

目 錄

第一課 初識 Pascal 語言	1
一、Pascal 語言概述	1
二、Pascal 語言的特點	1
三、Pascal 語言程序的基本結構	1
四、Pascal 語言系統的使用	2
五、第一個程序	3
第二課 賦值語句、輸出語句	4
一、常量、變量與算術表達式	4
(一) 常量	4
(二) 變量	5
(三) 算術表達式	5
二、賦值語句	6
三、輸出語句	6
(一) 輸出語句的兩種格式	6
(二) 輸出語句的功能	7
四、應用例析	7
第三課 帶格式的輸出語句及輸入語句	9
一、寫語句的輸出格式	9
二、輸入語句（讀語句）	10
三、順序結構程序設計	12
第四課 簡單的分支結構程序設計	14
一、PASCAL 中的布爾（邏輯）類型	14
二、關係表達式與布爾表達式	14
三、簡單的 IF 語句	15
第五課 if 嵌套與 case 語句	17
一、if 語句的嵌套	17
二、case 語句	17
三、選擇結構的程序設計	18
第六課 for 循環語句	21
一、for 語句的一般格式	21
二、For 語句執行過程	21
三、說明	21
四、應用舉例	21
第七課 WHILE 與 REPEAT 循環	23
一、WHILE 循環	23
二、直到循環（REPEAT-until 語句）	24
三、循環結構程序設計	24
第八課 一維數組	27
一、為什麼要使用數組	27
二、一維數組	27
三、一維數組應用示例	28
第九課 多維數組	30
一、多維數組的定義	30
二、多維數組元素的引用	30
三、多維數組的應用示例	31
四、數組類型的應用	32
第十課 字符與字符串處理	34
一、字符、字符串類型的使用	34
(一) 字符類型	34
(二) 字符串類型	34
二、字符串的操作	35
(一) 字符串的運算和比較	35

(二) 字符串的函數和過程	36
第十一課 枚舉、子界、集合及記錄類型	38
一、 枚舉類型	38
(一) 枚舉類型的定義	38
(二) 枚舉類型變量	38
(三) 枚舉類型的性質	38
(四) 枚舉類型應用舉例	39
二、 子界類型	40
(一) 子界類型定義	40
(二) 子界類型數據的運算規則	40
(三) 子界類型應用舉例	41
三、 集合類型	42
(一) 集合類型的定義和變量的說明	42
(二) 集合的值	42
(三) 集合的運算	43
四、 記錄類型	44
(一) 記錄類型的定義	44
(二) 記錄的嵌套	45
(三) 開域語句	45
五、 應用實例	47
第十二課 過程與函數	49
一、 函數	49
(一) 函數的說明	49
(二) 函數的調用	50
(三) 函數的應用舉例	50
二、 過程	50
(一) 過程的說明	51
(二) 過程的調用	51
(三) 過程的應用舉例	52
三、 過程、函數的數據傳遞	52
(一) 數值參數和變量參數	52
(二) 全程變量、局部變量及它們的作用域	53
四、 過程和函數的嵌套	54
五、 子程序（模塊化）結構的程序設計	54
第十三課 動態數據類型	57
一、 指針的定義及操作	57
(一) 指針類型和指針變量	57
(二) 開闢和釋放動態存儲單元	57
(三) 動態存儲單元的引用	58
(四) 對指針變量的操作	58
二、 鏈表結構	59
(一) 鏈表的基本結構	59
(二) 單向鏈表的基本操作	60
(三) 鏈表結點的插入與刪除	60
(四) 環形鏈表結構	62
(五) 雙向鏈表結構	62
三、 綜合例析	62
第十四課 文件	66
一、 文本文件	66
二、 有類型文件	71
三、 綜合例析	75

第一課 初識 Pascal 語言

信息學奧林匹克競賽是一項益智性的競賽活動，核心是考查選手的智力和使用計算機解題的能力。選手首先應針對競賽中題目的要求構建數學模型，進而構造出計算機可以接受的算法，之後要寫出高級語言程序，上機調試通過。程序設計是信息學奧林匹克競賽的基本功，在青少年朋友參與競賽活動的第一步必須掌握一門高級語言及其程序設計方法。

一、Pascal 語言概述

PASCAL 語言也是一種算法語言，它是瑞士蘇黎世聯邦工業大學的 N·沃思 (Niklaus Wirth) 教授于 1968 年設計完成的，1971 年正式發表。1975 年，對 PASCAL 語言進行了修改，作為“標準 PASCAL 語言”。

PASCAL 語言是在 ALGOL 60 的基礎上發展而成的。它是一種結構化的程序設計語言，可以用來編寫應用程序。它又是一種系統程序設計語言，可以用來編寫順序型的系統軟件（如編譯程序）。它的功能強、編譯程序簡單，是 70 年代影響最大一種算法語言。

二、Pascal 語言的特點

從使用者的角度來看，PASCAL 語言有以下幾個主要的特點：

1·它是結構化的語言。

PASCAL 語言提供了直接實現三種基本結構的語句以及定義“過程”和“函數”（子程序）的功能。可以方便地書寫出結構化程序。在編寫程序時可以完全不使用 GOTO 語句和標號。這就易於保證程序的正確性和易讀性。PASCAL 語言強調的是可靠性、易於驗證性、概念的清晰性和實現的簡化。在結構化這一點上，比其它（如 BASIC, FORTRAN77）更好一些。

2·有豐富的數據類型。

PASCAL 提供了整數、實型、字符型、布爾型、枚舉型、子界型以及由以上類型數據構成的數組類型、集合類型、記錄類型和文件類型。此外，還提供了其它許多語言中所沒有的指針類型。沃思有一個著名的公式：“算法 + 數據結構 = 程序”。指出了在程序設計中研究數據的重要性。豐富的數據結構和上述的結構化性質，使得 PASCAL 可以被方便地用來描述複雜的算法，得到質量較高的程序。

3·能適用於數值運算和非數值運算領域。

有些語言（如 FORTRAN 66, ALGOL 60）只適用於數值計算，有些語言（如 COBOL）則適用於商業數據處理和管理領域。PASCAL 的功能較強，能廣泛應用於各種領域。PASCAL 語言還可以用於輔助設計，實現計算機繪圖功能。

4·PASCAL 程序的書寫格式比較自由。

不象 FORTRAN 和 COBOL 那樣對程序的書寫格式有嚴格的規定。PASCAL 允許一行寫多個語句，一個語句可以分寫在多行上，這樣就可以使 PASCAL 程序寫得象詩歌格式一樣優美，便於閱讀。

由於以上特點，許多學校選 PASCAL 作為程序設計課程中的一種主要的語言。它能給學生嚴格而良好的程序設計的基本訓練。培養學生結構化程序設計的風格。但它也有一些不足之處，如它的文件處理功能較差等。

三、Pascal 語言程序的基本結構

任何程序設計語言都有著一組自己的記號和規則。PASCAL 語言同樣必須採用其本身所規定的記號和規則來編寫程序。儘管不同版本的 PASCAL 語言所採用的記號的數量、形式不盡相同，但其基本成分一般都符合標準 PASCAL 的規定，只是某些擴展功能各不相同罷了。下面我們首先來瞭解 Pascal 語言的程序基本結構。

為了明顯起見先舉一個最簡單的 PASCAL 程序例子：

例 1.1：

```
Program ex1_1;           ← 程序首部
var r,c,s:integer;       ← 說明部分
begin
  readln(r); {讀入圓的半徑 r}
  c:=3.14*2*r; {求周長 c}
  s:=3.14*r*r; {求面積 s} ← 註釋
  writeln(c, s); {輸出周長與面積}
end.                    ← 執行部分
```

程序體

從這個簡單的程序可以看到：

1·一個 PASCAL 程序分為兩個部分

程序首部和程序體（或稱分程序）。

2 · 程序首部是程序的開頭部分

包括：

(1) 程序標誌。用“program”來標識“這是一個 PASCAL 程序”。PASCAL 規定任何一個 PASCAL 程序的首部都必須以此字開頭。在 turbo pascal 語言中，首部也可省略。

(2) 程序名稱。由程序設計者自己定義，如例中的 ex1_1。在寫完程序首部之後，應有一個分號。

3 · 程序體是程序的主體，在有的書本裏也稱“分程序”。

程序體包括說明部分（也可省略）和執行部分兩個部分。

(1) 說明部分用來描述程序中用到的變量、常量、類型、過程與函數等。本程序中第二行是“變量說明”，用來定義變量的名稱、類型。PASCAL 規定，凡程序中用到所有變量、符號常量、數組、標號、過程與函數、記錄、文件等數據都必須在說明部分進行定義（或稱“說明”）。也就是說，不允許使用未說明先使用。在 Turbo Pascal for Windows 中若要在屏幕上輸入輸出資料，必須在說明部分加上 **uses wincrt;** 語句。

(2) 執行部分的作用是通知計算機執行指定的操作。如果一個程序中不寫執行部分，在程序運行時計算機什麼工作也不做。因此，執行部分是一個 PASCAL 程序的核心部分。

執行部分以“begin”開始，以“end”結束，其間有若干個語句，語句之間以分號隔開。執行部分之後有一個句點，表示整個程序結束。

4 · PASCAL 程序的書寫方法比較靈活。

當然，書寫不應以節省篇幅為目的，而應以程序結構清晰、易讀為目的。在編寫程序時儘量模仿本書中例題程序格式。

5 · 在程序中，一對大括號 { } 間的文字稱為注釋。

注釋的內容由人們根據需要書寫，可以用英語或漢語表示。注釋可以放在任何空格可以出現的位置。執行程序時計算機對注釋不予理睬。

四、Pascal 語言系統的使用

目前，常用的 Pascal 語言系統有 Turbo Pascal for Windows 與 Free Pascal，信息學奧林匹克競賽的一種指定語言就是 Free Pascal，下面我們就來學習 Free Pascal 系統的使用。

1 · Pascal 系統集成環境簡介

圖 1 和圖 2 所示的分別為 Turbo Pascal for Windows 和 Free Pascal 的集成環境。

圖 2 最頂上一行為主菜單。中間藍色框內為編輯窗口，可以進行程序的編輯。最底下一行為提示行，顯示出系統中常用命令的快捷鍵，如將當前編輯窗口中文件存盤的命令快捷鍵為 F2，獲得系統幫助的快捷鍵為 F1，等等。

