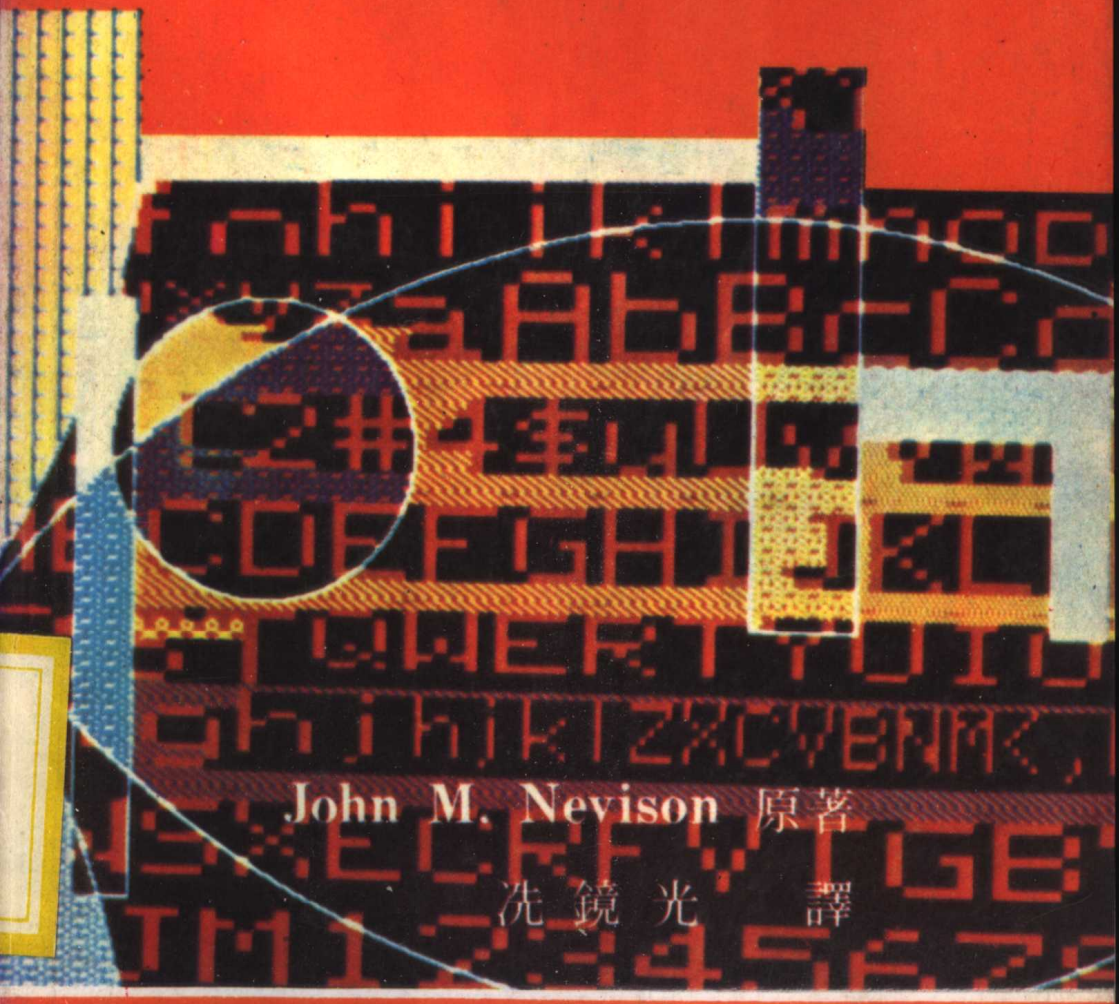
BASIC 程式格局

如何寫一個易於閱讀的程式



John M. Nevison 原著

冼鏡光 譯

電腦語言中心出版

BASIC 程式格局

如何寫一個易於閱讀的程式

冼鏡光　譯

電腦語言中心出版

BASIC 程式格局

編譯者：冼　鏡　光
出版者：
發行者：電腦語言中心
九龍彩虹道400號六樓
印刷者：合興隆印刷公司
香港仔宏利工業大廈七樓

定價港幣・$ 12.00

譯者前言

因爲工作上的關係，自己常常要製作相當大的程式，也要閱讀別人的大程式．但是相信很多人都有同感，也就是說看別人的不如自己寫，的確是如此，不過如果程式相當大，那末自己重新寫過就恐怕不可行了，因此大多數的朋友都有這個“痛苦”的經驗，尤其是看學生們的那些系統程式或編譯程式的報告時更是如此。因此，我常想，爲什麼大家不願意把程式寫得容易讀呢？也許理由很多，如果把學生除外（他們也許沒有那末多的時間），我想“私心”可能是一個很重要的因素；好比說，今年夏天，在麻省理工學院遇到一位比利時的社會學教授，在閒聊時談到一些軟體製作的問題，他說他自己有一部TRS-80.寫了很多有關自己研究範圍的程式，於是順手就拿出來厚厚一份報表，我的天！密密麻麻的BASIC 語句，我問他是怎麼看的，他抓抓頭苦笑，說是有時自己也會弄糊塗，那末爲什麼不寫得清楚一點呢？道理很簡單——別人也看不懂！我想這是很多程式員的心態，當然是好是壞不那末容易有個定評，然而自己總是不能避免閱讀自己的程式的，因此何必虐待自己！如果有保密的必要，辦法多得是呢！基於這個理由，我譯了這本書，不但是因爲它說到程式格局的問題，它還專門只說BASIC 的程式格局——BASIC 正是目前國內最容易接觸到的程式語言（不論是大型、迷你，或者是微型電腦都是如此），希望它拋磚引玉，在不久的將來會有更多這方面的專著出現，讓大家正視、了解這個問題的重要性。

原　序

這本書是爲那些希望能夠製作更好的BASIC程式的朋友而寫的，這類型的朋友範圍異常地大，從高中同學到科學研究人員，從大機構的經理到私人小型機器的玩意兒都可以涵蓋在內。William Strunk 曾經寫過了一本書，討論到如何寫作清晰的英文，後來E.B. White 把這本書加以修訂，重新出版，這就是有名的“The Elements of Style”。White說，在出版這本名著時Strunck 很謙虛地說，因爲該書很簡潔，所以說那是一本“小書”（a little book），事實上本書又何嘗不是如此？

