資訊科學譯叢 ①

Pascal programming

Laurence V. ATKINSON

Pascal程式設計

李汶雄

湯耀中 博士 校訂

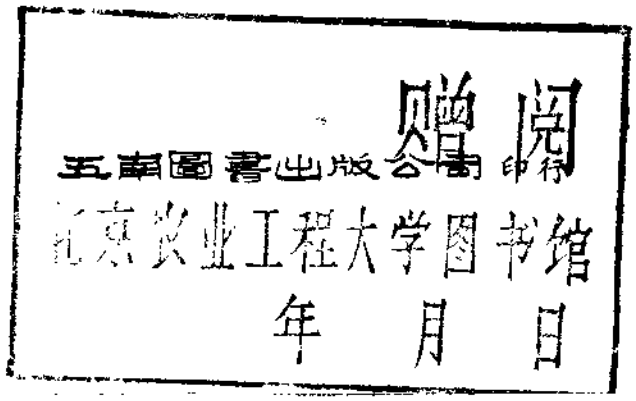
五南圖書出版公司 印行

243

科學新知/技術先導

Pascal程式設計

Laureace V. Atkinson 原著
湯耀中博士 校訂
李汶雄 編譯



Pascal 程式設計

中華民國72年 1 月初版

編譯者 李 汝 雄
發行人 楊 榮 川
發行所 五南圖書出版公司
局版臺業字第0598號
臺北市銅山街1~1號
電 話：3916542號
郵政劃撥：106895號
印刷所 明 文 印 刷 廠

售價230元

（本書如有缺頁或倒裝，本公司負責換新）

序

這本書一方面是對那些沒有絲毫程式概念的人作一番介紹，使之有所了解；另一方面，是要給那些學過某些高階語言，但沒學過Pascal 的人學習Pascal的機會。這就是我們計劃讓此書所扮演的雙重角色。

在本書中，我們竭盡所能的介紹一些有關計算機的基本知識，儘管您以前對計算機沒有一點知識，但願你讀了本書之後，對計算機能獲得一基本的概念。

第一至第八章，介紹一些程式觀念，這些觀念對於任一種高階程式語言而言，都是基本的。雖然你學的是Pascal，但事實上，很多其他程式語言的用法，都是相同的。學習優良的程式跟你所用的語言有關，這點在有關迴路（第六章）的討論中，尤其被強調。迴路常在大部分的程式中出現，但對初學者而言，卻是個麻煩的根源。所以在本書中特別重視迴路。

有經驗的程式家可以快速地讀這些章節，而且熟悉Pascal 解析的觀念。初學者則可為下面幾章打下堅固的基礎。

第九章介紹兩種別種語言所沒有的用法。這是Pascal

最新最重要的兩個觀念。這兩個觀念對於我們思考問題、寫程式的方法、對自己程式的信心，及程式在計算機執行的時間都有遠大的影響。這兩種觀念適用的範圍及他們與程式保障關係的研討，構成了一個相當完整的組織，說明了他們的重要性。從這章所學到的知識，將會在以下的幾章普遍用到。

第十至第十四章，敘述Pascal 可使你具有組織資料的能力。

第十五章是本書最短的一章，它介紹一些較少用的Pascal 指令。

這本書基本上是為教學目的而編的，有次序地教給初學者，使之能充分消化吸收；但是它也可以當做一般社會人士的參考用書，因此本書內容，兩者兼顧。熟讀本書，相信對Pascal 的程式設計，必有相當的理解。

編譯者

71年6月30日

於交通大學

此为试读,需要完整PDF请访问: www.ertongbook.com

Pascal 程 式 設 計

目　　次

第１章　程式與資料 …… 1

1·1　程式 …… 1
1·2　資料 …… 11
1·3　輸出 …… 30

第２章　算術資料型態 …… 35

2·1　整數資料型態 …… 35
2·2　實數資料型態 …… 38
2·3　算術表式 …… 43
2·4　整數溢位 …… 46
2·5　實數的精確度 …… 47
2·6　算術輸入 …… 51

