結構化的語言
PASCAL
STRUCTURED PASCAL

TREMBLAY
BUNT
OPSETH 原著

陳榮洲
蔡家顯 合譯
許永高

松崗電腦圖書資料有限公司

結構化的語言
PASCAL
STRUCTURED PASCAL

TREMBLAY
BUNT
OPSETH 原著

陳榮洲
蔡家顯 合譯
許永高

松崗電腦圖書資料有限公司 印行

結構化的語言
PASCAL

每本定價 220 元整

書號：210133

編著者：陳榮洲、蔡家顯、許永高

發行人：吳 守 信

發行所：道明出版社

台北市仁愛路二段一一〇號三樓

總經銷：松崗電腦圖書資料有限公司

台北市仁愛路二段一一〇號三樓

電話：*3930255 · 3930249*

郵政劃撥：*1 0 9 0 3 0*

印刷者：泉崗印刷設計股份有限公司

台北市仁愛路二段一一〇號三樓

電話：*3930255 · 3930249*

中華民國 七十 年十二月 初 版
中華民國七十一年 五 月 第二版
中華民國七十二年 三 月 第三版

本出版社經行政院新聞局核准登記，

登記證號為局版台業字第一七二九號

目 錄

第4章 向量及列表

第5章 字串與事物

第6章 副程式：函數與程序

第七章 PASCAL的程式撰寫風格

第8章 數值計算

第九章 進一步的字串處理

第10章 線性資料結構

第11章 樹

附錄：PASCAL的結構

第1章　PASCAL程式語言緒論

人類之間的交互作用大部份都是經由語言的媒介來達成。語言使得我們的思想與觀念能夠表達，沒有語言我們將很難去溝通。

在計算機中，程式語言充當有問題要解決的人們與能夠幫忙解決問題的計算機間的溝通媒介。語言能夠影響使用者的思想與文化。例如，愛斯基摩人光在雪的主題上就有很龐大的字彙出現。

一個有效的程式語言可以加強計算機程式的發展與表達力。在太過於沒有結構性的人類思想與計算機執行的精確度之間的鴻溝必須加以橫跨疏通。程式語言修改了程式員的一些思想。而語言的品質對於出產的程式品質有很大的影響。

1-1 導言

本書的目的是作爲課堂上之輔助，介紹PASCAL 這種程式語言給你，使你能夠利用此種語言解決你在課堂所遭遇的問題。

我們盡可能解說得詳細，並且儘量把握住要點，書中還舉了非常多的例子與問題來幫助讀者更容易瞭解。本章的目的是對這本書所用到的材料作一番透視，並對於PASCAL 語言的使用與發展作一大概之說明。

1-2 PASCAL語言的簡史

在早期的計算時代，撰寫程式是一非常可怕的工作。許多早期的計算機都是用“硬性連接”的方式來執行一特定的工作，若是要修改“程式”必須重新連接不同的計算機元件。John von Neumann是第一位提出儲存程式（stored program）的構想的人，亦即將程式中的指令（instructions）與資料（data)全部儲存在計算機（computer）的記憶體（memory)上。不久，人們開始尋找如何更方便的表示這些指令（instructions），因此就從以機器導向（machine-oriented)的低階組合語言（low-level assembly language)發展到以問題導向（problem-oriented）的高階程式語言（high-level programming language）。

第一個一般目的(general purpose)的問題導向程式語言爲FORTRAN（FORmula TRANslator），在1954年發表，專門爲科學的數值問題而設計。FORTRAN是一種有用的工具，並證明了問題導向程式語言（problem-oriented programming language）的價值與成本的效率。所以不久之後其他的高階語言便相繼出現了，有商業應用的COBOL語言（Common Business Oriented Language），數值的數學問題之ALGOL語言（ALGOrithmic Language），串列處理（list processing）應用（主要在人工智慧上）的LISP語言，字串處理（string manipulation）應用的SNOBOL 語言，以及已證明是相當有持久性的一般目的（general-purpose）語言PL/I。

PASCAL語言，是依十七世紀法國數學家的名字取的，它是由瑞士（Switzerland）的蘇黎克（Zurich）之科學技術聯邦機構的Niklaus Wirth於1968年提出的。但是第一個PASCAL編譯器（compiler）直到1970 年才能使用，不過到了1978年的十一月時已經有超過110個不同PASCAL編譯器（compiler）被發表。

PASCAL是個一般目的（general-purpose）的語言，它能被廣泛地應用在許多方面。雖然它有強大的程式力量，但它仍然相當精簡且容易學習。此種語言有一特別強的特性是在它的資料結構能力（data structuring capabilities），它能將一般程式員必須做的許多工作轉移給編譯器（compiler）完成。此種語言愈來愈流行，因爲它在設計之時與介紹之初已經作了許多標準化的努力。目前標準的PASCAL是在一項 The PASCAL Report〔Jensen and Wirth(1974)〕文件中定義的。

