C語言程式入門

莫天昇 譯

能夠掌握當代不論微型、小型，還是大型電腦結構共同特質的是C語言。這種“實作者的語言”，替小而又快的程式開創新機，使它們不必經過轉換便可在各種機器上執行。不論您具有程式設計的經驗與否，都可學會這種威力強大的語言，並應用到即時程式設計、訊號處理、電子工程、應用套裝軟體或複雜的個人計算等方面。

Thomas Plum 原著

C語言程式入門

莫天昇 譯

儒林圖書公司印行

C語言程式入門

原著發行日期：1983（First Edition）
原著書名：LEARNING TO PROGRAM IN C
原著者：Thomas Plum
譯者：莫天昇
發行人：楊鏡秋
出版者：儒林圖書有限公司
地址：台北市重慶南路一段111號
電話：3812302 3110883 3140111
郵政劃撥：106792號
吉豐印刷廠有限公司承印
板橋市三民路二段正隆巷46弄7號

行政院新聞局局版台業字第1492號
中華民國七十三年五月初版
定價新台幣 200 元正
26.00元

序

這是一本有關C程式設計語言的入門讀本。讀者不需先有程式設計的經驗，不過，熟悉別種程式設計語言的讀者可以在本書找到許多有用的說明，這些說明曾經加速許多有經驗的程式設計師學習C語言的過程。

因爲程式設計師大都關切可攜性與有效性，所以本書以通用目的電腦語言的方式處理C語言，不偏向任何特殊的應用領域；所選的例子以展現C語言的特質爲主。由於教學重於參考，因此不像編譯程式手册那樣詳細地討論語法。經過五年實地的教學才定下本書的結構。著者試圖在建立程式設計基礎（與語言及環境無關）與傳達程式館的細節、編譯程序、與軟體維護之間達到平衡，爲的是讓讀者在軟體工程實際的情況下，得到成爲勝任的程式設計師所需的資料。

除了技術性細節外，著者希望本書能使讀者樂於使用C語言。在各種當代的電腦中，C是程式設計時最佳的設計工具之一，它簡潔優雅，且在許多用途上具有威力。利用它，讀者可以建立不受電腦硬體限制而有效率的應用。當讀者在電腦專案中使用它時，或許會發現它在專業程式設計師之間廣受歡迎。

附錄A包括C語言的簡明指南及通常UNIX（和許多其他的）系統都會提供的程式館。不過本指南並非用來取代讀者的編譯程式經銷商所提供詳細的語法參考手册，而是撰寫可讀性C程式的規則摘要。

附錄B詳細地討論環境的問題，例如怎樣編譯程式，位元組與字組的大小，及編譯與機器相依的特質。此附錄也包括所有在正文出現的程式完整的列印，及大部份標有［問題］之關鍵性問題的答案。

標題爲［演習］的程式設計演習，是特地爲提升讀者的創造力而

設計的。很高興能在此感謝許多人士的鼓勵和協助。首先，特別感謝P.J. Plauger，不僅因爲他在製作本書所扮演的角色，而且由於他把C語言帶到許多不同的角落。事實上，除了當時尚未料到有出書的必要外，他可算是本書的共同作者。本書所出現“我們”或“我們的”等字眼並非“自稱”而已——我在將來的版本能夠和他合作的希望下保留這些字眼。同樣地，本書中“我們”經常與Plum Hall Inc. 研討會的講師有關，尤其是Steve Schustack和David Graham兩人，爲了澄清語言表達的方式，特別舉出許多明顯的例子。還有許多人士在本書的草稿上提供寶貴的意見，包括C. Gelber, John Justice, Brian Kernignan, Ian MacLead, Ed Rathje, Greg Rose, Teri Whynman, 和家母，Ruby Schunior.當然，本人的感謝並不表示他們完全同意本書的表達方式——從編寫，設計程式，測試，到選定字形都由著者一手包辦，無人查驗是否還有缺失。

由於家人堅定不移的支持與鼓勵，我衷心地感謝——Halls, Prichetts,Richters, Schuniors, Schanzes, 和Whynmans。

