BASIC

Business Subroutines

for the Apple II™ and IIe™

BASIC 商業子程序

微型電腦叢書

Alan G. Porter and Martin G. Rezmer著　招兆鏗　方錦成譯校

Addison-Wesley Publishing Company　萬里書店有限公司出版

BASIC BUSINESS SUBROUTINES FOR THE APPLE II AND IIe

BASIC 商業子程序

Alan G. Porter　　Martin G. Rezmer 著

招兆鏗　方錦城譯校

Addison-Wesley Publishing Company

萬里書店有限公司

BASIC 商業子程序
Alan G. Porter　Martin G. Rezmer 著
招兆鏗　方錦城譯校
出版者：萬里書店有限公司
香港鰂魚涌芬尼街2號D
電話總機：5-647511～4
承印者：金冠印刷有限公司
香港北角英皇道499號六樓B座
定　價：港幣二十五元

（一九八五年三月版）

前　言

好些讀者，通讀“Applesoft 指導”多遍，使用 Apple電腦仍覺困難；也許即使學會 INPUT、PRINT、及 FOR-NEXT,但仍然不容易按某種意義的順序組合這些語句。至少，對於一位非專職程序員，仍然會發生這種情況,欲使用電腦做一些事,却不知道如何運用電腦。本書的目的，是說明讀過“指導”之後，還需進一步學習什麼程序設計技巧和風格的問題。而其中第一步,就是爲某些最普遍的程序設計問題提供一些解答。我們準備先弄清楚每個問題，說明如何解法，並用 Applesoft 的 BASIC 給出一種正確的程序答案。此外，還說明如何按讀者自己的要求修改程序。宗旨是運用實例教授法，向讀者傳授程序設計的有效技術和良好風格。

本書供有進取心的初學者乃至高級程序員等不同程度的人員使用。書中各章提供一些工具和程序模塊（building block）供讀者以後編製程序使用。

這些模塊式資料，如第2章的輸入行編輯程序（input line editor），在第3章經擴充建成屏幕編輯程序（screen editor),等等，最後得到一組可供讀者日後編寫每個程序時使用的工具。憑藉這些工具編製的程序會更圓熟，更容易使用，節省編寫的時間，又便於需要時作修改。按照這種辦法，可達事半功倍之效。

我們特別對下列諸位致謝：

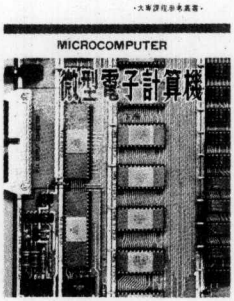
Carol Beal, Tom Bell, Zach Bovinette, Kathy Cukar, Takeshi Endo, Barb Odom, Vicki Porter, Jim Speir, Hal Tobin, Shelley Wright.

應用技術

科技英文縮寫詞典……

……李樹聲．梁兆楨主編

這本詞典蒐集了經常出現的科技工程詞目的縮寫體共約兩萬條，內容包括電子計算機、計測、機械工程、電機、電子工程、通訊、航空、宇宙等領域的技術用語，以及世界著名的學術組織機構、裝置、工程項目的名稱。



工業噴漆技術……趙國英編著
金屬染色技法……黃奇松編著
金屬電鍍工藝……林西音編著
鍍金技術……楊松堅編著
鍍銅技術……黃奇松編著
金屬眞空鍍……黃奇松編著
ＡＢＳ塑膠電鍍…黃奇松編著
塑膠工模技術……李科展著

工程繪圖……黃湛泉編著

本書爲大專課程參考叢書之一，書中圖文豐富，可以滿足工程繪圖的基本要求。文字說明簡潔，多採用立體化的輔助插圖以闡明投影原理，使讀者易以領會掌握。

本書既可作大專學校的課本，亦適合一般技工、技術員自修之需。

微型電子計算機（上、下册）

本書全面地介紹了微型電子計算機的工作原理，全書分上、下册。上册從微處理機的基本概念及發展談起，介紹一般結構及滙編程序。下册主要介紹存儲器、輸入輸出部分及中斷系統。每章除介紹一般特性外，還有Intel 18080和Motor la 6800兩種微處理機爲例進行講述外，並附有習題。