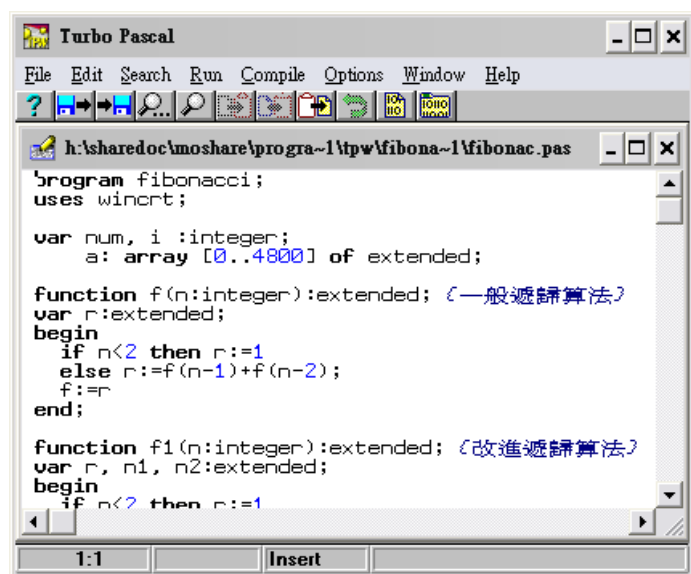


圖 1

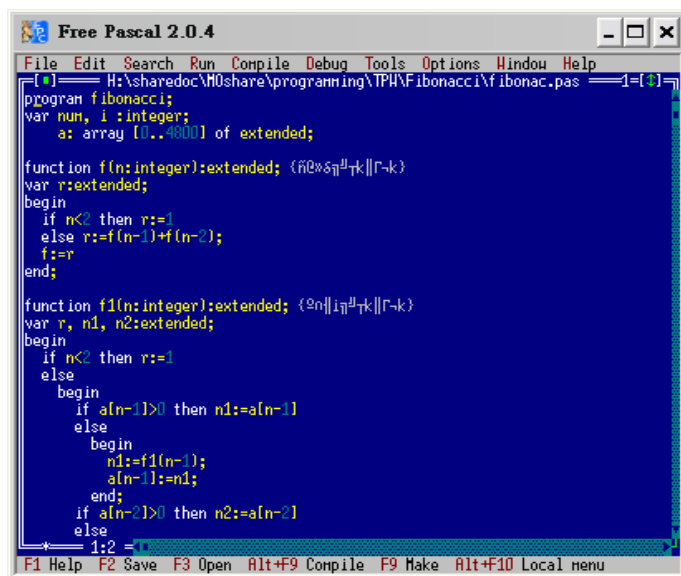


圖 2

2 · 新建程序窗口

按 F10 進行主菜單，選擇 FILE 菜單，執行其中 New 命令。就可建立一個新的程序窗口（默認文件名為 Noname00.pas 或 Noname01.pas 等）。

3・程序的輸入、編輯與運行

在當前程序窗口中，一行一行的輸入程序。事實上，程序窗口是一個全屏幕編輯器。所以對程序的編輯與其它編輯器的編輯方法類似，這裏不再重複。

當程序輸入完畢之後，一般要先按 Alt+F9（或執行 compile 菜單中 compile 命令）對程序進行編譯。如果程序有語法錯誤，則會在程序窗口的第一行處顯示第一個紅色錯誤信息。若無語法錯誤，則窗口正中央會出現一個對話框，提示編譯成功。接下來，我們可以運行程序了。

程序的運行可以通過按 ALT+R 打開 RUN 菜單中的 RUN 命令，或直接按快捷鍵 CTRL+F9。則可以在用戶窗口中輸出運行結果。通常在程序運行結束後系統回到 Pascal 系統的集成環境，因此要查看運行結果，要按 ALT+F5 將屏幕切換到用戶屏幕。

4・程序的保存與打開

當我們想把程序窗口中的程序存入磁盤時，可以通過按 F2 鍵（或執行 File 菜單中的 save 命令）來保存程序。第一次保存文件時屏幕上會出現一個對話框要求輸入文件名（默認擴展名為 .pas）。

當我們要將磁盤上的程序文件中的 PASCAL 程序裝入窗口時，可按 F3（或執行 File 菜單中的 Open 命令）來裝入程序，此時系統也會彈出一個對話框要求輸入要打開的文件名，或直接在文件對話框列表中選擇所要的文件，然後回到打開文件。

五、第一個程序

下面程序在運行時，會提示輸入一個圓的半徑，然後會在屏幕上畫一個圓。按回車後程序結束回到程序窗口。

```
Program ex_first;
Uses graph;
Var Gm,Gd,R :integer;
Begin
  Gd:=0;
  Write('Please enter the radius:');readln(R);
  Initgraph(Gm,Gd,' ');
  Setcolor(Green);
  Circle(320,240,R);
  Readln;
  Closegraph;
End.
```

注意，如果上面程序運行時會出現初始化圖形錯誤，請將系統目錄下 BGI 子目錄 EGAVGA.BGI 和 UNITS 子目錄中的 Graph.tpu 拷貝到系統目錄下 BIN 目錄即可。

練習一

- 1、請輸入上面的程序，並練習將其存盤、打開與運行上面程序。

第二課 賦值語句、輸出語句

上節課，我們學習了 Pascal 語言的程序基本結構，在一個程序中，所有的操作都由執行部分來完成，而執行部分又都是由一個個語句組成的。因此，下面開始我們要學習 pascal 語言的基本語句，並且在學習過程中逐步學會程序設計的基本方法。

這節課我們要學習兩種語句，即賦值語句與輸出語句。在語句學習之前我們要先瞭解一些 pascal 語言的基礎知識。

一、常量、變量與算術表達式

(一) 常量

在程序運行過程中，其值不能被改變的量稱為常量。如 123，145.88，'abc'，true 等。

1. 整型常量

整型常量採用我們平常使用的十進制整數表示。如 138，0，-512 等都是整型常量，而 18. 或 18.0 都不是整型常量。

pascal 中有一個標準標識符 MaxInt，它代表所使用的計算機系統允許的最大整型數，而最小的整型數即為 -MaxInt-1。

Turbo Pascal 還定義了長整型常量 MaxLongInt，其值為 2147483647。

2. 實型常量

實型常量包括正實數、負實數和實數零。pascal 中表示實型常量的形式有兩種。

(1) 十進制表示法

這是人們日常使用的帶小數點的表示方法。如 0.0，-0.0，+5.61，-8.0，-6.050 等都是實型常量，而 0.，.37 都不是合法的實數形式。

(2) 科學記數法

科學記數法是採用指數形式的表示方法，如 1.25×10^5 可表示成 1.25E+05。在科學記數法中，字母“E”表示 10 這個“底數”，而 E 之前為一個十進制表示的小數，稱為尾數，E 之後必須為一個整數，稱為“指數”。如 -1234.56E+26，+0.268E-5，1E5 是合法形式，而 .34E12，2.E5，E5，E，1.2E+0.5 都不是合法形式的實數。

無論實數是用十進制表示法還是科學表示法，它們在計算機內的表示形式是一樣的，總是用浮點方式存儲。

和整數相比，實數能表示的範圍大得多，但值得注意的是實數的運算整數的運算速度慢且無法像整數那樣精確表示，只能近似表示。

3. 字符常量

在 Pascal 語言中，字符常量是由單個字符組成，所有字符來自 ASCII 字符集，共有 256 個字符。在程序中，通常用一對單引號將單個字符括起來表示一個字符常量。如：'a'，'A'，'0' 等。特殊地，對於單引號字符，則要表示成 ''。對於 ASCII 字符集中，按每個字符在字符集中的位置，將每個字符編號為 0—255，編號稱為對應字符的序號。

4. 布爾常量

布爾型常量僅有兩個值，真和假，分別用標準常量名 true 和 false 表示。它們的序號分別為 1 和 0。

5. 符號常量

一個常量即可以直接用字面形式表示（稱為直接常量，如 124，156.8），也可以用一個標識符來代表一個常量，稱為“符號常量”。但符號常量必須在程序中的說明部分定義，也就是說先定義，後使用。

定義符號常量的一般格式：

CONST

<常量標識符>=<常量>;

說明：常量說明部分以關鍵字 const 開頭，後面的標識符為常量標識符，其中“=”號後的常量為整數、實數、字符、字符串（字符、字符串常量在後面章節中將作介紹）。而且，在常量說明部分可以將幾個常量說明成符號常量，共用一個關鍵字“const”。例如：

```
program ex_const;
const pi=3.14159; {常量說明}
var r,c,s:real;
begin
  write('Enter r='); readln(r);
```

```
c:=2*pi*r;  
s:=pi*r*r;  
writeln('c=',c);  
writeln('s=',s);  
end.
```

則在本程序中 `pi` 作為符號常量，代表實數 3.14159。也就是說，常量說明部分既定義了常量名及其值，又隱含定義了常量的類型。關於符號常量，應注意下列幾點：

- (1) 符號常量一經定義，在程序的執行部分就只能使用該常量標識符，而不能修改其值。
- (2) 使用符號常量比直接用數值更能體現“見名知義”的原則，也便於修改參數，故一個較好的程序中，應儘量使用符號常量，在執行部分基本上不出現直接常量。

(二) 變量

變量代表了一個存儲單元，其中的值是可變的，故稱為變量。如遊戲“魂鬥羅”中玩者命的個數最初為 3，當你死了一次命減少一，這裏命的個數就是一個變量（或者說命的個數存儲在一個存儲單元中）。即在程序運行過程中，其值可以改變的量，稱為變量。

變量有三個要素是：變量名、變量類型、變量值。

一個程序中可能要使用到若干個變量，為了區別不同的變量，必須給每個變量（存貯單元）取一個名（稱為變量名），該變量（存貯單元）存放的值稱為變量的值，變量中能夠存放值的類型為變量的類型。例如“魂鬥羅”遊戲中用於存放“命”的變量，在遊戲程序中的名字可取為 `N`，它的類型為整型，遊戲初始時這個變量的值為 3。

1. 變量名

用一個合法的標識符代表一個變量。如 `n`, `m`, `rot`, `total` 等都是合法變量名。在程序中用到的變量必須在說明部分加以說明，變量名應遵循自定義標識符的命名規則，並盡量遵照“見名知義”的原則，即用一些有意義的單詞作為變量名。

“自定義標識符”的命名規則為：自定義標識符必須以字母（包含下劃線“`_`”）開頭，後面的字符可以是字母或數字。標識符長度不超過 63 個字符。

2. 變量的類型

常量是有類型的數據，變量在某一固定時刻用來存放一個常量，因此也應有相應的類型。如整型變量用來存放整數，實型變量用來存放實數。

3. 變量說明

在程序中若要使用變量，變量的名稱及類型在程序的變量說明部分加以定義，變量的值則在程序的執行部分中才能賦給。

變量說明的一般格式：

VAR

<變量標識符>[，<變量標識符>]:<類型>;

（中括號內部分表示可省，下同）

其中 `VAR` 是 pascal 保留字，表示開始一個變量說明段，每個變量標識符或由逗號隔開的多個變量標識，必須在它的冒號後面說明成同一類型。一個程序中，可以說明許多不同類型的變量，每種類型變量之間用分號隔開，共用一個 `VAR` 符號。

例如：

```
var  
  age, day: integer;  
  amount, average: real;
```

其中，`Integer`（整型）、`Real`（實型）是標準標識符，它們是“類型標識符”，代表了確定的類型，如 `age` 和 `day` 被定義為整型變量，`amount` 和 `average` 被定義為實型變量。

一旦定義了變量，就確定了它的類型，也就是說，就確定了該變量的取值範圍和對該變量所能進行的運算。

(三) 算術表達式

1. 算術表達式的定義

pascal 語言中的算術表達式是由符合 pascal 語法規定的運算對象（包括常量、變量、函數）、算術運算符、圓括號組成的有意義的式子。如：`A+3.14159*5/8.4-Abs(-1123)`

2. 算術運算符

常用的有以下 6 個算術運算符：

(1) + (加)

(2) - (減)

(3) * (乘)