本書雖然不厚，但是我希望足夠地充實。書中的每一條規則都應該一讀再讀；也許您可以很快地把它讀一遍，但若您想要跟著這些規則所說的來寫程式，您將會發現常常地您得要翻回來參考它們。

爲什麼要有規則？

也許您要問“爲什麼要跟著什麼規則做事？”，其實答案却非常簡單：因爲它們能幫您寫出更好的程式。所謂更好的程式指的是看起來更正確，容易閱讀，容易使用的程式；多年以來，人們覺得流程圖（flow chart）是解釋程式的一項有力工具，但今天，有顯著的證據告訴我們，事實上並不然（見書後的參考文獻：Shneiderman，1977）；然而，一套好的程式格局（有時候我譯成風格，格，……等等，其實那就是style這一個字——譯者）却可以，因爲它把程式中應有的結構直接地放到程式裏頭。

另外一個更進一步使用這套規則的理由是它們提供了走向“結構式程式製作”的一條路，如果您仔細地研究一下，您不難發現一個熟習這一套規則以製作良好格局BASIC程式的人，很容易就可以用其它的程式語言來寫作結構良好的程式。當然，BASIC（指的是“標準”BASIC——譯者）誠然不能說

是一套能做結構式程式製作的語言，但是良好的格局却是走向結構式程式製作的第一步。

書中所講的規則在製作程式方面給寫作者很大的發揮範圍，而付上的例子告訴您作者個人所喜愛的解法，當然這絕對不是唯一的寫法。好比說，後面我們將會講到如何把某些程式語句向右邊縮進去若干格，而究竟要多少格，那是讀者個人的喜好，限定不得的。另外一個就是利用REM－5 的寫法來區別註解與程式語句了，這樣的寫法也只是一個特例而已，但是那一條規則却絕不含混：註解必須要與程式語句區分開來。

新手與老手

您只要製作、執行過程式，那就相當於在計算機園地中寫過文章，但要寫得通暢、流利，那就需要時間與磨鍊了；而有關書寫格局方面的規則却可以幫助您縮短以完成一個正確、清晰程式的時間。

沒有人可以輕而易舉地得到一套良好的程式製作技巧，不過，任何寫過些程式的人也許會發現本書所談到的規則令人煩厭，這是免不了的；好比說，您會覺得其中的某幾條很煩，很不好用，比如有關打程式的規則就是一例；還有一些可能與那些老手的好格局意念相衝突，有一些却又看起來好像沒什麼，不值一讀，然而，如果您已經有了不少有關這方面的經驗而來批評本書的規則的話，也許這些規則就沒多大用處了；不過，反過來說，任何有關某條規則的批評都應該就它所建議的方面有所建設性才對，因爲這說明了也許這是個更好的方法也說不定。