目 次

第1章　程序設計基礎

引　言

本書分成下列幾個題目：

- 數據輸入，第2章，
- 數據存儲及加工，第3～6章，
- 數據輸出，第7章，
- 綜合上述諸方法，編製一個總結性練習，第8章。

各章提供的子程序向你說明數據如何輸入電腦，如何存儲和加工，最後如何在屏幕或打印機印出。

最後一章，綜合所有這些方法，單獨編製一個私人日程表程序。

在最前一章，我們先討論子程序、程序設計風格和一些技術，以及有關編製和測試（testing）你本人程序的有效方法。我們也會提到以後各章用到的公共結構，幫助你從本書得到更大的收穫。

子程序（Subroutine）：意義和用法

簡言之，子程序指供一或多個程序反複使用的一段程序。其長度短至二、三行，長達數千行；但一般宜短，以便於理解和管理。理想的是一個子程序僅執行一項功能，例如只許從鍵盤（keyboard）輸入字母數字。僅執行單項功能，其結果是可以預期的，不致因出現異常反應而手足無措。倘若需要子程序執行多項功能，可由多個單一功能的子程序嵌套實現。

子程序同“正常”程序，區別在於子程序要使用 BASIC 的 GOSUB 和 RETURN 兩語句（statement）。GOSUB 用於“調用”程序（即主程序或出發程序）作訪問子程序，而 RETURN 用於被調用的子程序，作任務完成時的返回調用程序。沒有這兩個語句，子程序就是正常的 BASIC 程序。

GOSUB- RETURN 程序內每一 GOSUB 無條件地執行轉移到被引用的行號，一遇 RETURN，再控制返回緊接剛執行的 GOSUB 的下一個語句。

例

```
5000    GOSUB 6000
5100    PRINT X%*3
5200    END
6000    INPUT
6100    RETURN
RUN
?100
300
```

在此 GOSUB 例，一遇行5000，程序的執行序列就跳到行6000，請求輸入X %；由鍵盤輸入數 100 後（見 RUN 後一行上），於行 5100 恢復執行，將X % * 3 的答案（即 300 ）印出在屏幕上。

子程序的存在理由很簡單：令電腦盡可能地爲你多做一些工作。假如你的程序內有一功能需用到多次，那麼，或許你在每處使用，都輸入此功能的程序行；或者只打進一次，但在此功能程序行的最後，添一行 RETURN，以後每用到此功能，便使用一句 GOSUB。只打進命令 GOSUB，電腦就欣然爲你完成要求的全部工作，何樂而不爲？子程序還有一個宗旨是，保持程序的一致性。若一子程序在各程序都通用，其功能就完全按照同一的方式執行，無須記住每次在程序上使用它時的工作細節。

直觀上，我們可想像子程序不過是利用對一功能或動作的重複執行。對使用子程序執行的動作，往往可以放心，每次確實按同一方式執行。讓我們用如何發動汽車的例子來說明。現代的汽車，只要求你登車開匙。製造廠商爲你建立了一個“子程序”（實際上是一組嵌套

的子程序），一轉動鎖匙，就爲你服務。開匙動作啓動起一連串子程序，執行下面各項任務:

- ●決定是否要調節油嘴，
- ●打開油泵，
- ●試運行電器系統，
- ●開始發動引擎。

按此做，每次都有相同結果，除非系統出故障。

下面幾章，將提供可用於你的程序的一系列子程序。我們對這些子程序只給出幾種可能解法之一，需要時你按自己的要求作某些修改。

程序設計風格

我們想將電腦程序設計比喻爲一種藝術形式。正如一切的藝術形式，創作人具有某種“風格”，據此可決定其創作是否一件好作品或別的甚麼。我們一些人並非天生富有風格，也就是說，要盡可能向他人學習借鑑。風格隨時間而演變，我們大多數人每經一事會長一智。然而，甚少人回復到較早的作品，改進或更新其風格。這正是要求盡可能第一次就做好一項工作的重要原因。