(4) / (實數除) 得到結果為實型。如 $5.0/2.0=2.5$ ， $5/2=2.5$ ， $4/2=2.0$ 而不等於 2。

(5) DIV (整除) DIV 它要求除數和被除數均為整型，結果也為整型。如 $10 \text{ DIV } 2=5$ ， $10 \text{ DIV } 3=3$ ， $5 \text{ DIV } 10=0$ ， $-15 \text{ DIV } 4=-3$ 。DIV 運算只取商的整數部分，參與 DIV 運算的兩個對象不能為實型。

(6) mod (求餘)，也只能用於整數運算，結果為整數。例如： $10 \bmod 4=2$ ， $-17 \bmod 4=-1$ ， $4 \bmod (-3)=1$ ， $-4 \bmod 3=-1$ ，即 $a \bmod b=a-(a \text{ div } b)*b$ 。

3·運算優先順序

如果一個表達式裏出現兩個或兩個以上的運算符，則必須規定它們的運算次序。pascal 規定：

(1) 表達式中相同優先級的運算符，按從左到右順序計算；

(2) 表達式中不同優先級的運算符，按從高到低順序計算；

(3) 括號優先級最高，從內到外逐層降低；

在算術運算中運算符的優先順序與數學上的四則運算一致，即“先乘除後加減”（注：“MOD”、“DIV”運算的優先級與“*”、“/”相同）。

二、賦值語句

變量既然代表一個存儲單元，其值是可變的，那麼其中的值是怎麼提供的，又是怎麼改變的呢？可以通過賦值語句來進行。

1·賦值語句的格式

變量名:=表達式;

其中“:=”稱為賦值號。

2·執行過程

計算機先計算賦值號右邊表達式的值，然後將表達式的值賦給變量名代表的變量。如： $A:=(9*8)-(2-1); A:=A+1$

三、輸出語句

輸出語句的作用是將程序運算的結果輸出到屏幕或打印機等輸出設備。這裏通常是指輸出到屏幕。

(一) 輸出語句的兩種格式

1·write 語句

格式：

Write(表達式 1，表達式 2，.....);

如：

```
write(1,2,3,4);  
write(1.2,3.4,5);  
write('My name is Liping');
```

2·writeln 語句

格式：

Writeln(表達式 1，表達式 2，.....);

或

writeln;

(二) 輸出語句的功能

計算機執行到某一輸出語句時，先計算出輸出語句中的每個表達式的值，並將每一個表達式的值一個接一個地輸出到屏幕上。

Write 語句與 writeln 語句格式上都相似，但它們在功能上有所不同，兩個語句的區別在於，write 語句將其後括號中的表達式一個接一個輸出後，沒有換行。而 writeln 語句則在輸出各個表達式的值後換行。

例如以下兩個程序段的輸出分別為：

```
write(1,2,3,4);write(5,6);
```

輸出為：


```
123456
```

```
writeln(1,2,3,4);write(5,6);
```

輸出為：

```
1234  
56
```


四、應用例析

 **例 2.1：**某倉庫 5 月 1 日有糧食 100 噸，5 月 2 日又調進 20 噸，5 月 3 日賣出庫存的 3 分之二，5 月 4 日又調進庫存的 3 倍糧食，問該倉庫從 5 月 1 日到 5 月 4 日期間每天的糧食分別是多少噸？（輸出每天的庫存量）

分析：在這個問題中，主要要描述從 5 月 1 日到 5 月 4 日期間倉庫的糧食庫存量，且易知它是不斷變化的。因此我們可以用一個變量 A 來描述倉庫的糧食庫存量。

程序可寫如下：

```
Program ex2_1;  
Var A:integer;  
Begin  
  A:=100; Writeln('5/1:',A);  
  A:=A+20; Writeln('5/2:',A);  
  A:=A div 3; writeln('5/3:',A);  
  A:=A*4; writeln('5/4:',A); Readln;  
End.
```

 **例 2.2：**有三個小朋友甲乙丙。甲有 50 粒糖果，乙有 43 粒糖果，丙有 13 粒糖果。現在他們做一個遊戲。從甲開始，將自己的糖分三份，自己留一份，其餘兩份分別給乙與丙，多餘的糖果自己吃掉，然後乙與丙也依次這樣做。問最後甲、乙、丙三人各有書多少粒糖果？

分析：這個問題中我們關心的是在遊戲過程中每個小朋友的糖果個數，且他們所擁有的糖果數是在變化的。因此可用 a, b, c 三個變量分別存放甲乙丙三個小朋友在某一時刻所擁有的糖果數。對於每人，分糖後，他的糖果數一定為原來的糖果數 div 3（因為分糖過程糖果的數目不一定都剛好分完，用整除恰恰可以表示多餘的糖自己吃掉）。而其他兩人則增加與這個小朋友現在擁有的一樣的糖果。

程序可寫如下：

```
program ex2_2;  
var A,B,C:integer;  
begin  
  A:=50; B:=43; C:=13; {初始時每個小朋友所擁有的糖果數}  
  A:=A div 3; B:=B+A; C:=C+A; {甲小朋友分糖果後，每個人擁有的糖果數變化情況}  
  B:=B div 3; A:=A+B; C:=C+B; {乙小朋友分糖果後，每個人擁有的糖果數變化情況}  
  C:=C div 3; A:=A+C; B:=B+C; {丙小朋友分糖果後，每個人擁有的糖果數變化情況}  
  writeln('A=',A,'B=',B,'C=',C); {輸出結果}  
  readln;  
end.
```

注：上程序中倒數第三行中'A='表示一個字符串（即用一對單引號括起來的一串字符），對於字符串，輸出字符串的內容（即引號內的所得字符，而引號不輸出）。

以上程序的運行結果為：

```
A=51B=35C=16
```

練習二

1、已知某梯形的上底 $A=13$ ，下底 $B=18$ ，高 $H=9$ ，求它的面積 S 。

2、某機關組織游泳比賽。將一堆西瓜分給前三名，把該堆西瓜中的一半又半個西瓜獎給第一名；剩下的一半又半個西瓜給第二名；把最後剩下的一半又半個西瓜給第三名，但每次分時並沒切開任何一個西瓜，且剛好西瓜分完。問前三名各分到多少個西瓜

```
var n1,n2,n3,x:real; {n1,n2,n3 為第 1,2,3 名分到的西瓜數; x 為總西瓜數}
    ok:boolean;
begin
    n3:=1;
    ok:=false; {ok 為 true 時表示已找到答案}
    while not ok do
    begin
        x:=4*n3+3; {x/2+0.5=n1, (x-n1)/2+0.5=n2, n1+n2+n3=x => x=4*n3+3}
        n1:=x/2+0.5;
        n2:=(x+1)/4; {x/2+0.5=n1, (x-n1)/2+0.5=n2 => n2=(x+1)/4}
        if (n1-trunc(n1)=0) and (n2-trunc(n2)=0) and ((x-n1-n2)/2+0.5=n3) then
            ok:=true
        else
            n3:=n3+1;
    end;
    writeln('x:',x:0:0,' n1:',n1:0:0,' n2:',n2:0:0,' n3:',n3:0:0);
    readln;
end.
```

3、已知某圓的半徑 $R=139$ ，求該圓的周長 C 與面積 S ？

第三課 帶格式的輸出語句及輸入語句

一、寫語句的輸出格式

在 pascal 語言中輸出數據時是可以按照一定格式的，對整數隱含的輸出形式為按十進制數形式。對實數的輸出，隱含的形式是科學記數法形式（如果不想用科學記數法輸出而用小數形式輸出，要自己另行定義）。

事實上，輸出語句中的每個輸出項中的表達式之後可以加上格式說明，若輸出項後沒有加格式說明，則數據按系統隱含的格式輸出，還可加上一定格式符號按特定格式輸出。

1. 隱含的輸出格式

pascal 語言為整型量、實型量、布爾型量和字符串（用一對單引號括起來的字符序列）規定了每種數據所占的寬度（即一個數據占幾列），一個數據所占的寬度稱為“場寬”或“字段寬”。系統給出的隱含場寬稱為標準場寬。每一種 pascal 版本給定的標準場寬不盡相同。下表給出標準 pascal 和 pc 機上兩種 pascal 版所規定的標準場寬。

數據類型	標準 pascal	Turbo pascal
Integer	10	實際長度
Real	22	17
布爾型	10	4 或 5
字符串	串長	串長

在 Turbo Pascal 系統中，對於整型字符串的輸出都是按數據本身長度輸出，對於布爾型數據（只有 True 和 False 兩種值），TRUE 為 4 列，FALSE 為 5 列，一律採用大寫輸出。而 real 型數據的輸出時，則按 17 列輸出，其中第一列為符號位，正號不顯示，後四位為“E±nn”，中間的 12 列為尾數部分。如：

```
writeln(sqrt(75));
```

則輸出

```
□8.6602540379E+00。
```

而

```
writeln(sqrt(81));
```

則輸出

```
□9.0000000000E+00。
```

有時，在程序中往往根據實際情況，需要自己定義場寬。

2. 指定場寬

在寫語句中輸出項含有格式符號時，就是為了指定場寬。

(1) 指定單場寬。

格式：write(表達式:N) 或 writeln(表達式:N)，其中 N 為自然數，指定單場寬後，所有數據不再按標準場寬輸出，而按指定場寬輸出。若數據實際長度小於指定場寬時，則一律“向右靠齊，左留空格”。如：

```
write(1234:8);write('abcdef':12)
```

輸出結果：

```
□□□□1234□□□□□□abcdef
```

對於標準實型數據指定單場寬時，如果場寬大於標準場寬時，右靠齊按標準場寬格式輸出 17 位，左留空格。若場寬小於標準場寬時，第一位仍為符號位，最後四位仍為“E±nn”，中間部分為尾數顯示部分。如果指定的寬度小於 8 位，則數據按 8 位格式“±*. *E±nn”輸出（若第一位符號位為正號則以空格顯示）。

(2) 指定雙場寬

如果輸出項是實數時，如果希望輸出的實數不用科學記數法輸出，而用小數形式輸出，可以用指定雙場寬方法輸出。

PASCAL 教程

雙場寬輸出格式為：`write(實型表達式:m:n)`，其中 `m` 和 `n` 都是自然數，`m` 用以指定整個數據所占的寬度，`n` 指定輸出實數的小數位數。如：

```
write(sqrt(75):9:4);
```

輸出：

```
□□□8.6602
```

如果雙場寬不能滿足輸出數據的最低要求，系統自動突破指定的場寬限制，按實際長度輸出。

如：`write(sqrt(75):5:4)`；要使小數點後有 4 位數字，而總場寬為 5，是不可能的（因為還有一個小數點，小數點前面還有一個數字）。它最低限度要有 6 列，即輸出為：8.6602

📖 例 3.1：寫出下列程序在 `turbo pascal` 下的輸出結果

```
program ex3_1;
  const s='abcdefg';
  var
    i:integer;
    r:real;
    c:char;b:boolean;
  begin
    i:=1234;r:=1234.5678;
    c:'#';b:=true;
    writeln(i,i:6,i:3);
    writeln(r,r:12:5,r:8:5);
    writeln(c,c:5);
    writeln(s,s:10,s:5);
    writeln(b,b:5,b:3);
  end.
```