適合於教學環境的程式語言編譯器最好是低成本，編譯快速，過度診斷性（ultradiagnostic），學生導向的。例如在滑鐵盧大學（University of Waterloo）發展的WATFOR/WATFIV，以及康乃爾大學（Cornell University）發展的PL/C，已經在FORTRAN與PL/I的可接受性上由學術界花下了很大的心血。由於PASCAL之簡單性使得它很自然地適合於教學環境。事實上，許多的編譯器（compiler）都適用在迷你電腦（minicomputer）與微電腦（microcomputer）上。

1-3 程式語言的使用

如同前面所述地，一個程式語言能夠幫助將一個問題轉換成可執行的計算機程式（executable computer program）。實際上，此種定義良好（well-defined）的語言，不僅加強了解決問題的表達力，同時也加強了其發展能力。

一旦一個問題的解答已經寫成某種程式語言的計算機程式，它必須被翻譯成計算機的機器語言（machine language）以便程式運轉（run）時用。機器語言（machine language）並不是程式員導向的（programmer-oriented）；機器語言不過是一長串數目字，它是用這種數字方式寫的並且爲計算機所辨認的。將

一高階程式語言寫的程式（有時稱爲原始程式）翻譯成相對的機器語言（有時稱爲目的程式）是由一個特殊程式所處理的，此特殊程式即爲編譯器（compiler）。

對於一種程式語言，可以有許多編譯器。例如，對於不同型態的機器將有不同的編譯器來工作。即使在同一機器上，也可能對於一種語言同時具備有數種編譯器，每種編譯器都強調不同的特性與能力。

有時編譯器的動作對於程式員（programmer）而言是看不見的，但也不盡然。例如，編譯器能夠偵測出程式中的錯誤，並阻止程式繼續執行下去。有些錯誤則暫時跳過不予測出，直到翻譯完的機器語言程式（又稱目的程式）實際被執行時才發現。有關編譯時錯誤（compile-time error）與執行時錯誤（run-time error）的區別在第二章將有更完整的描述。

1-4 本書的研討方式（approach）

我們在討論PASCAL語言時，也將我們的程式例子在Manitoba大學的IBM S/360機器中運轉過。在PASCAL報告中所描述的語言與Manitoba編譯器中所定義的語言兩者有一些小差別。這些小差別並不影響初學的程式員，但是當程式撰寫的經驗多了以後可能有較大的影響出現。

剛開始學習任何的程式語言時，很容易鑽進牛角尖中。我們將藉著層次地呈現嘗試將此問題減輕。當語言中的某一特性剛被介紹時，對於初學的程式員（beginning programmer）我們是以馬上用得到的方式來描述的。至於有其他變化與額外的類型則延至遇上實際問題時再來給與動機。

近些年來常被提到與論著的程式撰寫方式爲"結構化程式撰寫（structured programming）"。結構化程式撰寫實際上比平常的程式撰寫多一點限制。但是使用此方法的結果證明學生能在較短的時間中製造更好的程式出來。本書所介紹的程式即爲結構化程式撰寫的形式。

第2章　PASCAL語言基本概念

本章介紹PASCAL的資料、變數、和資料處理方法來解決一些簡單問題。並討論簡單的輸入輸出工作。

經由上述討論，讀者將能寫一些簡單的PASCAL程式。

此外，我們將解釋一下，一個PASCAL程式如何利用其編譯程式（compiler）來執行。並列出程式。

2-1 資料、資料型態、及基本算符

計算機（俗名電腦）一般處理伴隨著多量資訊（information）或資料（data）的工作，藉著特定程式將一連串指令（instructions）輸入計算機，程式員（programmer）即可指示如何處理這堆資料，不同於其它程式語言，PASCAL 提供程式員數種基本資料型態（data type）外，還允許程式員依據基本資料型態定義新資料型態；本節討論基本資料型態，及關於資料在 PASCAL 程式中的表示方式和處理資料之運算。

2-1.1 資料型態

PASCAL 語言中包括五種基本資料型態：數值（numeric）、非數值（nonnumeric）、邏輯（logical）、和兩種特殊型態指標（pointer）和集合（set），教材中除了集合外均已詳述，不同的資料型態就有不同的內部表示方式，及用不同的機器指令（machine instructions）處理之，爲了使計算機正確在翻譯資料，程式中必須依據程式員之規定以表示各型態的資料。

PASCAL 中，數值資料以兩種方式表示之，即整數（integer）和實數（real）。

整數書以一串數字（digit），或冠以正、負符號，整數所能表示的最大值視使用的計算機及編譯器（compiler）而定，下列是整數常數的例子：

```
   1
  +1
   0
 -425678
  999999
```