本書的子程序反映了我們的風格，也許有些人喜歡，有些人並不，不過這是藝術。下面幾小節，將概括我們程序設計風格中某些要素，其中最重要就是那些有助於提高程序易讀性和理解程度。我們希望盡可能地堅持這些要素。人總有時會失誤，但願學了我們的風格，你能夠將中意的幾點，取作自己的風格。

有意義的變量名：

毫不奇怪，並非所有變量名都有含意。即使某變量名的含意你很清楚，但對讀你程序的其他人，也會完全被弄糊塗；部分原因在於，Applesoft 的BASIC 只識別變量名的前兩個字符。雖然BASIC僅考慮前兩個字符，但只要在程序中的變量名保持前兩個字符唯一，就不影響對長名字的使用。我們仍然按習慣，在必要時使用長名字定

義提到的變量。有時因照顧前兩字符的唯一性，變量名顯得有點特別，但仍然能夠辨明該變量的作用。

請記住，BASIC程序中的變量，對整個程序都有效，任何一個變量，可以在程序任何地方賦值，然後在任何其他地方使用此值。正是靠這種辦法和子程序交換信息。我們在調用子程序前，先要向它所使用的變量賦值。子程序完成其任務後，同樣這些變量，即使其中某些或許被子程序改變了值，仍然提供給程序的其他部分使用。

行號和子程序

我們在程序中使用不少子程序。既然一定要用行號不用標號（即名字）稱呼子程序，如何在我們心目中很有條理使用行號？我們不對子程序重新編號，一旦建立一個子程序，給其第一行定行號後，就保持不變。最後對整個程序作統一編號看來很整齊，但子程序要到處移動，統一編號就不值得。因此，需要的話僅對主程序重新編號，子程序的編號不變。

我們不應隨心所欲地使用行號，要有計劃。

第一，我們希望你能夠把我們一些程序段納入你現存的程序。鑑於大多數人喜歡用小值的行號，我們就選用大值行號，以免和你的程序發生衝突。

第二，我們用心選用行號，可以使程序的運行速度盡可能快些。BASIC 尋找一行號時（譬如 GOSUB 10000 或者 GOTO 11000），先和當前行號比較大小，若當前行號小，就從當前行號出發尋找所需的行；否則，從主存儲內第一個行號起尋找。

例如，考慮下面的程序：

```
100
.
.
.
1000    PRINT "HELLO"
1100    GOSUB 2000
1200    GOTO 1000
2000    PRINT "RANDY"
2100    RETURN
```

當 BASIC 執行行1100

```
GOSUB   2000
```

時，便從行1200起尋找行2000。但執行行1200時，就必須從主存儲的

```
GOTO   1000
```

第一行，即行 100 ，開始尋找。

倘若此例是一個大程序，從行 100 走到1000要費許多時間，因爲需查大量的行號；然而由行1100到行2000僅需檢查兩個行號，速度就快得多。正因爲 BASIC 這種特點，我們才考慮將一些子程序置於比 GOSUB 或 GOTO 所在行號大的行上。顯然，這方法並非所有場合都可以使用，不過是容易記住的一種約束。

注釋語句（Remark Statement ）

大多數程序員在其程序內不充分使用注釋（ REM）語句。

REM REM語句是不執行的行語句。在程序中，它用來提供一些注解和說明，供程序原作者及其他人員以後查閱時，了解程序的目的和方法。

例

```
1090   REM
1095   REM
1100   REM
1105   X = 11.005    REM  THIS IS A FIXED VALUE
1110   Y = 5.026     REM  THIS IS A FIXED VALUE
1115   REM
1120   REM
1125   REM
```

本例的行1090－1100以及1115－1125，用作突出其間隔的內容。這種辦法使程序更易閱讀,使你將注意力集中在行1105及1110上。這兩行跟着的注釋行分別指出這兩行的值爲何,從何來或作何用。在這種注釋行內，可隨便描述，只要能夠提醒你自己，這些行作什麼工作便可。