運行結果如下：

```
1234□□12341234
□1.2345678000E+03□□1234.567801234.56780
#□□□□#
abcdefg□□□abcdefgabcdefg
TRUE□TRUETRUE
```

3·應用例析

📖 例 3.2：已知 `A=253`，`B=43`，輸出 `A*B` 的運算式子。即輸出如下：

```
253*43=10879
253
* 43
759
+1012
10879
```

分析：對於該問題，我們只要控制好輸出時右靠齊即可。即前四行的總寬度一樣（例如為 12），第五行總寬度比前面少 1。第六、七行總寬度與前四行一樣。

參與程序如下：

```
program ex3_2;
var a,b:integer;
begin
  a:=253;b:=43;
  writeln(a:10,'*',b,'=',a*b);
  writeln(a:12);
  write('*':8);writeln(b:4);
  writeln('-----':12);
  writeln(a*3:12);
  write('+':6);writeln(a*4:5);
  writeln('-----':12);
  writeln(a*b:12);
end.
```

二、輸入語句（讀語句）

在程序中變量獲得一個確定的值，固然可以用賦值語句，但是如果需要賦值的變量較多，或變量的值經常變化，則使用本節介紹的輸入語句——讀語句，將更為方便。讀語句是在程序運行時由用戶給變量提供數據的一種很靈活的輸入動作，它有兩種格式：

讀語句的一般格式：

```
read(<變量名表>);  
readln[(<變量名表>)];
```

其中變量名表是用逗號隔開的若干個變量名組成的。


功能：從標準輸入文件(即 INPUT，一般對應著鍵盤)中讀入數據，並依次賦給相應的變量。

說明：

(1) read 和 readln 是標準過程名，它們是標準標識符。

(2) 執行到 read 或 readln 語句時，系統處於等待狀態，等待用戶從鍵盤上輸入數據，系統根據變量的數據類型的語法要求判斷輸入的字符是否合法。如執行 read(a) 語句，a 是整型變量，則輸入的字符為數字字符時是合法的，當輸入結束時，則自動將剛接受的一串數字字符轉換為整數賦給變量 a。

(3) 在輸入數值型(整型或實型)數據時，數據間要用空格或回車分隔開各個數據，輸入足夠個數的數據，否則仍要繼續等待輸入，但最後一定要有回車，表示該輸入行結束，直到數據足夠，該讀語句執行結束，程序繼續運行。

 **例 3.3**：設 a、b、c 為整型變量，需將它們的值分別賦以 10，20，30，寫出對應下列語句的所有可能輸入格式。
Read(a,b,c);

根據(3)，即可列出所有可能輸入格式

```
(a) 10□20□30←  
(b) 10□20←  
30←  
(c) 10←  
20□30←  
(d) 10←  
20←  
30←
```

其中“←”表示回車鍵。下同。

(4) read 語句與 readln 語句的第一個區別是：

read 語句是一個接一個地讀數據，在執行完本 Read 語句(讀完本語句中變量所需的數據)後，下一個讀語句接著從該數據輸入行中繼續讀數據，也就是說，不換行。如：


```
Read(a,b);  
Read(c,d);  
Read(e);
```

如果輸入數據行如下：

```
1□2□3□4□5□6□←
```


則 a，b，c，d，e 的值分別為 1，2，3，4，5，如果後面無讀語句則數據 6 是多餘的，這是允許的。

Readln 則不同，在讀完本 Readln 語句中變量所需的數據後，該數據行中剩餘的數據多餘無用，或者說，在讀完本 Readln 語句中變量所需數據後，一定要讀到一個回車，否則多餘的數據無用。

 **例 3.4**：設要達到例 3.3 同樣的目的，但語句改為：
readln(a,b);readln(c)

則例 3.3 中的 4 種輸入格式只有(b)(d)是有效的。

(5) readln 語句與 read 語句的第二個區別是：read 後一定要有參數表，而 readln 可以不帶參數表，即可以沒有任何輸入項，只是等待讀入一個換行符(回車)。經常用於暫停程序的運行，直到輸入一個回車。

 **例 3.5**：設有下列語句：

```
read(a,b,c);  
readln(d,e);  
readln;  
readln(f,g);
```

其中，所有變量均為整型。再設輸入的數據如下：

```
1□2←
```

```
3 4 5 6 7 8
9 10
11
12 13
列表給出每個變量的值。
```

分析：可以假想有一“數據位置指針”，每讀一個數據後，指針後移到該數據之後，每執行一個 readln 語句後，指針移到下一個數據行的開頭。

各變量的值如下表所示。

變量名	a	b	c	d	e	F	g
值	1	2	3	4	5	11	12

(6) 為了避免可能出現的錯誤，建議在程序中按下列原則使用讀語句：(A) 如果沒有特殊需要，在一個程序中盡量避免混合使用 read 語句和 readln 語句；(B) 儘量用 readln 語句來輸入數據，一個數據行對應一個 readln 語句；(C) 由於執行 read 或 readln 語句時，系統不會提供任何提示信息，因此，編程時最好在 readln 語句之前加以適當提示，例如：

```
write('Input a,b,c:');
readln(a,b,c);
```


在執行時，屏幕上顯示：

```
Input a,b,c:■
```

其中，“■”為光標。執行 readln 語句後，系統處於待輸入狀態，只有輸入了所需數據後才繼續往下執行。

三、順序結構程序設計


到目前為止，我們可以用讀、寫語句和賦值語句編寫一些簡單的程序。通過閱讀這些程序，可以逐步熟悉 pascal 程序的編寫方法和應遵循的規則，為以後各章的學習打基礎。

 **例 3.6：** 試編一程序，輸入一梯形的上底、下底、高，求該梯形的面積。

分析：整個程序分為三段：輸入、計算、輸出。程序中用 a, b, h 三個變量分別存放梯形的上、下底與高，s 存放面積。要而使用這些變量都要先說明，程序的執行部分中先輸入上、下底與高，接著求面積 s，最後輸出結果 s。

源程序如下：

```
program ex3_6; {程序首部}
var a,b,h,s:real; {程序說明部分}
begin
  write('Input a,b,h:');
  readln(a,b,h); {程序執行部分}
  s:=(a+b)*h/2;
  write('s=',s:10:3);
end.
```

 **例 3.7：** 某幼兒園裏，有 5 個小朋友編號為 1, 2, 3, 4, 5，他們按自己的編號順序圍坐在一張圓桌旁。他們身上都有若干個糖果，現在他們做一個分糖果遊戲。從 1 號小朋友開始，將他的糖果均分三份（如果有多餘的，則他將多餘的糖果吃掉），自己留一份，其餘兩份分給他的相鄰的兩個小朋友。接著 2 號、3 號、4 號、5 號小朋友也這如果做。問一輪後，每個小朋友手上分別有多少糖果。

分析：這道問題與第二課中的例 2 基本一樣，只不過這裏有 5 位小朋友，且他們初始時糖果的數目不確定。這裏用 a, b, c, d, e 分別存放 5 個小朋友的糖果。初始時它們的值改為由鍵盤輸入。其它都與第二課中的例 2 類似。

參考程序如下：

```
program ex3_7;
var a,b,c,d,e:integer;
begin
  write('Please Enter init numbers ');readln(a,b,c,d,e);
  a:=a div 3;b:=b+a;e:=e+a; {1 號均分後，1、2、5 號的糖果數變化情況}
  b:=b div 3;c:=c+b;a:=a+b; {2 號均分後，1、2、3 號的糖果數變化情況}
  c:=c div 3;b:=b+c;d:=d+c; {3 號均分後，2、3、4 號的糖果數變化情況}
  d:=d div 3;c:=c+d;e:=e+d; {4 號均分後，3、4、5 號的糖果數變化情況}
  e:=e div 3;d:=d+e;a:=a+e; {5 號均分後，4、5、1 號的糖果數變化情況}
  {輸出結果}
  writeln('a=',a);
```

```
writeln('b=',b);  
writeln('c=',c);  
writeln('d=',d);  
writeln('e=',e);  
readln; {暫停}  
end.
```

例 3.8 : 編一程序求半徑為 R 的圓的周長與面積?

分析：程序要先輸入半徑 R ，然後求周長 c 和面積 s ，最後輸出 c 和 s 。

源程序如下：

```
program ex3_8;  
const PI=3.14159;  
var r,c,s:real;  
begin  
  write('Enter R=');readln(r);  
  c:=2*pi*r;  
  s:=pi*sqr(r);  
  writeln('c=',c:10:2);  
  writeln('s=',s:10:2);  
end.
```

在程序中，為了輸出實型周長 c 和面積 s 時，按照小數形式輸出，採用了指定雙場寬格式。

練習三

1、編一程序，將攝氏溫度換為華氏溫度。公式為： $f = \frac{9}{5}c + 32$

其中 f 為華氏溫度， c 是攝氏溫度。

2、編一程序，輸入三角形的三邊 a 、 b 、 c （假設這三邊可以構成一個三角形），求三角形的面積 S ？

提示：可利用海倫公式 $s = \sqrt{p(p-a)(p-b)(p-c)}$, $p = \frac{a+b+c}{2}$

第四課 簡單的分支結構程序設計

在現實生活中，我們每天都要進行根據實際情況進行選擇。例如，原打算明天去公園，但如果明天天氣不好，將留在家裏看電視。所以人也會根據條件進行行為的選擇。計算機也會根據不同情況作出各種邏輯判斷，進行一定的選擇。在這課與下一課中，我們將會發現，我們是通過選擇結構語句來實現程序的邏輯判斷功能。

一、PASCAL 中的布爾（邏輯）類型

在前面，我們學習了整型 (integer) 與實型 (real)。其中 integer 型數據取值範圍為 -32768 到 32767 之間所有整數。而 real 型數據取值範圍為其絕對值在 10⁻³⁸ 到 10³⁸ 之間的所有實數。它們都是數值型的（即值都為數）。布爾型 (Boolean) 是一種數據的類型，這種類型只有兩種值，即“真”與“假”。

1. 布爾常量

在 Pascal 語言中“真”用 true 表示，“假”用 False 表示。所以布爾類型只有 TRUE 與 FALSE 兩個常量。

2. 布爾變量 (BOOLEAN)

如果我們將某些變量說明成布爾型，那麼這些變量就是布爾變量，它們只能用於存放布爾值 (ture 或 false)。

例如，VAR A, B: BOOLEAN;

3. 布爾類型是順序類型

由於這種類型只有兩個常量，Pascal 語言中規定 true 的序號為 1，false 的序號為 0。若某種類型的常量是有限的，那麼這種類型的常量通常都有一個序號，我們稱這種類型為順序類型。如前面我們學過的整型 (integer)，以及後面要學到的字符型 (char) 都是順序類型。

4. 布爾類型的輸入與輸出

(1) 輸出

```
var a,b:boolean;  
begin  
  a:=true;b:=false;  
  writeln(a,b);  
end.
```

結果:TRUEFALSE

(2) 布爾類型變量不能直接用讀語句輸入

布爾類型變量不能通過讀語句給它們提供值。事實上，我們可以通過間接方式對布爾變量進行值的輸入。例如，以下程序是錯誤的：

```
var a,b,c:Boolean;  
begin  
  readln(a,b,c); {錯誤語句}  
  writeln(a,b,c);  
end.
```