第３章　字元資料型態 …… 53

3·1　字元 …… 53
3·2　標準函數 …… 55
3·3　字元的輸入 …… 57

第4章　控制的有條件流程…… 61

4·1　if — 指述…… 61
4·2　case —指述…… 68
4·3　複合指述…… 74
4·4　巢狀條件…… 76

第5章　布林資料型態…… 85

5·1　布林常數…… 85
5·2　標準函數…… 88
5·3　布林運算子…… 91
5·4　布林變數…… 99

第6章　迴　路……103

6·1　未決迴路……104
6·2　已決迴路……136
6·3　表列輸出……144

第7章　程序與函數……145

7·1　無參數函數……145
7·2　地方性資料……150
7·3　識別字的有效區……153
7·4　參數……153
7·5　函數……160
7·6　前向參照……166

7·7 遞迴 …… 167
7·8 正規程式與函數 …… 187

第8章 程式結構 …… 193

8·1 個案研究 1 …… 193
8·2 仿真常式 …… 205
8·3 漸次設計及漸次測試 …… 207
8·4 個案研究 2 …… 209

第9章 符號及子區間型態 …… 221

9·1 透明性與保障性 …… 222
9·2 使用者定義型態 …… 225
9·3 符號型態 …… 229
9·4 子區間型態 …… 246
9·5 個案研究 3 …… 255

第10章 集合型態 …… 267

10·1 集合的定義 …… 267
10·2 集合型態的宣稱 …… 268
10·3 集合的運作 …… 269

第11章 記錄型態 …… 285

11·1 WITH指述 …… 290
11·2 記錄不定部 …… 291
11·3 壓縮型記錄 …… 296

第12章　檔案型態……299

12·1　本地檔案……299
12·2　本文檔案……307
12·3　外部檔案……307
12·4　案例研討 4……308

第13章　陣　列……321

13·1　密集型陣列及字串……332
13·2　多維陣列……336
13·3　案例研究 5……355

第14章　指標型態……369

14·1　指標型態與變數……369
14·2　儲存方式……370
14·3　遞迴性質料結構……371
14·4　串列……374
14·5　樹狀結構……377
14·6　一資料結構的實例……385

第15章　Goto 指述……393

15·1　指述標號……393
15·2　GO TO 指述的使用……394

附　錄

附錄 1　PASCAL 語法圖 403
附錄 2　保留字 413
附錄 3　預設識別字 414
附錄 4　運算子 415
英中名詞對照表 419

第 1 章

程式與資料

1-1 程 式

1.1.1 演算法、程式及順序流程

將一個過程分成一系列的步驟來表示的方法稱爲演算法（algorithm）。例如以下的三個步驟構成一個演算法：

1. 將蛋、麵粉、糖、奶油及人造奶油攪拌好；
2. 將拌合物放入糕餅模內；
3. 放入烤爐內烤 30 分鐘。

使用計算機幫忙解決問題時，需要將各種演算法存放在計算機內部，因此要有一種適當的符號來表示我們的演算法，此符號稱爲程式語言（programming language）。演算法中各個步驟所執行的動作，就由對應的程式語言內所謂的指述（statement）完成。一序列完整的指述集合稱爲一個程式（program）。計算機執行（run）一個程式，就表示它在依序執行各個指述。

1.1.2 編譯與執行

計算機並不直接執行上述的指述，而須先將每一個指述轉換成對應的機器碼（machine code）。這個轉換過程（稱做編譯，compilation

此为试读，需要完整PDF请访问：www.ertongbook.com

）是由一特別爲此一目的而設計的程式所完成。我們可以定義出一個將某一種計算機語言（原始語言， source language）轉換成另一種計算機語言（目的語言， target language）的演算法，也可以寫一個程式執行此一轉換；這個程式就叫做編譯器（ compiler ）。

一計算機上的PASCAL 編譯器就是一個將PASCAL 程式翻譯成同一機器上的機器碼的程式。任何一套計算機只要有PASCAL 編譯器，就可以在上面執行PASCAL程式。通常編譯器都會列出（list，即印出）原程式，同時在各列之前加上序號。

一個程式可能因爲不完整或包含某些錯誤而無法完成編譯的過程。一語言文法的規則稱做此語言的語法（ syntax ）。在計算機術語內，任何違反語法的現象稱爲語法錯誤（ syntax error ）。如下列的一句英文。

Programming (with Pascal is. fun.