一旦程序的原作者走了，程序上又沒有注釋行，誰都不願意承接其修改工作，只好拋棄這個不寫注釋行的程序，由其他程序員從初稿

開始重寫。所以，使用注釋語句是一個好辦法，盡可能地爲你本人及以後的程序用戶提供更多幫助。

我們在本書中的程序，都用注釋行分隔其中的主要部分，清楚、詳細地說明如何工作及爲何工作。我們也盡可能地在子程序體內使用一些注釋行，說明走過的一段歷程，即使只用一組空白注釋行分隔上下文，也會增加程序的清晰性。

對於 GOTO 和 GOSUB 這些使用行號的程序分歧（branch），我們各用一注釋行說明程序的走向。我們也常常採用分歧，直到說明某一段程序工作內容的注釋行。

不管誰，職位如何，都會覺得調試程序是一件麻煩事情。倘若程序內有較多的注釋行，你就能夠更快地完成它，接着處理其他程序。

一行內多個語句

大多數 BASIC 語言版本，允許在一行上寫幾個程序語句，在它們之間通常用冒號分隔。Applesoft 也容許這種做法。一般來說，這方法並不可取，我們不採用，也不推薦，儘管因爲減少處理一些行號，確實提高了整個程序的運行速度。但如IF-THEN 這種語句，往往需在同一行上出現多個語句，這是例外情況。但是，行若很長，也許應該採用 GOSUB，轉變成子程序爲妙。我們一直提到，只有 REM 語句才可附在一行上，故只要合意，可隨心所欲地加進 REM語句。

編寫你的程序

從你寫程序起，就開始形成你本人的程序設計風格，它可能和我們這裏介紹的完全不同。然而，歸納你全部程序，應該得出兩個要點：程序應受用戶歡迎，應受得住測試。

受用戶歡迎的程序

受用戶歡迎的程序也受程序員歡迎。一個程序若容易理解，容易預期和使用，看來會受用戶歡迎。倘若程序滿足這些要求，程序員就會受用戶歡迎；否則，便不受歡迎。總之，你的程序越容易使用，大家就越樂於使用。

測試你的程序

一切程序在提交給用戶之前，都應該經過徹底的測試。測試工作很費時，程序越大，需要測試的變量和條件就越多。可是你一定得記住，程序受用戶歡迎的最基本要素是程序的正常運行，驗證一個程序工作是否正常的唯一辦法，是對它進行測試。我們坦白承認，在我們提交的程序中，用戶最終可能會發現還有錯誤。除非你用幾年功夫進行測試，否則休想找出你程序內部的全部錯誤，當然，你一定得盡可能把工作做徹底一些。倘若你分幾步測試一個程序，在開發過程，許多問題都可能會被發現，而且在出現問題之前可以得到改正。在你開發程序的過程中，最好讓其他人測試你的工作，這個主意挺有效。

令Apple 爲你服務

寫程序的根本目的，就是要利用電腦爲你做一些工作。在你設計程序的時候，要事先考慮到將來會出現的一些問題，並安排電腦自動處理。這個觀點把我們引回到關於程序模塊的概念上去。

我們在此書提出建立程序模塊，希望使Apple既爲程序員又爲用戶服務。程序員可以因反覆採用預製的程序片段而受惠，用戶也由於使用了一個協調而有專門性的程序而得益。

我們建議你購買一個軟件開發系統或者工具盒，市場上已經有多種此類產品銷售。其中對協助開發工作很有用的一種產品是 Apple DOS 工具盒（tool kit）。它向你提供多個程序，包括程序員輔助程序（programmer's aid），可以幫助你對程序進行重新編號和合併，刪除注釋語句，成爲一種變量交錯引用表（variable cross-reference table）；對於計劃編寫軟件的人員，這種服務性程序使用起來眞是得心應手。新的 Apple IIe 在碟片（diskette）上，附設一個合併程序和一個重新編號程序，但缺少其他服務性程序。