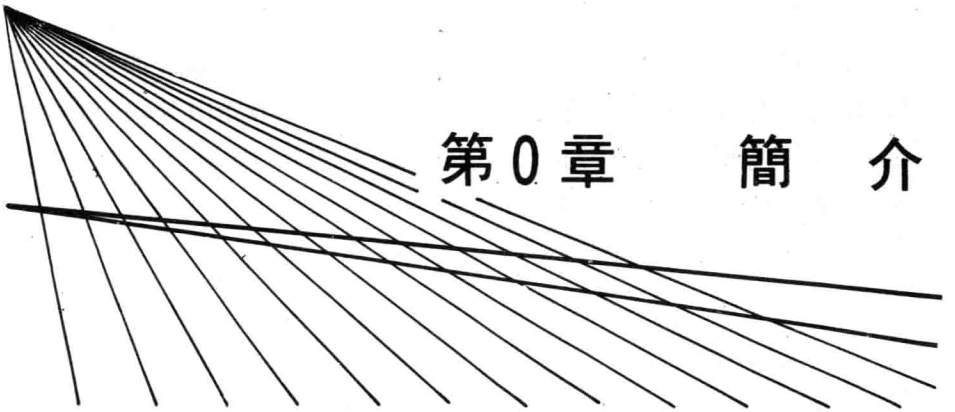
特別感謝Dave Plauger, Tana Plauger,和Whitesmiths 公司其餘的同仁。Don Bolinger所寫的本文格式程式ctext使本書的打字稿得以完成。而Whitesmiths堅強的編譯程式，讓著者在三種結構不同的機器執行測試時，不需修改任何程式。在本書的製作階段，他們的Idris 作業系統提供令人信賴的支援。同樣地感謝VenturCom Inc.,因爲本書經常利用UNIX的Venix版本。

C語言的創造者，貝爾實驗室的Dennis Ritchie,值得所有有使用C語言的人們讚揚。我們特別強調一點，C不是由任何委員會設計出來的。著者很欣慰地感謝Plum Hall卓越的同仁，Sonya Whynman和Linda Deutsch,因爲他們的努力使本書得以呈現在各位面前。

而最感謝的是內人Joan Hall，她是Plum Hall 的董事長，也是讓著者由助教授變成優秀的講師最大功臣。

Thomas Plum

目 錄



第0章 簡 介

0.1 導 論

C語言迷的特點是把電腦當成人人觸手可及的硬體。前不久，大約一九七六年左右，有人預測電腦的形式會變得愈來愈集中，散佈各地的終端機向巨大的電腦中心存取資料。微電腦和迷你電腦下跌的價格反而造成完全不同的結果。通常電腦使用者都清楚他們所用的機器，記憶體有多大，處理速度有多快；大多數的辦公室把電腦視爲生活要素中的機器（與遙控抽象的觀念相反），甚至許多家庭也是如此。這種情況對像C之類的語言有利，因爲C語言以優美的方式嵌入許多當代計算機共通的特質。

的確，我們所訓練的許多人士，他們的職位並非程式設計師，而是工程師，在具有電路和處理機的知識下來學C語言。對他們而言，C語言變成他們工程工具套件中另一項重要的工具。

對C有興趣的另一類團體是系統程式設計師，即設計程式供其他程式設計師使用的專家——我們會在下一節討論編譯程式及作業系統。這類工作，有效性——使程式快且小——是主要的考慮。通常，也包括出售的可攜性程式，可以在各種不同的機器上執行。

同樣地，也有程式設計師撰寫應用套裝程式，能夠配合有特殊需要的使用者所使用的電腦。再次，C提供有效性與可攜性的組合使嚴

謹的軟體製造商認爲它很理想。

所有C的使用者由於它是配合計算機特質的小型語言而獲益。一開始C便具有這些優點。它從B語言演化而來，B是由Ken Thompson 在一九七〇年根據Martin Richard 的BCPL修改而成的。UNIX 作業系統原來是用PDP-11的組合語言撰寫，在發明C之後，很快地於一九七二年用C改寫。改寫的結果造成壓倒性的肯定，C馬上取代組合語言在UNIX環境中的地位（C完整的歷史可參考Ritchie et. al. [1978]）。

在它大部份的歷史中，C已經完全與UNIX作業系統結合在一起，UNIX 是它的老家，也是它第一個主要的應用。許多對C的興趣來自UNIX的盛行，然而，此種關連的歷史性大於實質性，最近的趨勢已讓C語言展示它獨立的地位。許多獨立的廠商已經生產一系列非UNIX環境的編譯程式，例如與貝爾實驗一樣著名的Whitesmiths, Ltd. 的確，加上許多玩家對語言的興趣，八十年代早期是C語言微處理機版本昌盛的時代。