二、關係表達式與布爾表達式

1. 什麼是關係表達式

用小括號、>、<、>=、<=、=、<>將兩個算術表達式連接起來的式子就稱為關係表達式（比較式）。

如：3+7>8，x+y<10，2*7<=13 等都是關係表達式。

2. 關係表達式的值

很顯然，這幾個關係表達式中第一個是正確的，第三個是錯誤的，而第二個表達式可能是對的，也可能是錯的。所以我們很容易發現，這些表達式的值是“對”的或“不對”的（或者說，是“真”的或“假”的），即關係表達式的值為布爾值。表示該比較式兩端式子的大小關係是否成立。如 3+2>6 是錯的，故它的值為 FALSE。同樣，45>=32 是對的，故該表達式的值為 true。

關係表達式用於表示一個命題。如：“m 為偶數”可表示為：m mod 2=0。“n 為正數”可表示為：n>0。

3. 布爾運算及布爾表達式

PASCAL 教程

為了表示更複雜的命題，Pascal 還引入三種邏輯運算符：not、and、or。它們分別相當於數學上的“非”、“且”和“或”的意義。

這三個運算符的運算對象為布爾量，其中 not 為單目運算，只有一個運算對象，and 與 or 為雙目運算，有兩個運算對象。它們的運算真值表如下：

A	b	Not a	a and b	a or b	a xor b
false	false	true	false	false	False
false	true	true	false	ture	True
True	false	false	false	true	True
True	true	false	true	true	False

於是，對於一個關係表達式，或多個關係表達式用布爾運算符連接起來的式子就稱為布爾表達式。布爾表達式的值也為布爾值。

如果一個表達式裏出現兩個或兩個以上的運算符，則必須規定它們的運算次序。pascal 規定：

- (1) 表達式中相同優先級的運算符，按從左到右順序計算；
- (2) 表達式中不同優先級的運算符，按從高到低順序計算；
- (3) 括號優先級最高，從內到外逐層降低；

對於一個複雜的表達式可能同時包含算術運算、關係運算和邏輯運算以及函數運算。

運算的優先順序為：括號>函數>not>*、/、div、mod、and>+、-、or、xor>關係運算。

對於複雜的命題，我們可以用布爾表達式來表示。例如，命題：“m，n 都是偶數或都是奇數”可表示為“(m mod 2=0) and (n mod 2=0) or (m mod 2=1) and (n mod 2=1)”。

三、簡單的 IF 語句

1. 格式

I 型：

IF <布爾表達式> THEN 語句;

II 型：

IF <布爾表達式> THEN 語句 1 ELSE 語句 2;

(注意 IF 語句中語句 1 後無“;”號)

2. 功能

- (1) 執行沒有 ELSE 的 IF 語句時，先計算<布爾表達式>的值，若為 TRUE 則執行語句，否則不執行任何操作。
- (2) 執行有 ELSE 的 IF 語句時，先計算<布爾表達式>的值，若為 TRUE 則執行語句 1，否則執行語句 2；

3. 示例

 **例 4.1**：輸入一個整數 a，判斷是否為偶數。（是輸出“y e s”否則輸出“n o”）。

```
Program ex4_1;
Var a:integer;
Begin
  Write('a='); readln(a);
  If (a mod 2 =0) then writeln('yes')
  Else writeln('no');
  Readln;
End.
```

 **例 4.2**：華榕超市里賣電池，每個電池 8 角錢，若數量超過 10 個，則可打 75 折。

```
Program ex4_2;
var Num:integer;Price,Total:real;
Begin
  Write('Num=');readln(Num);
```

```
Price:=0.8;
If Num>10 then Price:=Price*0.75;
Total:=Num*Price;
Writeln('Total=',Total:0:2);
Readln;
End.
```

例 4.3：編寫一與電腦猜“紅”或“黑”的遊戲。

分析：用 1 代表紅，0 代表黑。先由計算機先出答案，然後再由人猜，猜對輸出“YOU WIN”否則輸出“YOU LOST”。為了模擬猜“紅”或“黑”的隨意性，程序中需要用到隨機函數 `random(n)`。函數是什麼呢，例如大家都知道 $|-2|=2$ ， $|58|=58$ ，那麼 $|x|=?$ 。如果我們用 y 表示 $|x|$ ，那麼 $y=|x|$ 就是一個函數，也就是說函數是一個關於一個或多個自變量（未知量，如上例中的 x ）的運算結果。

在 pascal 語言中，系統提供了許多內部函數，其中包括 $|x|$ 函數，當然它用 `abs(x)` 表示。我們如果要求 x^2-y 的絕對值，可以調用內部函數 `abs(x*x-y)` 即可求得。`Random(n)` 也是一個內部函數，調用它能得到 $0\sim n-1$ 之間的整數（但它不確定的，或說是隨機的）。同時由於函數是一個運算結果，所以函數的調用只能出現在表達式中。

```
Program ex4_3;
Uses crt;
Var Computer,People:integer;
Begin
  Randomize;
  Computer:=random(2);
  Write('You guess (0-Red 1-Black) :');readln(People);
  If People=Computer then writeln('YOU WIN')
  Else writeln('YOU LOST');
  Readln;
End.
```

練習四

- 1、某車站行李托運收費標準是：10 公斤或 10 公斤以下，收費 2.5 元，超過 10 公斤的行李，按每超過 1 公斤增加 1.5 元進行收費。試編一程序，輸入行李的重量，算出托運費。

第五課 if 嵌套與 case 語句

一、if 語句的嵌套

在 if 語句中，如果 then 子句或 else 子句仍是一個 if 語句，則稱為 if 語句的嵌套。

📖 例 5.1：計算下列函數

$$y = \begin{cases} 1 & , \quad x > 0 \\ 0 & , \quad x = 0 \\ -1 & , \quad x < 0 \end{cases}$$

分析：根據輸入的 x 值，先分成 $x > 0$ 與 $x \leq 0$ 兩種情況，然後對於情況 $x \leq 0$ ，再區分 x 是小於 0，還是等於 0。

源程序如下：

```
program ex5_1a;
var
  x:real;
  y:integer;
begin
  writeln('Input x:');readln(x);
  if x>0 then y:=1 {x>0 時，y 的值为 1}
  else {x≤0 時}
    if x=0 then y:=0
    else y:=-1;
  writeln('x=',x:6:2,'y=',y);
end.
```

顯然，以上的程序中，在 then 子句中嵌套了一個 II 型 if 語句。當然程序也可以寫成如下形式：

```
program ex5_1b;
var
  x:real;y:integer;
begin
  writeln('Input x:');readln(x);
  if x>=0 then
    if x>0 then y:=1
    else y:=0
  else y:=-1;
  writeln('x=',x:6:2,'y=',y);
end.
```

但是對於本題，下面的程序是不對的。

```
y:=0;
if x>=0 then
  if x>0 then y:=1
else y:=-1;
```

明顯，從此人的程序書寫格式可以看出，他想讓 else 與第一個 if 配對，而事實上，這是錯的。因為 pascal 規定：else 與它上面的距它最近的 then 配對，因此以上程序段的邏輯意義就與題義不符。

要使上程序段中 else 與第一個 then 配對，應將程序段修改為：

```
y:=0;
if x>=0 then
begin
  if x>0 then y:=1
end
else y:=-1;
```

二、case 語句

上面我們知道可以用嵌套的 if 語句實現多分支的選擇結構。但是如果分支越來越多時，用嵌套的 if 語句實現多分支就顯得繁雜。當多分支選擇的各個條件由同一個表達式的不同結果值決定時，可以用 case 語句實現。它的選擇過程，很象一個多路開關，即由 case

PASCAL 教程

語句的選擇表達式的值，決定切換至哪一語句去工作。因此在分支結構程序設計中，它是一種強有力的手段。在實現多路徑分支控制時，用 case 對某些問題的處理和設計，比用 if 語句寫程序具有更簡潔、清晰之感。

1. 情況語句的一般形式：

```
case <表達式> of
  <情況標號表 1> : 語句 1;
  <情況標號表 2> : 語句 2;
  :
  :
  <情況標號表 n> : 語句 n;
[else 語句;]
end;
```

其中 case、of、end 是 Pascal 的保留字，表達式的值必須是順序類型，它可以是整型、布爾型及以後學習的字符型、枚舉型和子界型。情況標號表是一串用逗號隔開的與表達式類型一致的常量序列。語句可以是任何語句，包括複合語句和空語句。

2. case 語句的執行過程

先計算表達式（稱為情況表達式）的值，如果它的值等於某一個常量（稱為情況常量，也稱情況標號），則執行該情況常量後面的語句，在執行完語句後，跳到 case 語句的末尾 end 處。

3. 說明

- (1) 情況表達式必須是順序類型的；
- (2) 情況常量是情況表達式可能具有的值，因而應與情況表達式具有相同的類型；
- (3) 情況常量出現的次序可以是任意的；
- (4) 同一情況常量不能在同一個 case 語句中出現兩次或兩次以上；
- (5) 每個分語句前可以有一個或若干個用逗號隔開的情況常量；
- (6) 如果情況表達式的值不落在情況常的範圍內，則認為本 case 語句無效，執行 case 語句的下一個語句。turbo pascal 中增加了一個“否則”的情況，即增加一個 else 子句，但也是可省的。
- (7) 每個常量後面只能是一個語句或一個複合語句。

 **例 5.2**：根據 x 的值，求函數 y 的值：


$$y = \begin{cases} x+1 & , \quad 0 < x < 100 \\ x-1 & , \quad 100 \leq x < 200 \\ -1 & , \quad \text{其餘} \end{cases}$$

分析：利用 case 語句進行程序設計，關鍵在於巧妙地構造情況表達式。本例中三種情況可用一個表達式區分出來： $\text{Trunc}(x/100)$ 。因為 x 在 $(0 \sim 100)$ 之間時表達式值為 0； x 在 $[100, 200)$ 時表達式值為 1；其餘部分可用 else 子句表示。

源程序如下：

```
program ex5_2;
var x,y:real;
begin
  write('Input x:'); readln(x);
  case trunc(x/100) of
    0:y:=x+1;
    1:y:=x-1;
    else y:=0;
  end; {end of case}
  writeln('x=',x:8:2,'y=',y:8:2);
end.
```

三、選擇結構的程序設計

 **例 5.3**：輸入一個年號，判斷它是否是閏年。

分析：判斷閏年的算法是：如果此年號能被 400 除盡，或者它能被 4 整除而不能被 100 整除，則它是閏年。否則，它是平年。

源程序如下：

```
program ex5_3;
var year:integer;
begin
  write('Input year:');readln(year);
  write(year:6);
  if (year mod 400=0 ) then
    writeln('is a leap year.')
  else
    if (year mod 4=0)and(year mod 100<>0) then
      writeln('is a leap year.')
    else writeln('is not a leap year.');
```

例 5.4 ：判斷 1995 年，每個月份的天數。

分析：程序分為：輸入月份，計算該月的天數，輸出天數

源程序如下：

```
program ex5_4;
var month,days:integer;
begin
  write('Input month:');readln(month);
  case month of
    1,3,5,7,8,10,12:days:=31;
    4,6,9,11 :days:=30;
    2 :days:=28;
    else days:=0;
  end;
  if days<>0 then writeln('Days=',days);
end.
```