在下面幾小節，給出 Apple 爲你服務的其他一些提示。

查閱參考手册

你閱讀下一章之前，最好先查閱Applesoft Basic Programming

Reference Manual（Applesoft Basic 程序設計參考手册）中，內容關於某些有用和有益的建議的兩節。下面我們對其中一些作適當解釋。

首先，請看附錄D，“空間節省法”，下面列舉一些有關的提示：

●提示1.“每行使用多個語句”。這辦法並不好。對你或任何其他人來說，程序的易讀性因此受影響，給程序的調試帶來更多困難。

●提示2.“刪去全部REM 語句”。程序完全調試好之後，這才是一個好主意。

●提示3.“盡可能用整型代替實型數組”。這是一個極好的主意。

●提示4.“用變量代替常量”。又是一個很好的主意。

●提示6.“重複使用相同的變量”。這辦法可能有危險。你若一定要這樣做，那麼應規定一些字母的組合，來表示“廢物”（garbage）變量，而且在整個程序設計過程，始終使用相同的一些。

●提示9. 使用X＝FRE(0)清除舊的字符串，這是好事情，請記住按此做。

其次，閱讀附錄E，“提高你的程序速度”。下列其中一些提示：

●提示1.“用變量代替常量”。

●提示2.“把最常用的變量置於程序的頂部。”

●提示4.“常引用的行號應盡早在程序上定位。”

這三點提示尚需强調，因事情雖小，影響却大。

請注意：Apple IIe 的買主在其手册上查閱不到這幾點提示。就此而言，新版本並非一定更好。

Apple II電腦家族的一些差異

Apple IIe 僅對80列插件板作了一些必要的修改，故本書適用一切Apple II用戶。我們的全部例子以及測試程序都限定爲40列屏幕的格式，同時亦適用於擴充的 IIe 80列插件板。兩種裝置都適用，意味着我們需要在程序中作某些統一的處理。FLASH命令是一個明顯的例子，對你的菜單和屏幕起支配作用，但對於擴充的80列插件板就不合用。倘若你正在使用40列，就可以用INVERSE的同樣辦法（見下一章）處理FLASH。

80列插件板亦有開關控制，接通與否可在程序控制下檢查，即窺視（PEEKing）主存儲地址C300內前10個字節，並和80列插件板ROM 的前10個字節（byte）比較。

也許80列插件板的最可取之處，在於可以採用小寫和大寫字母，向你提供很美觀的屏幕。假如市場有出售，就值得推薦。

關於這本書的結構

本書可堪學習的經驗概括，就是所提供的程序。儘管各人情況不同，也許閱讀此書最容易的方法是，走馬看花地翻閱各章內容，得到一個印象，然後詳細逐個研究，其中的程序如何建立（請回顧前言上提過的實例教授法）。劃出程序的流程圖很有幫助。再一次更詳細閱讀文中的內容，然後輸入程序，並在每個測試點驗證你的工作進程。下一步啓動你的調試（debugging）程序，糾正輸入程序時的出錯。待調試完畢，你就會對程序有一個深刻的印象。在你覺得滿意離開之前，請勿忘記在你的碟片上作好後備（back up）。

以後幾章分成設計、用戶特點、程序員特點等部分，希望通過這些部分幫助你組織好思路，提高編寫程序的效率。以下是我們提出的幾點：

設計　　確定基本的程序功能。

用戶特點　確定具體的用戶功能以及會遇到的一些特殊情況。

程序員特點　確定必需滿足設計標準的某些具體特性。

書中每遇到新用的BASIC命令，都將給出一些扼要的解釋和例子。這些例子僅僅起提醒你的作用，因爲根據要求，假定你查閱過Applesoft II 參考手册。

下面各章列出的程序格式，不能夠原封不動照搬到Apple II。爲便於閱讀，我們（通過採用我們的字加工程序）已經將注釋語句修飾過。

其中幾章建立在其他章的基礎上。例如，第2章的行編輯程序，結合第3章補充的內容，便得到屏幕編輯程序。下表說明各章如何結