我們把C稱做當代電腦的“可攜性組合程式”，是一種愼重的矛盾修飾法。“可攜性”，意指同一個程式可在各種不同的電腦上執行；“組合程式”，意指程式小而快，且能完全掌握機器內的資料。所以，C是否適合某項應用的簡單準繩，是下列兩道問題的答案是否肯定：“是否考慮使用組合語言？”與“程式需要在不同的硬體上執行嗎？”

雖然C的用途廣泛，不過我們不採取“食譜”的方式研究程式設計。讓讀者清楚地了解語言與它在電腦中的作用，是我們主要的目標。選取的程式實例與演習並非由於它們有用－雖然有些很有用－只是爲了展現C語言的特徵。主題和說明的順序是由幾年來專業化的C語言教學演化而成。我們訓練的人士當中，有許多位第一個重要的程式設計專案是以C語言完成，所以本書不僅涵蓋程式設計的基礎，同時也收錄C語言的細節。本書的某些句子也許稍後在其他地方引用，在

表達的方式上，我們儘量避免“半眞半假”的事情在稍後才解釋清楚，但是某些主題（例如函式的分離編譯）會完全地擱置，直到所有的片段能夠合在一起。

所以，本書的組織如下。第一章（“電腦和C”）描寫當代電腦的特徵。在嚴謹的介紹方法下，我們只假設你對電腦有些認識，毫無疑問地，有經驗的電腦使用者可以略讀本章。

第二章（“資料”）提供位元，二進位，八進位，十六進位，和字元碼的介紹（或複習），並詳細地討論純量變數的宣告，（本書不涵蓋最近某些編譯程式提供的 enum 型式）且定義程式的“元素”。再介紹指定敍述，控制流程，輸出，程式大小，#define, 及#include；提供程式實例的基礎。

第三章（“運算子”）是最長的一章，涵蓋整個C語言大部份的運算子。先說明字元的輸入／輸出，好讓讀者和程式交談。全章都在描述優先次序及結合律，最後還有摘要。用在 scanf 函式的位址運算子，替稍後討論的陣列與指標舖路，並提出可攜性定義型態的方案。以和輸入與輸出有關的環境資料做結尾（開啓和關閉名稱檔案，及處理各種庫存函式，不在本書的範圍內。）。

第四章（“敍述與控制流程”）完整而詳細地涵蓋C所有的控制結構。語法與可讀性，這兩項主題都描述。本章介紹控制結構的設計，簡化可讀性程式所須注意的規畫。

第五章（“函式”）涵蓋C函式所有的語法層面。描述分離式編譯與連結，和擴充的語法檢查命令 lint。二維陣列延到本章，如此才可同時討論它們的初始化。

第六章（“軟體發展”）討論分析，設計，實作，和維護。包括實例研究問題（“黑傑克”）整個細節。

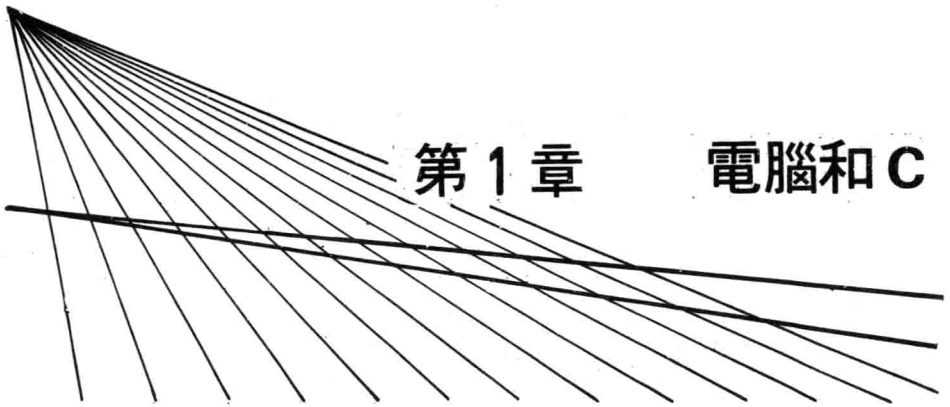
第七章（“指標”）涵蓋C的基本指標。高等應用－動態儲存體分配，串列，和函式指標－並不包括在這本入門讀本。