例 5.5 ：期末來臨了，班長小 Q 決定將剩餘班費 x 元錢，用於購買若干支鋼筆獎勵給一些學習好、表現好的同學。已知商店裏有三種鋼筆，它們的單價為 6 元、5 元和 4 元。小 Q 想買儘量多的筆（鼓勵儘量多的同學），同時他又不想有剩餘錢。請您編一程序，幫小 Q 制訂出一種買筆的方案。

分析：對於以上的實際問題，要買儘量多的筆，易知都買 4 元的筆肯定可以買最多支筆。因此最多可買的筆為 $x \div 4$ 支。由於小 Q 要把錢用完，故我們可以按以下方法將錢用完：

若買完 $x \div 4$ 支 4 元錢的筆，還剩 1 元，則 4 元錢的筆少買 1 支，換成一支 5 元筆即可；若買完 $x \div 4$ 支 4 元錢的筆，還剩 2 元，則 4 元錢的筆少買 1 支，換成一支 6 元筆即可；若買完 $x \div 4$ 支 4 元錢的筆，還剩 3 元，則 4 元錢的筆少買 2 支，換成一支 5 元筆和一支 6 元筆即可。

從以上對買筆方案的調整，可以看出筆的數目都是 $x \div 4$ ，因此該方案的確為最優方案。

源程序如下：

```
program ex5_5;
var a,b,c:integer; {a,b,c 分別表示在買筆方案中，6 元、5 元和 4 元錢筆的數目}
    x,y:integer; {x,y 分別表示剩餘班費和買完最多的 4 元筆後剩的錢}
begin
  write('x='); readln(x) {輸入 x}
  c:=x div 4; {4 元筆最多買的數目}
  y:=x mod 4; {求買完 c 支 4 元筆後剩餘的錢數 y}
  case y of
    0 : begin a:=0; b:=0; end;
    1 : begin a:=0; b:=1; c:=c-1; end;
    2 : begin a:=1; b:=0; c:=c-1; end;
    3 : begin a:=1; b:=1; c:=c-2; end;
  end;
  writeln('a=',a,'b=',b,'c=',c);
end.
```

練習五

1、輸入三角形的三個邊，判斷它是何類型的三角形（等邊三角形？等腰三角形？一般三角形？）。

2、輸入三個數，按由大到小順序打印出來。

3、計算 1901 年 2099 年之間的某月某日是星期幾。

4、輸入兩個正整數 a ， b 。 b 最大不超過三位數， a 不大於 31。使 a 在左， b 在右，拼接成一個新的數 c 。

例如： $a=2$ ， $b=16$ ，則 $c=216$ ；若 $a=18$ ， $b=476$ ，則 $c=18476$ 。

提示：求 c 的公式為：

$$c=a \times K+b$$

其中：

$$K = \begin{cases} 10 & , \text{ 當 } B \text{ 為一位數時 } (0 < B < 10) \\ 100 & , \text{ 當 } B \text{ 為二位數時 } (10 \leq B < 100) \\ 1000 & , \text{ 當 } B \text{ 為三位數時 } (100 \leq B < 1000) \end{cases}$$

第六課 for 循環語句

在實際應用中，會經常遇到許多有規律性的重複運算，這就需要掌握本章所介紹的循環結構程序設計。在 Pascal 語言中，循環結構程序通常由三種的循環語句來實現。它們分別為 FOR 循環、當循環和直到循環。通常將一組重複執行的語句稱為循環體，而控制重複執行或終止執行由重複終止條件決定。因此，重複語句是由循環體及重複終止條件兩部分組成。

一、for 語句的一般格式

```
for <控制變量>:=<表達式 1> to <表達式 2> do <語句>;  
for <控制變量>:=<表達式 1> downto <表達式 2> do <語句>;
```

其中 for、to、downto 和 do 是 Pascal 保留字。表達式 1 與表達式 2 的值也稱為初值和終值。

二、For 語句執行過程

- (1) 先將初值賦給左邊的變量（稱為循環控制變量）；
- (2) 判斷循環控制變量的值是否已“超過”終值，如已超過，則跳到步驟 (6)；
- (3) 如果未超過終值，則執行 do 後面的那個語句（稱為循環體）；
- (4) 循環變量遞增（對 to）或遞減（對 downto）1；
- (5) 返回步驟 (2)；
- (6) 循環結束，執行 for 循環下面的一個語句。

三、說明

- (1) 循環控制變量必須是順序類型。例如，可以是整型、字符型等，但不能為實型。
- (2) 循環控制變量的值遞增或遞減的規律是：選用 to 則為遞增；選用 downto 則遞減。
- (3) 所謂循環控制變量的值“超過”終值，對遞增型循環，“超過”指大於，對遞減型循環，“超過”指小於。
- (4) 循環體可以是一個基本語句，也可以是一個複合語句。
- (5) 循環控制變量的初值和終值一經確定，循環次數就確定了。但是在循環體內對循環變量的值進行修改，常常會使得循環提前結束或進入死環。建議不要在循環體中隨意修改控制變量的值。
- (6) for 語句中的初值、終值都可以是順序類型的常量、變量、表達式。

四、應用舉例

📖 例 6.1：輸出 1—100 之間的所有偶數。

```
program ex6_1;  
var i:integer;  
begin  
  for i:=1 to 100 do  
    if i mod 2=0 then write(i:5);  
end.
```

📖 例 6.2：求 $N! = 1 * 2 * 3 * \dots * N$ ，這裏 N 不大於 10。

分析：程序要先輸入 N ，然後從 1 累乘到 N 。程序如下：

```
program ex6_2  
var  
  n,i : integer; {i 為循環變量}  
  S : longint; {s 作為累乘器}  
Begin  
  write('Enter n='); readln(n); {輸入 n}  
  s:=1;  
  for i:=2 to n do s:=s*i; {從 2 到 n 累乘到 s 中}  
  writeln(n,'!=',s); {輸出 n! 的值}  
end.
```

練習六

1、求 $s=1+4+7+\dots+298$ 的值。

2、編寫一個評分程序，接受用戶輸入 10 個選手的得分 (0-10 分)，然後去掉一個最高分和一個最低分，求出某選手的最後得分 (平均分)。

3、用一張一元票換 1 分、2 分和 5 分的硬幣，每種至少一枚，問有哪幾種換法（各幾枚）？

第七課 WHILE 與 REPEAT 循環

一、WHILE 循環

對於 for 循環有時也稱為計數循環，當循環次數未知，只能根據某一條件來決定是否進行循環時，用 while 語句或 repeat 語句實現循環要更方便。

while 語句的形式為：


while <布爾表達式> do <語句>;

其意義為：當布爾表達式的值為 true 時，執行 do 後面的語句。

while 語句的執行過程為：

- (1) 判斷布爾表達式的值，如果其值為真，執行步驟 2，否則執行步驟 4；
- (2) 執行循環體語句 (do 後面的語句)；
- (3) 返回步驟 1；
- (4) 結束循環，執行 while 的下一個語句。


說明：這裏 while 和 do 為保留字，while 語句的特點是先判斷，後執行。當布爾表達式成立時，重複執行 do 後面的語句 (循環體)。

 **例 7.1**：求恰好使 $s=1+1/2+1/3+\dots+1/n$ 的值大於 10 時 n 的值。

分析：“恰好使 s 的值大於 10”意思是當表達式 s 的前 n-1 項的和小於或等於 10，而加上了第 n 項後 s 的值大於 10。從數學角度，我們很難計算這個 n 的值。故從第一項開始，當 s 的值小於或等於 10 時，就繼續將下一項值累加起來。當 s 的值超過 10 時，最後一項的項數即為要求的 n。

程序如下：

```
program ex7_1;
Var
  s : real;
  n : integer; {n 表示項數}
begin
  s:=0.0; n:=0;
  while s<=10 do {當 s 的值還未超過 10 時}
  begin
    n:=n+1; {項數加 1}
    s:=s+1/n; {將下一項值累加到 s}
  end;
  writeln('n=',n); {輸出結果}
end.
```

 **例 7.2**：求兩個正整數 m 和 n 的最大公約數。

分析：求兩個正整數的最大公約數採用的輾轉相除法求解。以下是輾轉的算法：

分別用 m, n, r 表示被除數、除數、餘數。

- (1) 求 m/n 的餘數 r。
- (2) 若 r=0，則 n 為最大公約數。若 r≠0，執行第 (3) 步。
- (3) 將 n 的值放在 m 中，將 r 的值放在 n 中。
- (4) 返回重新執行第 (1) 步。

程序如下：

```
program ex7_2a;
var m,n,a,b,r:integer;
begin
  write('Input m,n:');
```

```
readln(m,n);
a:=m; b:=n; r:=a mod b;
while r<>0 do
begin
  a:=b;b:=r;
  r:=a mod b;
end;
writeln('The greatest common divide is:',b:8);
end.
```

二、直到循環（REPEAT—until 語句）

用 while 語句可以實現“當型循環”，用 repeat-until 語句可以實現“直到型循環”。repeat-until 語句的含義是：“重複執行循環，直到指定的條件為真時為止”。

直到循環語句的一般形式：

```
Repeat
  <語句 1>;
:
:
  <語句 n>;
until <布爾表達式>;
```

其中 Repeat、until 是 Pascal 保留字，repeat 與 until 之間的所有語句稱為循環體。

說明：

- (1) repeat 語句的特點是：先執行循環，後判斷結束條件，因而至少要執行一次循環體。
- (2) repeat-until 是一個整體，它是一個（構造型）語句，不要誤認為 repeat 是一個語句，until 是另一個語句。
- (3) repeat 語句在布爾表達式的值為真時不再執行循環體，且循環體可以是若干個語句，不需用 begin 和 end 把它們包起來，repeat 和 until 已經起了 begin 和 end 的作用。while 循環和 repeat 循環是可以相互轉化的。

對於例 7.2 中求兩個正整數的最大公約數，程序可用 repeat-until 循環實現如下：

```
program ex7_2b;
var
  m,n,a,b,r : integer;
begin
  write('Input m,n=');
  readln(m,n);
  a:=m; b:=n;
  repeat
    r:=a mod b;
    a:=b;b:=r;
  until r=0;
  writeln('The greatest common divide is',a);
end.
```

以上我們已介紹了三種循環語句。一般說來，用 for 循環比較簡明，只要能 use for 循環，就儘量用 for 循環。只在無法使用 for 循環時才用 while 循環和 repeat-until 循環，而且 while 循環和 repeat-until 循環是可以互相替代的。for 循環在大多數場合也能用 while 和 repeat-until 循環來代替。一般 for 循環用於有確定次數循環，而 while 和 repeat-until 循環用於未確定循環次數的循環。

當一個循環的循環體中又包含循環結構程序時，我們就稱之為循環嵌套。

三、循環結構程序設計

📖 例 7.3：求 $1!+2!+\dots+10!$ 的值。

分析：這個問題是求 10 自然數的階乘之和，可以用 for 循環來實現。程序結構如下：

```
for n:=1 to 10 do
begin
  求 n! 的值
  累加 n! 的值
End
```