包含兩個語法錯誤：

1 逗號不能在 is 與 fun之間出現，

2 左括弧不得出現，不然就得在Pascal 之後加右括弧。

第二個錯誤給了我們一個提示：一個語法錯誤可能有很多方法可以改正。因此編譯器能發現錯誤的存在，但不一定能正確的指出錯誤發生的位置或其原因。

一個合乎語法的句子可能是無意義的。例如一個簡單的英文句子的格式可能爲：

名詞子句＋動詞＋名詞子句

則底下的兩個句子的語法都是正確的：

1. *A girl recites a poem.*
2. *A poem recites a girl.*

第二個句子的意義是多麼的滑稽！一個程式的意義以語意（semantic）來描述。往後將陸續討論語法與語意的規則。

當編譯器發現一程式含有語法或語意錯誤時，計算機就不會去執行這個程式；此時程式就須改正，重新編譯，直到沒有錯誤發生，計算機才會去執行（execute）它。

一個程式在執行時仍可能發生錯誤（譬如你要求計算機印出在Z之後的字母），往後，讀者會了解爲什麼有些錯誤只在執行時才能發現。當發生執行時錯誤（run-time error），則程式必須修正，重新編譯然後再執行。附錄五列出了各種錯誤及錯誤碼的一覽表。

1.1.3 程式的輸出

所有的程式都應該產生輸出（通常在終端機，terminal，或列表機，line printer 上）。若輸出裝置不具備小寫字母的印出能力時，所有的字母都以大寫形式印出。

在PASCAL語言裏，輸出是由

write

及

writeln

兩個程序（procedure）完成。一個程序就是爲執行某一特定工作的一些指令的集合。

括弧內參數若多於一個就以逗號分開。參數可用來傳遞工作所需的個體。對write和writeln來說，欲印出的資料（data）就由參數

來傳遞；程序本身由系統提供，參數則由使用者（user）提供。

本章的最後會詳細討論輸出的觀念，這裏先介紹writeln 的使用規則。首先說明沒有參數的情形，當指述

```
writeln
```

執行時，印字頭（print head） 會移到下一列的起頭位置。如果有參數，則各個參數的值會依次沿著由左至右的方向印出，然後印字頭移到下一列的起始位置。最簡單的情況，參數可能是數或是字串（string，一個或一個以上的字元）。在PASCAL 中，字串是被括在引號 " " 內的。底下是一些writeln 指述的例子：

```
1. writeln (2001)
2. writeln (-9)
3. writeln ('I think therefore I am')
4. writeln
5. writeln ('A prime number:', 17, 'two more:', 5, 7)
```

如果上面的五個指述一個接一個的被執行，則其輸出結果爲：

```
2001
-9
I think therefore I am

A prime number:       17two more:       5       7
```

整數的輸出格式是由預設的欄寬（field width）決定，其值因編譯器而異，因此每個整數之前的空格數也不一定。字串的輸出不會產生額外的空格，但如例1.3 與例1.5 所指一般，字串內可含有空格。

含有小數的數稱做實數且以標準浮點格式（standard floating point format） 印出，這種格式由係數與指數兩部份構成，中間以字母 "E" 分開。$\alpha E\beta$ 的值即爲 $\alpha \times 10^{\beta}$。係數的值限制在其絕對值