適當地使用空白列與向右移若干格的寫法是本書的中心論題，這兩個技巧會戲劇性地增加程式的可讀性，但却需要在打程式時多所留神；也許，有些讀者會想到，如果以規則3，16，17，18 爲準來寫一個程式，幫我們“自動”排好程式語句豈不大妙？正是，第六章討論的就是這樣的一個程式。

標準BASIC

接著我們要談談所謂“標準”BASIC。美國國家標準協會（American National Standard Institute，簡稱ANSI）除了定出了ANSI FORTRAN 與ANSI COBOL 之外，還定出了標準版的ANSI BASIC，一般都叫做最小BASIC（Minimal BASIC），它是很多以後加以擴張、加上好像字串處理，畫圖能力的各個BASIC的核心部份。不過，這套標準BASIC有個嚴重的缺憾，它不接受空白列，好在不少好一點的BASIC編譯程式都放寬了這條限制，然而還是有些BASIC不可以，所以本書規則3中的REM-5寫法的用意就是讓您在此嚴格的規定下還能夠“模擬”空白列以寫出清晰可讀的程式，使用REM-5寫法，本書中的每一程式都可以轉寫成標準版的BASIC。

目前還有另一些BASIC系統規定所有語句必須要左邊對齊，這個對齊的觀念是廢話，因爲它毀掉任何想要向右邊縮進幾格的寫法，也毀掉了程式的可讀與清晰性，好在這還可以（雖然並非完全地）克服，您不妨試用第六章的程式。

長不過一頁的程式

本書的程式都限定在報表紙一頁之內，這型較大的程式使您要注意到某些語言中的功能，特別是控制結構。如果您要寫較大的程式，您就應該學一點有關結構式程式製作的觀念，以及學習一套能夠讓您很容易這麼做的程式語言這些語言包括了Dartmouth學院的結構式BASIC，PL/I，COBOL，ALGOL與PASCAL 等。在結構式程式製作中，其中有一條重要的準則是：程式中任何一個單位長度不宜超過一頁，於是練習如何寫好小的BASIC程式，對於寫作這些程式單位來說正是一個好的練習機會。

第六章中還有一些爲那些胸懷大志要寫大程式的懶蟲而寫的建議，我們寫

出這些建議的原因在於有些讀者除了BASIC之外根本就沒有其它可作結構式程式製作的語言。

誌 謝

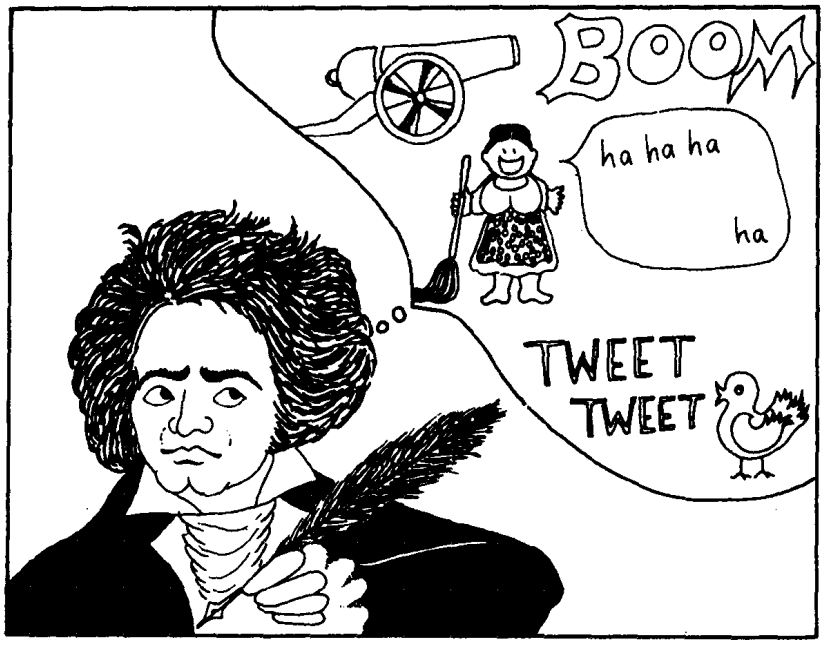
本書的程式是在CallData，Dartmouth學院兩地使用Dartmouth分時系統做的，而且本書的排版也是經由此系統完成，這兩套系統使本書得以及早出版。

