I.R. Wilson
A.M. Addyman

A Practical Introduction to Pascal入門

吳聰明校訂
譚巽言　譯

雲陽出版社

I.R. Wilson
A.M. Addyman

A Practical Introduction to Pascal 入門

吳聰明校訂
譚巽言　譯

雲陽出版社

PASCAL　入門

210

作　者　譚翼言譯・吳聰明校訂
出版者　儒陽出版社
台北市光復南路17巷46號
台北郵政信箱36-60
7629705　7610482
登記證局版台業字第0908號
發行人　陳文惠
台北市光復南路17巷46號
7629705　7610482
印刷者　建利印刷廠
台北市東園街260巷25號
3071088
基　價　貳圓捌角
版　次　中華民國72年4月初版

譯者序

隨著科學進步、工商發展，近年來電子計算機之應用，已在我國日趨廣泛爲大衆使用的工具，無論在商業上，工業上及日常生活中皆是如此。然而在程式語言方面，PASCAL語言已逐漸爲大家所接受，而普遍成爲各級學校使用的教學語言，此乃因爲PASCAL語言本身結構非常簡單，且富於變化，它可以做爲各種資料結構型態的說明範例；並且很容易就能瞭解它，而利用它來處理問題，解答問題。過去一般大專學校皆以FORTRAN程式語言作爲必修課程，自PASCAL語言爲大衆所普遍接受之後，PASCAL語言已取代FORTRAN程式語言，而成爲大專學校的必修課程。市面上雖已有中文的PASCAL語言書籍，但大都不能作爲初學者了解PASCAL語言輪廓的介紹性讀本；譯者鑑於以上各點，認爲本書很適合作爲初學者了解PASCAL語言基本結構的讀本，特將此書譯成中文，以便利更多的同學能夠吸收運用。

本書共分十四章，對PASCAL程式之基本格式、流程控制、資料類別，及輸入、輸出方式，以及較複雜的結構化資料類別等，都有非常淺顯的介紹，使讀者對PASCAL語言具有基本的認識與了解。讀完本書之後，若想更進一步瞭解PASCAL語言，則可以選用K. Jensen及N. Wirth所著的Pascal-User Manual and Report作爲參考讀本，它對PASCAL語言有更深入的說明與介紹。希望本書對同學們在了解PASCAL語言方面，能有所幫助；翻譯此書因時間

倉促，恐有未盡理想之處，尙請諸位先進指正，此亦爲譯者所盼望的。

本書能順利出版，要感謝國立台北工專電子科教授胡熙慶老師，給予譯者莫大的鼓勵與支持；更感謝講師錢文南先生及助教吳聰明先生對於本書所做的校訂。同時感謝致理商專劉會南小姐幫忙整理原稿，並將本書中所有例題程式重新打字。

譯者 譚巽言 謹誌

民國71年11月19日於台北

原序

經由Addyman 自身的觀察證明，在整個歐洲和美洲的社會事業機構，PASCAL 已經迅速地增長成爲普遍的教學語言（Comput. Bull., Series 2,8, June 1976,31）。乃是基於兩種原因，其一是因這種語言它本身的特點，其二是因有一個效能很高且毫不費力的編譯器。至於後者，作者曾經對於整個CDC CYBER 編譯器做過實例調查，發現它的速率乃是FORTRAN編譯器的1.8倍，也就是PASCAL編譯器做了1.8次，而FORTRAN編譯器才做完。

這些語言和編譯器的特點，也得視系統程式設計者和微處理機之使用者的幫助與否而定。關於使用者的範圍來說，作者相信PASCAL將會取代BASIC 程式語言。

可以確定的，在曼徹斯特大學的計算機科學系裏的大學生們大半以上是用PASCAL 來寫作程式。還有在曼徹斯特曾經有過一個關於基本的程式技巧方面之介紹性的演講課程，然而本書已以此爲根據做爲基礎；除了那些演講之外，課本的內容還包括兩類實用練習的組合。其一乃是針對簡短的，由筆和紙即可寫作的習題之解答所開的會議；其二乃是需要學生們寫一些完整的程式，並且在線上作業的終端機上用“編輯同時執行”的方式來執行這些程式。因此本書的每一章之結束都有一些適當的習題及問題就是爲了這些目的。雖然這些習題和問題的解答，並沒有全部出現在課本中，不過教授先生們如果需要，可以要求作者而得到這些完整的解答。

關於例題的應用，完整的PASCAL 語言是用實用的形態而表示出來。的確，解答一個問題所用之資料結構的設計的重要性和問題本身的結構是一樣的，因此對於這個部分，提供了七章以便能夠