顯然，通過 10 次的循環可求出 $1!$ ， $2!$...， $10!$ ，並同時累加起來，可求得 s 的值。而求 $T=n!$ ，又可以用一個 `for` 循環來實現：

```
t:=1;
for j:=1 to n do
  t:=t*j;
```

因此，整個程序為：


```
program ex7_3a;
var
  t,s:real;
  j,n:integer;
begin
  s:=0;
  for n:=1 to 10 do
  begin
    t:=1;
    for j:=1 to n do
      t:=t*j;
    s:=s+t;
  end;
  writeln('s=',s:0:0);
end.
```

以上的程序是一個二重的 `for` 循環嵌套。這是比較好想的方法，但實際上對於求 $n!$ ，我們可以根據求出的 $(n-1)!$ 乘上 n 即可得到，而無需重新從 1 再累乘到 n 。

程序可改為：

```
program ex7_3b;
var
  t,s:real;
  n:integer;
begin
  s:=0;t:=1;
  for n:=1 to 10 do
  begin
    t:=t*n;
    s:=s+t;
  end;
  writeln('s=',s:0:0);
end.
```


顯然第二個程序的效率要比第一個高得多。第一程序要進行 $1+2+\dots+10=55$ 次循環，而第二程序進行 10 次循環。如題目中求的是 $1!+2!+\dots+1000!$ ，則兩個程序的效率區別更明顯。

 **例 7.4：**一個炊事員上街採購，用 500 元錢買了 90 只雞，其中母雞一隻 15 元，公雞一隻 10 元，小雞一隻 5 元，正好把錢買完。問母雞、公雞、小雞各買多少只？

分析：設母雞 I 只，公雞 J 只，則小雞為 $90-I-J$ 只，則 $15*I+10*J+(90-I-J)*5=500$ ，顯然一個方程求兩個未知數是不能直接求解。必須組合出所有可能的 i, j 值，看是否滿足條件。這裏 I 的值可以是 0 到 33， J 的值可以 0 到 50。

源程序如下：

```
programr ex7_4;
var
  i,j,k:integer;
begin
  for i:=1 to 5 do
  for j:=1 to 8 do
  begin
    k:=90-i-j;
    if 15*i+10*j+5*k=500 then writeln(i:5,j:5,k:5);
  end;
end.
```

 **例 7.5：**求 100—200 之間的所有素數。

分析：我們可對 100—200 之間的每一整數進行判斷，判斷它是否為素數，是則輸出。而對於任意整數 i ，根據素數定義，我們從 2 開始，到 i ，找 i 的第一個約數。若找到第一個約數，則 i 必然不是素數。否則 i 為素數。

源程序如下：

```
program ex7_5;
Var
  i : integer;
  x : integer;
begin
  for i:=100 to 200 do
  begin
    x:=2;
    while (x<=trunc(sqrt(i)))and(i mod x<>0)do
    begin
      x:=x+1;
    end;
    if x>trunc(sqrt(i)) then write(i:8);
  end;
end.
```

練習七

- 1、輸入一個正整數 n ，將 n 分解成質因數冪的乘積形式。

例如： $36=2^2 \times 3^2$

- 2、輸出如下圖形。

```
      *           *           *
     ***         ***         ***
    *****     *****     *****
   *********   *********   *********
  ***********  ***********  ***********
 *****
*****
```

- 3、編寫一程序，驗證角穀猜想。所謂的角穀猜想是：“對於任意大於 1 的自然數 n ，若 n 為奇數，則將 n 變為 $3 \times n + 1$ ，否則將 n 變為 n 的一半。經過若干次這樣的變換，一定會使 n 變為 1。”

- 4、有一堆 100 多個的零件，若三個三個數，剩二個；若五個五個數，剩三個；若七個七個數，剩五個。請你編一個程序計算出這堆零件至少是多少個？

第八課 一維數組

一、為什麼要使用數組

例 8.1：輸入 50 個學生的某門課程的成績，打印出低於平均分的同學號數與成績。

分析：在解決這個問題時，雖然可以通過讀入一個數就累加一個數的辦法來求學生的總分，進而求出平均分。但因為只有讀入最後一個學生的分數以後才能求得平均分，且要打印出低於平均分的同學，故必須把 50 個學生的成績都保留下來，然後逐個和平均分比較，把高於平均分的成績打印出來。如果，用簡單變量 a_1, a_2, \dots, a_{50} 存放這些數據，可想而知程序要很長且繁。

要想如數學中使用下標變量 a_i 形式表示這 50 個數，則可以引入下標變量 $a[i]$ 。這樣問題的程序可寫為：

```
tot:=0; {tot 表示總分}
for i:=1 to 50 do {循環讀入每一個學生的成績，並累加它到總分}
begin
  read(a[i]);
  tot:=tot+a[i];
end;
ave:=tot/50; {計算平均分}
for i:=1 to 50 do
  if a[i]<ave then writeln('No.',i,' ',a[i]); {如果第 i 個同學成績小於平均分，則將輸出}
```

而要在程序中使用下標變量，則必須先說明這些下標變量的整體—數組，即數組是若干個同名（如上面的下標變量的名字都為 a ）下標變量的集合。

二、一維數組

當數組中每個元素只帶有一個下標時，我們稱這樣的數組為一維數組。

1. 一維數組的定義

(1) 類型定義

要使用數組類型等構造類型以及第 6 章要學習的自定義類型（枚舉類型與子界類型），應在說明部分進行類型說明。這樣定義的數據類型適用整個程序。

類型定義一般格式為：

```
type
  <標識符 1>=<類型 1>;
  <標識符 2>=<類型 2>;
  :
  :
  <標識符 n>=<類型 n>;
```

其中 type 是 Pascal 保留字，表示開始一個類型定義段。在其後可以定義若干個數據類型定義。<標識符>是為定義的類型取的名字，稱它為類型標識符。

類型定義後，也就確定了該類型數據取值的範圍，以及數據所能執行的運算。

(2) 一維數組類型的定義

一維數組類型的一般格式：

```
array [下標 1..下標 2] of <基類型>;
```

說明：其中 array 和 of 是 pascal 保留字。下標 1 和下標 2 是同一順序類型，且下標 2 的序號大於下標 1 的序號。它給出了數組中每個元素（下標變量）允許使用的下標類型，也決定了數組中元素的個數。基類型是指數組元素的類型，它可以是任何類型，同一個數組中的元素具有相同類型。因此我們可以說，數組是由固定數量的相同類型的元素組成的。

再次提請注意：類型和變量是兩個不同概念，不能混淆。就數組而言，程序的執行部分使用的不是數組類型（標識符）而是數組變量（標識符）。

一般在定義數組類型標識符後定義相應的數組變量，如：


```
type arraytype=array[1..8]of integer;
var a1,a2:arraytype;
```

其中 arraytype 為一個類型標識符，表示一個下標值可以是 1 到 8，數組元素類型為整型的一維數組；而 a1，a2 則是這種類型的數組變量。

也可以將其全並起來：

```
var a1,a2:array[1..8]of integer;
```

當在說明部分定義了一個數組變量之後，pascal 編譯程序為所定義的數組在內存空間開闢一串連續的存儲單元。

例如，設程序中有如下說明：

```
type rowtype=array[1..8]of integer;
   coltype=array['a'..'e']of integer;
var a:rowtype;b:coltype;
```

2．一維數組的引用

當定義了一個數組，則數組中的各個元素就共用一個數組名（即該數組變量名），它們之間是通過下標不同以示區別的。對數組的操作歸根到底就是對數組元素的操作。一維數組元素的引用格式為：

數組名[下標表達式]

說明：

(1) 下標表達式值的類型，必須與數組類型定義中下標類型完全一致，並且不允許超越所定義的下標下界和上界。

(2) 數組是一個整體，數組名是一個整體的標識，要對數組進行操作，必須對其元素操作。數組元素可以象同類型的普通變量那樣作用。如：a[3]:=34；是對數組 a 中第三個下標變量賦以 34 的值。read(a[4])；是從鍵盤讀入一個數到數組 a 第 4 個元素中去。


特殊地，如果兩個數組類型一致，它們之間可以整個數組元素進行傳送。如：

```
var a,b,c:array[1..100] of integer;
begin
   c:=a;a:=b;b:=c;
end.
```

在上程序中，a，b，c 三個數組類型完全一致，它們之間可以實現整數組傳送，例子中，先將 a 數組所有元素的值依次傳送給數組 c，同樣 b 數組傳給 a，數組 c 又傳送給 b，上程序段實際上實現了 a，b 兩個數組所有元素的交換。


對於一維數組的輸入與輸出，都只能對其中的元素逐個的輸入與輸出。在下面的應用示例中將詳細介紹。

三、一維數組應用示例

 **例 8.2**：輸入 50 個數，要求程序按輸入時的逆序把這 50 個數打印出來。也就是說，請你按輸入相反順序打印這 50 個數。
分析：我們可定義一個數組 a 用以存放輸入的 50 個數，然後將數組 a 內容逆序輸出。

源程序如下：

```
program ex8_2;
type arr=array[1..50]of integer; {說明一數組類型 arr}
var a:arr;i:integer;
begin
   writeln('Enter 50 integer:');
   for i:=1 to 50 do read(a[i]); {從鍵盤上輸入 50 個整數}
   readln;
   for i:=50 downto 1 do write(a[i]:10); {逆序輸出這 50 個數}
end.
```

 **例 8.3**：輸入十個正整數，把這十個數按由大到小的順序排列。

將數據按一定順序排列稱為排序，排序的算法有很多，其中選擇排序是一種較簡單的方法。

分析：要把十個數按從大到小順序排列，則排完後，第一個數最大，第二個數次大，……。因此，我們第一步可將第一個數與其後的各個數依次比較，若發現，比它大的，則與之交換，比較結束後，則第一個數已是最大的數（最大的泡往上冒）。同理，第二步，將第二個數與其後各個數再依次比較，又可得次大的數。如此方法進行比較，最後一次，將第九個數與第十個數比較，以決定次小的數。於是十個數的順序排列結束。

例如下面對 5 個進行排序，這個五個數分別為 8 2 9 10 5。按選擇排序方法，過程如下：

PASCAL 教程

```
初始數據   : 8  2  9 10  5
第一次排序 : 8  2  9 10  5
              9  2  8 10  5
              10 2  8  9  5
              10 2  8  9  5
第二次排序 : 10  8  2  9  5
              10  9  2  8  5
              10  9  2  8  5
第三次排序 : 10  9  8  2  5
              10  9  8  2  5
第四次排序 : 10  9  8  5  2
```

對於十個數，則排序要進行 9 次。源程序如下：

```
program ex8_3;
var a:array[1..10]of integer;
    i,j,t:integer;
begin
  writeln('Input 10 integers:');
  for i:=1 to 10 do read(a[i]); {讀入 10 個初始數據}
  readln;
  for i:=1 to 9 do {進行 9 次排序}
  begin
    for j:=i+1 to 10 do {將第 i 個數與其後所有數比較}
      if a[i]<a[j] then {若有比 a[i]大,則與之交換}
      begin
        t:=a[i];a[i]:=a[j];a[j]:=t;
      end;
    write(a[i]:5);
  end;
end.
```