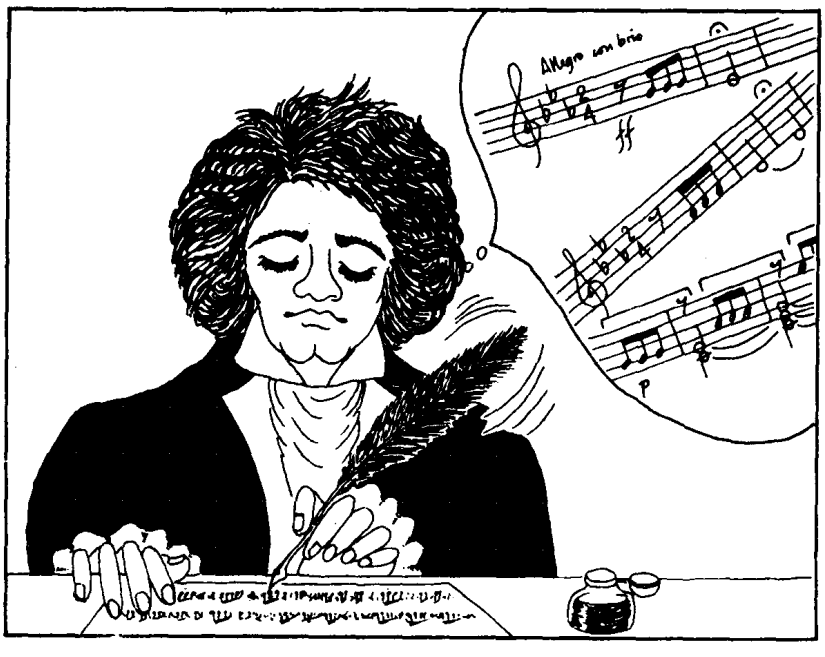
本書的作者要向在Dartmouth的Stephen Garland致謝，感謝他提供的一些特殊幫忙；還有，Timothy Stein爲我查對STYLIST，Stephen Wait 協助計算機排版，Nancy Farrell 與Linda Micheli 仔細的校對，編輯William Gruener 對本書的熱心鼓勵，最後是Nancy Ross McJennett 她身兼精密的職業設計者與寬容的家庭主婦兩職，均對本書有所幫助，作者一併向他（她）們致謝。當然，如果本書還有什麼錯誤，那末，這就是作者的不對了。

John M. Nevison

波士頓，1977

原著書名爲 The Little Book of BASIC, 2nd, Addison-Wesley, 1978


BOOM
ha ha ha
ha
TWEET
TWEET


Allegro con brio
ff
p

目　次

第一章

從解題到程式製作

2 BASIC程式格局

在西方語言的早期發展史中，希臘人發明了所謂的“耕牛書法”，寫起來活像牛耕田，第一列自左而右寫，寫完之後第二列接下來自右而左，第三列又是自左而右，……當然這種寫法對眼睛可是大大地有好處，但是對讀者而言却未必，因爲他要兩種閱讀法：正讀與反讀是也。之後，希臘人簡化了他們的系統，使得每一列都自左而右讀，後來，他們又把某些列縮到右邊一點，讓眼睛有個休息的時候，於是“段”就產生了；自此而後，就有不少的化簡以便於作者與讀者之間的溝通。

說到計算機程式語言，大都還是在孩提的階段，還有很多關於程式在作者與讀者之間交通的東西值得學習；BASIC到今天十五歲了，它易學好寫，但是它寫成的程式是否易讀却就決定於您如何寫它了。有些BASIC程式寫起來就活像耕牛式書法，而其它的却是流暢得很，因此本書的目的就在於幫助您，教您如何製作一個正確而且易讀的程式。

縱使是從來不示人一列程式語句的傢伙也會同意：一個程式不但要能正確地做出結果，還要能夠容易閱讀；這道理不難，雖然他不把自己的程式給別人看，但是他却得讀自己的程式，而他自己比之於其他的人不一定會強到那兒去，他也會犯錯，也會把程式研讀若干次之後才能把錯誤除去，所以說，如果他寫的程式可讀性高，那末就會節省了他自己的時間。

除此之外，當我們窩在自己的天地裡依本書的規則製作程式，到有一天我們把自己的程式拿出來公諸同好後，我們都會發現彼此間的了解並沒有什麼困難。在我們所寫的那末多程式中，也許總會有那末一兩個程式就清晰這一點上給別人的印象非常深刻，我們可能會說，這些個程式有他們的“風格”，或者說它們有它們的“格局”（style）更適些罷！

解　題

不管是誰在寫一個程式，他總可以說是在解一些問題，這些問題範圍很廣，大到說如何讓計算機做些於人類有益、有用的問題，小到只是爲讓某個人某件事做起來容易些等等不一而足。當然，程式設計師因著經驗的增加會使他自己解題的能力日增，但爲什麼會如此呢？似乎人類對此並沒有什麼充分的了解，恐怕解題這回事大部分還是神秘兮兮的緣故罷。