充分的認識與了解。希望本書能用來做爲學生和許多有經驗的程式設計者以及尚未熟知PASCAL 者之最初的學習課程。

本書的目的乃是提供程式的介紹，並不扮演最後的參考工作；後者的工作乃是由K. Jensen 及N. Wirth ，所著之 Pascal- User Manual and Report (Springer- Verlag, New York, 1975) 來擔任之。

逐步的改善設計和發展程式的技巧，此兩者都是非常有用的建議，但無論如何在組合方面明確的圖解表示法（如結構圖，樹狀順序圖表，流程圖，等）在本書中是不被排除的。

作者自認接受同事的幫助與鼓勵，尤其是C. C. Kirkham博士，J. Rushby博士及F. H. Sumner 教授，在此向他們致謝；同時也要謝謝J. Scrivens 小姐的耐心打字及Southampton 大學的D. Barron 教授之細心校閱打字原稿。

在本書中所出現的文法結構順序圖表，是在曼徹斯特大學的計算機繪圖單位的幫助之下而繪製的；而程式例題乃是用連接於MU5研究用的計算機之DIABLO 印字機印出來的。

曼徹斯特大學　　I. R. WILSON
1978 年 3月　　A. M. ADDYMAN

目　　錄

第一章

緒論

計算機是一種能夠在高速（百萬分之一秒）而且可靠的情況下，作準確的計算行爲及資料或訊息之動作的電子機器。它可以重複執行本身所儲存及瞭解的"指令"（*instructions*）所組合而成的工作。這些指令爲一些關於計算機本身記憶體內的數字之計算，及搬移或將數字儲存至記憶體內之動作；還有一個就是指示著下一個要被執行之指令的動作。因此，對於需要得到比較正確的答案之問題解答，包括龐大的數量計算或處理，計算機可以說是最恰當的工具了。至於其用途的變化很大，從飛機跑道的計算到銀行的自動業務處理，都在範圍之內。

在計算機上，插入一羣代表問題之"資料"的數據及一些適當組合的計算機指令，來解答問題是有此可能的；不管怎麼說，這都是一種另人生厭而且易於錯誤的工作。在計算機上作問題的解答，是使用許多程式語言中之一種語言，來配製出問題的解答。爲了避免英語缺乏精確及有著兩種或兩種以上的含義，使用的人允許解答是非常簡短而且明白清晰地被指定及載明。例如："time flies"是一個註釋或一個命令呢？所配製出的解答，稱之爲"程式"（*program*）及所配製出的動作，稱之爲"程式動作"（*programming*）。一個完美的程式，應該在解答中能夠反映出問題的性質及解答中所包括的步驟；這些步驟，被稱爲"結構化"（*algorithm*），在本

書中提到結構化的表示式是使用 PASCAL 語言。

PASCAL是在1970年，由瑞士之蘇黎克人 Professor Niklaus Wirth 所發明的，其命名是在最早發明實用的計算器之著名的數學家 Blaise Pascal之後。此種語言允許一個程式解答之結構及細節是用能被處理的資料或訊息及能被執行的動作之名詞來表示之。的確，很容易地就確定了解答的結構，並用 PASCAL 來作出問題的解答，和使用"舊有的"計算機語言，如 FORTRAN，ALGOL 及 BASIC 比較起來，是更簡單及更上等一點。

通常，使用計算機之處理來解答一個問題時，會包括一些組織結構的問題，下面有七個步驟必須加以辨識清楚。

(1)瞭解問題的性質，有那些可以利用的資料或訊息，所需要產生的資料或訊息及有那些假設需要設定；詳細的查詢工作應該在前述的行動之前就必須完成。要利用到的資料或訊息將會構成"輸入"或"資料"而進入程式中，至於資料的格式，單位，所限制的數值及大小也都將會影響到程式的設計。所產生的資料或訊息則將構成"輸出"而離開程式。還有一點必須注意的，要保證所給予之其所需要的資料或訊息的項目，其格式及單位爲其所需要的。

(2)決定執行解答工作時，所需要的動作，以及在解答的過程中，需要被儲存的資料或訊息之類別／大小。結構的設計及所關聯到的"資料結構"是問題解答工作中的一個決定性的重要部分。計算機的效率，全部問題的正確解答，明白清晰地及有效能地在後面所進行的修正工作，這些全部都需要被考慮進去。可以利用一些專門的技巧來幫助處理這些工作，但是勸告初學者，記取本書中一些較佳實例的忠告。

(3)用 PASCL 寫出或"譯出"（*code*）所儲存的資料或訊息及動作爲明確的陳述。

(4)配製所要寫的程式及其資料爲計算機能夠清晰易讀的形態。有許多特殊的打字機能夠做這些工作；其直接地與計算機連接，稱之爲"線上作業"（*on Line*）。其他，連續轉動打擊的紙帶或打

