而不只是執行已設定好的那一個。
6. 有時比較喜歡在每一個數目列內只有一個BASIC敍述，若有好幾個敍述則以冒號（:）分開是比較好的，或者有時候甚至是必須的，如果你使用解譯BASIC ，一般而言，在程式除錯和執行完畢後，總是希望將其“壓縮”（compress）成較少的程式列，這意謂著每一列有較多的敍述——因其可以執行得較快。而在編譯 BASIC中，就不須有這一項麻煩。
7. 這些規則有時都可以不遵守，當我不遵守時，會解釋其原因。

有許多充分的理由讓我們嘗試去遵守這些規則。其中之一，是當我們寫的程式既長又複雜時，在了解程式執行的邏輯途徑（有時稱“控制流程”（flow of control）），才不會使其變得困難而增加程式的複雜度。

把程式儘可能的縮減成小單位的另一原因，是用以節省我們的工作。我們只要知道如何處理螢幕上的資訊、檢視資料的有效性、或在檔案中搜尋特定的一筆資料，便可把這些相同的程式碼（code）*[1] 用在許多不同的程式上。假設你的副常式從1100 列開始一直到1290 列，是用以檢視某一特定的程式中，使用者是不是以可接受的格式輸入資料。我們可以在別的程式中，藉著載入（load）該程式來使用這些相同的程式碼，其方法爲輸入：

---

*[1] 我將時常使用“碼”（code）這個名詞做爲“程式敍述”（program statement）的同義字。而“Coding”就意謂著“撰寫程式敍述”，但却不是“programming”的同義字。Programming 包括設計程式一般形狀以及如何表現給使用者，而Coding只是使用電腦語言的特殊設備來完成設計而已。

```
DELETE 1-1099
DELETE 1291- (last number in program)*2
SAVE "DATACHEK",A
```

這會將1100到1290列存放在另一個叫DATACHEK的檔案中。以後當你要用在別的程式上時，只要輸入MERGE DATACHEK，這些列就會加到你希望它們出現的地方。

如果你沒用過這種強而有力的MERGE工具，則有兩件事要特別注意。第一，如果你要把磁碟中的程式檔合併到螢幕上的另一個程式時，磁碟上的檔案必須要以ASCII格式儲存，而不是BASIC所常用的"tokenized"格式。這就是儲存"DATACHEK"後面要加上",A"的原因。你會注意到以ASCII格式來儲存程式要花費較長的時間，而其結果的磁碟檔會比用普通方式儲存的相同程式還長。在下一章會對此形式所代表的意義做深入的討論。

第二件要注意的是列號(line number)。如果你正在建立的程式已經有1100列，或是任何與DATACHEK中相同數目的列，則當加進DATACHEK時，這些列會被破壞。處理的方法是在你所有的程式中都保留1100到1290列給資料檢查碼，或者是在合併前使用RENUM命令，把DATACHEK的列號轉換到其他組的可用數目上（並且記得把重新編號的版本以ASCII的格式儲存）。

在整本書中，當我列出以後程式可能會再用到的程式碼時，我會使用獨特的數目組，以便其可用在別的程式中，完成相同的目的。所以如果你把列出的程式碼輸入電腦中，我建議你所用的列號和我所用的一樣，除非你計劃最後編譯所有程式。

導論的最後註解：在BASIC中寫大程式幾乎免不了重新架構，甚至你在每存入一單列碼進入磁碟前都仔細計劃也是一樣。如果你寫的程式以1000開始而以1090結束，則以後若需要增加一列時，也許就不小心進入程式中某一個以前是獨立的部份。BASIC提供新編號的能力，但你不能指定重新

---

*2 BASIC 2.0允許你使用"DELETE 1291-"來刪除1291列以後，一直到程式終結的列。早先的BASIC要求你在任何情況下。均需指明DELETE的最後一列。