不過，目前倒是有幾個試驗的觀念嚐試說明上面的問題；在數學方面，提供了大量的、沒有現實世界那末多枝枝節節的障礙的問題，於是數學家George Polya 就以此爲出發點來探究解題的內涵；Polya 寫了一本書"怎樣解題"在這本書中他建議了一個一般性的處理方法：了解問題，擬定解題的計劃，執行計劃，檢討所得的解答。當然，當您要在計算機上來解決問題時，您必定能夠自 Polya 有系統的過程中學到不少東西。

從圖1．1中您可以看出，對問題的了解不但要了解它的正面，還要了解它的反面。就程式設計師來說，他應該先要問一下自己，他所處理的問題究竟要不要動用計算機，若您只對結果有興趣的話，往往用其它的方法會快得多；本書的許多問題事實上您用小計算機就可以完成，甚至於要排列九十個名字，您把它寫在卡片上來排也許還會好一點。

當您了解問題內涵，而且覺得非得動用計算機之後，您就得發展出一套解題的計劃了；請記住，當這個解題計劃完成之後您還是應該仔細地想想，看是否有另外更好的可能性，通常您會得到另一個較好的方法；請不要因爲增加了額外的修正工作時間而洩氣。

了解問題

什麼是未知量？有些什麼資料？有些什麼樣的條件？這些條件能滿足嗎？這些條件足以決定未知量嗎？或者是不夠？或者是不相關？還是根本

就相反呢？請繪出一張圖，引入一些適當的符號，把條件分成幾部份。您能寫得出來嗎？

定出一個計劃

以前見過嗎？或者您看過一個類似而只有一點兒不同的問題？您是否知道一些有關聯的問題？您是否知道一些對此問題可能有用的定理？請看看那些未知量！再試想一下您熟悉的問題中有沒有具有相同或相似的未知量的。

如果您已經解過了類似的問題，那末您可直接應用它嗎？您能用它的結果嗎？它的方法合用嗎？您可以引入某些輔助的工具使得它可以被應用嗎？您能換個方式來敍述這個問題嗎？還有沒有其他的敍述方法呢？

如果您不能解答這個問題，那末不妨首先試著去解一些相關的問題。您能想到一個關聯更大的問題嗎？或者是一個更廣義的問題？或者是一個更特殊的問題？或者是類似的問題？您能解決這個問題的某些部份嗎？把條件刪掉幾個，我們還可以求出未知量嗎？能求出多少？又是如何改變的？您能從資料上頭導出一些有用的結果嗎？您認爲其它的資料比較適合求出那些未知量嗎？您能改變未知量或資料或是兩者（如果需要的話）使得新未知量與新資料比較接近嗎？您是否使用到資料的全部？您是否用到了條件的全部？您是否已經考慮到問題中每一個重點呢？

執行計劃

接下來，要把您的解題計劃付諸實現，每一個步驟都要仔細地檢查，您能夠很明顯、清晰地看出這些步驟是正確的嗎？您能證明嗎？

研究你的解答

您能檢查您的結果嗎？您能解答問題中的論據嗎？您能用不同的方法

導出這個結果嗎？您能夠應用這個結果、方法到其他的問題嗎？

圖1・1　解題要訣

完成你的工作，構作程式

本書的主旨——教您寫一個易讀的程式——本身就是構作程式過程中的一個重要步驟，自然也是把工作用計算機處理的過程中的一環。圖1・2中把上面所說的兩件事分解開來說明，在圖1・2中最重要的顯示就是：計算機程式比它本身在功能上的美感就它的目的而言要來得大一些，因爲它要幫助某些人去做某些有用的工作，所以了解實際上工作的需要，以及那一部份要用計算機來運算就是程式設計師的第一件，也是最重要的一件工作。

程式設計師的第二件工作就是得隨時注意到因爲時間變動而產生不同的要求，把自己的程式加以修飾；不管是個什麼樣的解答，您不會是永遠地一成不變的，最後，這些變動會使得您程式結構上的改變，或者是根本把舊的程式作廢。如果您一開始就有這樣的準備，那末就會預先安排很多可能會出現的問題了。

問　題	準備在計算機上作業	構作程式
了解問題	了解要做的是什麼工作，而且以後可能會做什麼樣的改變。	了解程式要完成些什麼事。
擬定計劃	把人員與計算機程式組織起來，這個組織當然得要很有彈性，以適應以後隨時會變更的工作。	設計程式，把各個程式片斷組織起來，選擇適當的資料結構，解題的方法，計算機，程式語言等；再