實數記以任一數字串，或含正、負符號，及包含小數點，實數可書以傳統的十進位（decimal）形式，或對極大及極小的數目記以科學記法或浮點格式（floating-point form），後者以一實數或整數接著字母 E（指數）及或含正、負符號的整數表示指數，此種格式解釋爲：字母前的數字乘以 E 後面 10 的整數次方。下面是

實數常數分別記以兩種格式表記的範例：

41341413.35	4.134141335E7
+22.2	+2.22E+01
75.	75E0
-0.000000000000000003	-3E-18

字串（character string）（非數值資料）是PASCAL 中第二種主要的資料型態，字串是由一連串字元符號（character symbol）所組成，每個字元必須爲字母或數字（0123456789）或者是特殊字元（+-*/()=.,$'blank）中的一個，爲了表示起見，字串都括於單引號之間，而單引號不爲字串之一部分，只是標明字串的開始和結尾，亦即分界號（delimiters），下列是正確字串常數的例子：

```
'COMPUTER SCIENCE'
      '67'
'$.50 GOLD PIECES'
    '3+3-4=2'
```

連續的，上前字串格式可能招致某些問題，即字串中若要含有單引號，這個問題可用兩個單引號（單引號對）當作單引號插入字串裡，例如，'COULD NOT'正確的縮寫不是'COULD-N'T'，而是'COULDN''T'。

字串的長度定義爲字元的個數（包括空白），字串裡的單引號對僅算作一個字元，因爲實際上也只表示一個單引號，因此，字串'DON''T'的長度是5非6；字串的最小長度是1，而最大長度隨著編譯器決定，但一般都相當大，這裡我們假定字串的最大長度是256。

PASCAL 中，第三種資料型態是邏輯型態，即資料接受值爲“眞”或“假”，分別表記以TRUE和FALSE，有關邏輯的應用將在第九章中討論，第四種資料型態指標也將在第十章中討論。

異於其它程式語言，PASCAL 允許程式員定義集合，此種型態包含和名稱相關的各個元素，在2-2節中我們將看到有關的例子。

爲使計算機正確地表示資料，資料需指定一適當的形式，程式員必須決定所用

資料的型態，及遵從各型態表示的規則，資料的型態同時決定了可處理該資料的運算。

2-1.2 資料處理

數值運算是PASCAL中最基本的部分，本節中討論數值算符（operator）的符號和用法，此外，這些算符均與演繹語言（algorithmic language）相同，數值函數將在2-2節討論。

數值算符

PASCAL中的算符有兩種型態。即二元的（binary）和一元的（unary），二元算符用於具兩個算元（operand）的運算，二元算符包括：

1. 減法，例3－4
2. 加法，例4＋3
3. 乘法，例4＊3
4. 除法，例4／3（實數除法）
   及例4 DIV 3（整數除法）

PASCAL中不接受指數算符，指數運算可用連續相乘或演繹法達成之。

數值運算的型態

除了除法外，所有算符均能運用於整數、實數或任意組合的運算，二元算符涉及不同型態的資料時，其一資料值將被轉換成另一型態，若一算元爲實數而另一是整數算元，則整數算元轉換成實數算元，然後才對這兩個實數算元作運算，下面是不同型態的運算例子：

運　算	結果之型態
5.6E02＋7.8E33	實數
4＋3	整數
32－8.9	實數

5*7.45E－34	實數
5.99E23*3.87E34	實數

除法有 / 及 DIV 兩種算符，DIV 算符只接受整數算元並執行整數除法，因此，1 DIV 3 結果得 0，4 DIV 4 結果得 1，1 DIV 3.0 是錯誤的，因爲其一算元爲實數，/ 算符可以具整數或實數算元，並且其結果爲一實數，因此，1 / 3 結果得 0.3333333，與 1./3 或 1E+0 / 3.00 結果相同。

上述某些算符亦可執行其它運算，如將在後面幾章討論的集合運算。

任何數值算符均可能導致溢位（overflow）或不足位（underflow）的問題。

溢位發生在運算結果過大，以致在記憶空間裡無法正確地表示該整數或實數值的錯誤情形。

溢位可能發生在：

1. 兩個大的正數或負數相加，
2. 大的正數減去大的負數，
3. 兩個大數目相乘，正或負，
4. 大的數除以相當小的數（僅 / 算符）。

不足位發生在運算結果太小，而無法在記憶空間裡正確地表示該整數或實數值的錯誤情形。

不足位可能發生在：

1. 兩個非常小的數相乘，
2. 非常小的數除以相當大的數（僅 / 算符）。

確保計算結果落於整數或實數值的儲存範圍是程式員的責任，因計算機保存溢位或不足位的數值是不完整的且通常是無用的，計算機一般的回應是印出溢位或不足位的錯誤訊息，同時結束程式的執行；但在某些運算中則不給任何警告信息，而計算機卻以這些不完整的資料繼續執行，因此當可能發生溢位或不足位的情形時，程式員必須特別小心。

本節中，我們討論算符、運算型態及運算時可能發生的問題。