須大於或等於1且小於10，有效位數則視編譯器而定；指數部份則爲帶正負號的整數。底下是一些實數與其對應的標準浮點格式：

實數	S.f.p. 格式
14.2	1.420000E+01
-613.94173258	-6.139417E+02
3.1415926535897932384 6	3.141593E+00
0.000001234567 89	1.234568E-06
97.0	9.700000E+01

算術表式（arithmetic expression）的值也可以輸出。指述：

```
writeln (4,' +', 19,' =', 4 + 19)
```

會產生底下的輸出結果：

```
4 +        19 =        23
```

數字可視爲字元（character）也可視爲整數。如：

```
4 * 19 =        76
```

是由指述：

```
writeln ('4 * 19 =', 4 * 19)
```

所產生。引號內的4 * 19代表的是“4” “*” “1” “9” 四個字元；而引號外的是表示“4乘19”。

算術表式在下一章會詳細討論，至於布林表式（boolean expression）之值雖然也可輸出，但將在第五章才討論。

1.1.4 一個完整的程式

每個PASCAL程式都由一個格式如下的指述起頭：

```
program α (output);
```

其中 α 是一個用來識別這個程式而不影響程式內部的識別字，稱做程式名稱（program name）。

每個識別字（identifier）都是由若干個字元構成，每個字元可是英文字母或數字，但第一個字元一定要是英文字母，且大小寫字母是不分的（全都視爲大寫）。PASCAL 並不限制識別字的長度；但由於識別字不能含有空格且不能跨列，所以實際上還是有一點點限制。爲實際的需要，多數的PASCAL 編譯器都只認前8個字元，但允許超過8個字元的識別字出現。

program是一個保留字（reserved word）。保留字不能在任何非規定所允許的地方出現，例如**program**就只能在程式的開頭出現。本書所有的保留字都以黑體字印刷。但在程式要輸入計算機時，保留字和其他的識別字型式上是完全相同的。

接在程式名稱之後的參數，是程式和其操作環境藉以溝通的橋樑。如果需要用到列表機或終端機當做輸出裝置，則標準輸出檔案（standard output file）名稱output必須出現在程式名稱之後。

組成程式的所有指述必須在保留字**begin**和**end**之間，且**end**之後要有一個點。底下是一個完整的PASCAL程式：

```
0   program OurFirstProgram (output);
1   begin
2       writeln ('Yippee - it works!')
3   end.
```

序碼純粹爲了討論方便，並非程式的一部份。

相連的指述係以分號分隔。通常分號緊接在其所分隔的兩個指述

中的前一個的後面。在 **end** 之前的指述不需接著分號；但由於 PASCAL 允許空指述（empty statement）的出現，所以就算有分號也不算錯。底下是一個含兩個指述的程式：

```
0    program Add7and16 (output);
1    begin
2      writeln ('This is our second program');
3      writeln ('it adds 7 and 16 to give', 7+16,'.')
4    end.
```

當程式執行時，會有底下的結果產生：

```
This is our second program
it adds 7 and 16 to give      23.
```

如果第三列的writeln 指述中的第二個參數（7+16）以（7—16）代替，則輸出變成：

```
This is our second program
it adds 7 and 16 to give      -9.
```

雖然程式本身沒有編譯時錯誤，而且成功的執行，正常的結束，但 7 加 16 並不等於－9；也就是有時計算機照我們所告訴它的做了，但却不是我們本意要做的，稱這種錯誤爲邏輯錯誤（logical error）。

上面的敍述强調了一點：一個程式成功的執行完畢後，並不意味輸出結果也完全正確！

1.1.5 程式本文的佈置

㈠註解：在程式中加入敍述性文字，對別人了解這個程式是很有幫助的，但編譯器對這些文字並不做特別的處理。PASCAL 編譯器對任何出現在大括號（{ 和 }）內的字元都視爲程式的註解（comment），只在程式列出時同時列出而不做其他的處理。組成註解的字元內不能含有右大括號} 如果你使用的計算機的字元集合不包含大括號，可用“(*”和“*)”代替。

此为试读，需要完整PDF请访问：www.ertongbook.com