第八章（“結構”）涵蓋C語言的 struct。不包括 union，自我

參考結構，和位元欄。

附錄A構成C的“口袋式指南”，摘錄可讀的語法規則與最常用的C程式館。還包括 printf 和 scanf的格式，常犯的錯失，C“成語”，與 ASCII 碼。

附錄B包含本文中所有關鍵性問題的答案，與程式列印及和環境有關的注意事項。作者嚴謹地選取這些結構，好讓編譯程式製造商以簡單的架構，描述他們編譯程式的差異和特質。



第1章　電腦和C

在我們開始研究程式設計之前，先談談我們所用的電腦一般的特徵。以C設計程式的電腦有許多共同的特質。爲了避免表達的方式偏重某種特別的機器，我們從一架名叫 Igor 的電腦故事談起，這架電腦的工作與Transylvanian希爾頓大飯店的員工類似。

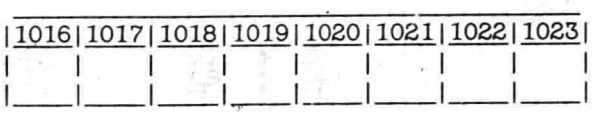
1.1　Igor 和數字遊戲

Igor 和他儲存的機構，組成我們的電腦系統。

1. 系統的主記憶體（或儲存體），由大小相同的編號容器組成。

在Transylvanian Hilton 中，儲存體嵌在牆壁的架格上，每格有一張細長的紙，而且每格都標上號碼。旅館恰巧有1024 個插孔，電腦行話爲 1 K；插孔號碼由 0 排到 1023。插孔看起來如下：

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	20	21	22	23	24	25



2. 輸入：來自外界的資訊可以進入插孔，或者透過 Igor，或者由某人直接放入插孔。輸出：複製紙上的資訊然後送給某人的方式，可以把資訊傳給外界。同樣地，Igor能做這件事，或者由其他人直接做。

   如果 Igor 他自己執行所有的輸入和輸出，可使系統較簡單，但是萬一他非常忙碌，他或許該找位助手幫他做複製的工作。

3. Igor 具有一些預先設定的能力。

   我們剛剛提過，他能把資訊放入插孔，也能從插孔抄出。除此之外，還能將資訊由某個插孔複製到另外一個插孔；執行簡易運算如加、減、乘、除。例如，他能將兩個插孔的數值抄出，加在一起後，把結果存回插孔。Igor預先設定的能力就是他的硬體——因爲無法改變，所以叫做“硬體”。

4. 雖然 Igor 只能完成少數的運算，但是執行起來又快又可靠。

   重要地是即使 Igor 整天甚至整月的工作都不會出任何差錯。可靠性是系統能否適當運轉的重要因素。

5. 部份儲存體專供處理機的指令使用。

   經理發覺，將某段不用儲存資訊的插孔，拿來儲存 Igor 的指令非常有用。在執行某個插孔內的指令後，Igor便到下個順序的插孔執行下個指令。同時，插孔可能放著指示 Igor 到別個插孔的指令。每天早上 Igor 一到，就到指定的插孔找出指令，然後開始工作。完整的指令集合構成一個程式，Igor 程式的集合構成他的軟體——因爲改起來相當容易，所以叫做“軟體”。

   有了以上這套能力，我們能爲 Igor 描述一個簡單的程式：結算

顧客的帳單。

插孔	內　　容
0	將插孔 9 抄入插孔 13。
1	將插孔 10 複製給會計室。
2	將插孔 11 複製給會計室。
3	從會計室得到的輸入，抄入插孔 12。
4	如果沒有輸入的話，到插孔 7 執行指令。
5	將插孔 12 加到插孔 13。
6	到插孔 3 執行指令。
7	將插孔 13 複製給顧客。
8	到插孔 14 執行指令。
9	"0.00"
10	"Please send the charges for room number: "
11	"123"
12	
13	
14	

現在我們來個“數字遊戲”。像上面的寫法，指令又臭又長。不過有辦法簡化，將 Igor 的指令編號，然後用更緊湊的形式書寫。

6. 處理機預先設定的能力，以數值化的指令碼識別。

以下是 Igor 的一些指令碼：