過孔的長方形卡片，代表每一字符；這些紙帶及卡片藉著快速的"輸入"裝置送入計算機中。

(5)命令計算機將PASCAL程式轉換爲一些計算機的指令，同時遵循服從這些指令。這兩個處理動作，被稱爲"編譯工作"(*Compilation*)及"執行"(*running*)。編譯工作是由已經存在計算機內之比較特殊的程式，被稱爲"編譯程式"(*compiler*)來完成的。這些用來作編譯，取得前面所儲存的資料，等的命令，經常被寫成"工作控制陳述語"(*job control statements*)放在程式的開頭處。每一個計算機都有不同的記號，來表示工作控制陳述語，在書寫這些陳述語之前，必須先查閱相關的文件資料。

(6)計算機執行程式，將假設程式沒有發生錯誤(參閱(7))，同時得到結果。這些答案可以顯示在"輸出裝置"上，如線上作業用以鍵入程式的終端機或列表機(*line printer*)，後者在快速之下，一次印出一行(每分鐘1000行)。這些結果還可以做一些檢查而加以更正。

(7)解答工作除了產生所需要的資料或訊息外，不需要再作一些其他的動作。不管怎麼樣，由計算機來處理較複雜的問題之解答，及由所需要的組織結構產生清晰明白的程式，在第一次嘗試的時候，往往都不會成功。從程式中找出錯誤的所在，其動作稱之爲"偵錯"(*debugging*)。這些錯誤之起源，是因爲程式不能被編譯程式所接受，而當作其爲正確的PASCAL 程式。這種錯誤稱之爲"編譯時之錯誤"(*compile time errors*)，另外有一種錯誤爲"執行時之錯誤"(*RUN time errors*)，其發生在計算機"遵從"(*obeys*)或稱之爲執行(*executes*)那些解答之動作的時候。努力而又小心謹愼地利用各式各樣的資料(正確及錯誤)來"試驗"(*testing*)解答之動作，可以發現出很多可能發生的錯誤。

對於一個程式是否爲一個正確的PASCAL 程式，至少必須符合所定義的語言之文法結構。所謂 PASCAL 之文法結構卽爲"文

法結構順序圖表"（ *syntax diagrams* ）所指示的，其表示每一個種類之句子，其所允許的一些情形，以及在那些地方這些句子可以出現。這些順序圖表將出現在本書的各章節中，此外，附錄Ⅰ中有完整的順序圖表。有一個使用這些圖表的例子，如下：

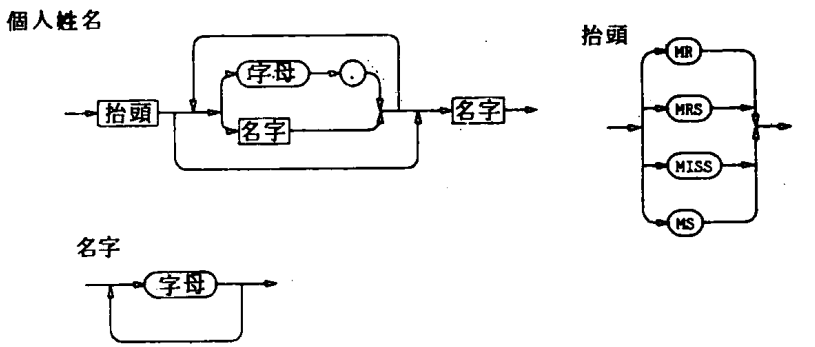

圖1-1

圖 1 － 1 中的順序圖表，是描述所有人的姓名之一般所使用的形式。長方形方格所包圍的文字，是在別的地方所定義的；圓形的圓圈所包圍的實際的字符，其爲必須出現的或沒有其他釋義的文字或符號。還有要注意到，"抬頭"（ *title* ）有好幾種可能的形式；在姓的前面之名字，可以不寫或寫成一個或更多名字之首字母的縮寫，及／或全部寫出。

讀者可以發現，嘗試用文句寫出完整的程式，對於測驗其是否能夠瞭解每一章的資料，是有很大幫助的。在每一個程式之最頂端的註解或在文句中的描述，是定義問題的解答方法。作一個比較來驗證，所嘗試去作的解答是否能令人滿意。在每一章結束的地方，有習題及程式作業，可以當作是一個很好的嘗試。

習　題

使用文法結構順序圖表來描述所購買物品的明細表，至於項目的形

式如下列例子所示。你可以假設存有不同類別的貨品，以及所限制的數量。

4磅 馬鈴薯	0.47	英磅
2盎斯（英兩）蒜	0.15	英磅
5個 蘋果	0.30	英磅