練習八

1、輸入一串小寫字母（以“.”為結束標誌），統計出每個字母在該字符串中出現的次數（若某字母不出現，則不要輸出）。

```
例：
輸入：aaaabbbbccc.
輸出：a:4
      b:3
      c:3
```

2、輸入一個不大於 32767 的正整數 N，將它轉換成一個二進制數。

```
例如：
輸入：100
輸出：1100100
```

3、輸入一個由 10 個整數組成的序列，其中序列中任意連續三個整數都互不相同，求該序列中所有遞增或遞減子序列的個數。

```
例如：
輸入：1 10 8 5 9 3 2 6 7 4
輸出：6
對應的遞增或遞減子序列為：
1 10
10 8 5
5 9
9 3 2
2 6 7
7 4
```

第九課 多維數組

一、多維數組的定義

當一維數組元素的類型也是一維數組時，便構成了二維數組。二維數組定義的一般格式：

```
array [下標類型 1] of array [下標類型 2] of 元素類型;
```

但我們一般這樣定義二維數組：

```
array [下標類型 1, 下標類型 2] of 元素類型;
```

說明：其中兩個下標類型與一維數組定義一樣，可以看成“下界 1..上界 1”和“下界 2..上界 2”，給出二維數組中每個元素（雙下標變量）可以使用下標值的範圍。of 後面的元素類型就是基類型。

一般地，n 維數組的格式為：

```
array [下標類型 1, 下標類型 2, ..., 下標類型 n] of 元素類型;
```

其中，下標類型的個數即數組的維數，且說明了每個下標的類型及取值範圍。

二、多維數組元素的引用

多維數組的數組元素引用與一維數組元素引用類似，區別在於多維數組元素的引用必須給出多個下標。

引用的格式為：

```
<數組名>[下標 1, 下標 2, ..., 下標 n]
```

說明：顯然，每個下標表達式的類型應與對應的下標類型一致，且取值不超出下標類型所指定的範圍。

例如，設有說明：

```
type matrix=array[1..5,1..4]of integer;  
var a:matrix;
```

則表示 a 是二維數組，共有 $5 \times 4 = 20$ 個元素，它們是：

```
a [1,1] a [1,2] a [1,3] a [1,4]  
a [2,1] a [2,2] a [2,3] a [2,4]  
a [3,1] a [3,2] a [3,3] a [3,4]  
a [4,1] a [4,2] a [4,3] a [4,4]  
a [5,1] a [5,2] a [5,3] a [5,4]
```

因此可以看成一個矩陣，a [4,2] 即表示第 4 行、第 2 列的元素。由於計算機的存儲器是一維的，要把二維數組的元素存放到存儲器中，pascal 是按行（第一個下標）的次序存放，即按 a [1,1] a [1,2] a [1,3] a [1,4] a [2,1] ... a [5,4] 的次序存放於存儲器中某一組連續的存儲單元之內。

對於整個二維數組的元素引用時，大多採用二重循環來實現。如：給如上說明的二維數組 a 進行賦值：a[i,j]=i*j。

```
for i:=1 to 5 do  
  for j:=1 to 4 do  
    a[i,j]:=i*j;
```

對二維數組的輸入與輸出也同樣可用二重循環來實現：

```
for i:=1 to 5 do  
begin  
  for j:=1 to 4 do  
    read(a[i,j]);  
  readln;  
end;  
for i:=1 to 5 do  
begin  
  for j:=1 to 4 do  
    write(a[i,j]:5);
```

```
writeln;  
end;
```

三、多維數組的應用示例

📖 例 9.1 : 設有一程序:

```
program ex9_1;  
const n=3;  
type matrix=array[1..n,1..n]of integer;  
var a:matrix;  
    i,j:1..n;  
begin  
    for i:=1 to n do  
        begin  
            for j:=1 to n do  
                read(a[i,j]);  
            readln;  
        end;  
    for i:=1 to n do  
        begin  
            for j:=1 to n do  
                write(a[j,i]:5);  
            writeln;  
        end;  
    end.  
end.
```

且運行程序時的輸入為:

```
2 1 3 ←  
3 3 1 ←  
1 2 1 ←
```

則程序的輸出應是:

```
2 3 1  
1 3 2  
3 1 1
```

📖 例 9.2 : 輸入 4 名學生數學、物理、英語、化學、pascal 五門課的考試成績，求出每名學生的平均分，打印出表格。

分析：用二維數組 a 存放所給數據，第一下標表示學生的學號，第二個下標表示該學生某科成績，如 a[i,1]、a[i,2]、a[i,3]、a[i,4]、a[i,5] 分別存放第 i 號學生數學、物理、英語、化學、pascal 五門課的考試成績，由於要求每個學生的總分和平均分，所以第二下標可多開兩列，分別存放每個學生 5 門成績和總分、平均分。

源程序如下：


```
program ex9_2;  
var a:array[1..4,1..7]of real;  
    i,j:integer;  
begin  
    fillchar(a,sizeof(a),0);  
    {函數 fillchar 用以將 a 中所有元素置為 0}  
    writeln('Enter 4 students score');  
    for i:=1 to 4 do  
        begin  
            for j:=1 to 5 do {讀入每個人 5 科成績}  
                begin  
                    read(a[i,j]);  
                    {讀每科成績時同時統計總分}  
                    a[i,6]:=a[i,6]+a[i,j];  
                end;  
            readln;  
            a[i,7]:=a[i,6]/5; {求平均分}  
        end;  
    {輸出成績表}  
    writeln('No. Mat. Phy. Eng. Che. Pas. Tot. Ave.');
```

```

        write(a[i,j]:9:2);
    writeln;
end;
end.

```

四、數組類型的應用


 **例 9.3**：輸入一串字符，字符個數不超過 100，且以“.”結束。判斷它們是否構成回文。

分析：所謂回文指從左到右和從右到左讀一串字符的值是一樣的，如 12321，ABCBA，AA 等。先讀入要判斷的一串字符（放入數組 letter 中），並記住這串字符的長度，然後首尾字符比較，並不斷向中間靠攏，就可以判斷出是否為回文。

源程序如下：

```

program ex9_3;
var
    letter:array[1..100]of char;
    i,j:0..100;
    ch: char;
begin
    {讀入一個字符串以'.'號結束}
    write('Input a string:');
    i:=0; read(ch);
    while ch<>'.' Do
    begin
        i:=i+1; letter[i]:=ch;
        read(ch)
    end;
    {判斷它是否是回文}
    j:=1;
    while (j<i)and(letter[j]=letter[i])do
    begin
        i:=i-1;j:=j+1;
    end;
    if j>=i then writeln('Yes.')
    else writeln('No.');
```

 **例 9.4**：奇數階魔陣

魔陣（幻方）是用自然數 1，2，3...， n^2 填 n 階方陣的各個元素位置，使方陣的每行的元素之和、每列元素之和及主對角線元素之和均相等。奇數階魔陣的一個算法是將自然數數列第一個數 1 從方陣的中間一行最後一個位置排起，然後按以下規則排 2 ~ n^2 。

假設現在位置為 x, y ，下一個位置為 i (行號), j (列號)

- (1) 對 $A_{n,n}$ ($x=n, y=n$) 的下一個總是 $A_{n,n-1}$ ($i=n, j=n-1$)，
其它位置總是向右下角排，即 $A_{x,y}$ 的下一個是 $A_{x+1,y+1}$ ($i=x+1, j=y+1$)；
- (2) 列排完（即 $j=n+1$ 時），則轉排第一列（即 $j←1$ ）；
- (3) 行排完（即 $i=n+1$ 時），則轉排第一行（即 $i←1$ ）；
- (4) 若 $A_{i,j}$ 已排進一個自然數（即 $A_{i,j}>0$ ），則排 $A_{i-1,j-2}$ ($i←i-1, j←j-2$)。

例如 3 階方陣，則按上述算法可排成：

```

4 3 8
9 5 1
2 7 6

```

有了以上的算法，解題主要思路可用偽代碼描述如下：

```

1  i0←n div 2+1, j0←n /*排數的初始位置 a[i0,j0]*
2  a[i0,j0]←1;
3  for k:=2 to n*n do
4      計算下一個排數 k 位置 (i, j);
5      if a[i, j]>0 then
6          i←i-1;
7          j←j-2;
8      endif
9      a[i, j] ← k;

```

9 endfor

對於計算下一個排數位置，按上述的四種情形進行，但我們應先處理第三處情況。算法描述如下：

```
4.1  if (i=n) and (j=n) then
4.2      j←n-1; /*下一個位置為(n,n-1)*/;
4.3  else
4.4      i←i mod n +1; /*當 i<n 時，i←i+1；當 i=n 時，i←1*/
4.5      j←j mod n +1;
4.6  endif;
```

源程序如下：

```
program ex9_4;
var
  a : array[1..99,1..99] of integer;
  i,j,k,n : integer;
begin
  fillchar(a,sizeof(a),0);
  write('n=');readln(n);
  i:=n div 2+1;j:=n;
  a[i,j]:=1;
  for k:=2 to n*n do
  begin
    if (i=n)and(j=n) then
      j:=j-1
    else
      begin
        i:=i mod n +1;
        j:=j mod n +1;
      end;
    if a[i,j]<>0 then
      begin
        i:=i-1;
        j:=j-2;
      end;
    a[i,j]:=k;
  end;
  for i:=1 to n do
  begin
    for j:=1 to n do
      write(a[i,j]:5);
      writeln;
    end;
  end.
end.
```

練習九

1、輸入 N 個同學的語、數、英三科成績，計算他們的總分與平均分，並統計出每個同學的名次，最後以表格的形式輸出。

2、輸出楊輝三角的前 N 行 ($N<10$)。

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
```

第十課 字符與字符串處理

一、字符、字符串類型的使用

(一) 字符類型

字符類型為由一個字符組成的字符常量或字符變量。

字符常量定義：

Const

字符常量='字符';

字符變量定義：

Var

字符變量:char;

字符類型是一個有序類型，字符的大小順序按其 ASCII 代碼的大小而定。函數 succ、pred、ord 適用於字符類型。

例如：

後繼函數：succ('a')='b'
前繼函數：pred('B')='A'
序號函數：ord('A')=65

 **例 10.1**：按字母表順序和逆序每隔一個字母打印。即打印出：

a c e g I k m o q s u w y
z x r v t p n l j h f d b

程序如下：

```
program ex10_1;
  var letter:char;
begin
  for letter:='a' to 'z' do
    if (ord(letter)-ord('a'))mod 2=0 then write(letter:3);
  writeln;
  for letter:='z' downto 'a' do
    if (ord(letter)-ord('z'))mod 2 =0 then write(letter:3);
  writeln;
end.
```

分析：程序中，我們利用了字符類型是順序類型這一特性，直接將字符類型變量作為循環變量，使程序處理起來比較直觀。

(二) 字符串類型

字符串是由字符組成的有窮序列。

字符串類型定義：

type <字符串類型標識符>=string[n];

var

字符串變量：字符串類型標識符;

其中:n 是定義的字符串長度，必須是 0~255 之間的自然整數，第 0 號單元中存放串的實際長度，程序運行時由系統自動提供，第 1~n 號單元中存放串的字符。若將 string[n]寫成 string，則默認 n 值為 255。

例如：

```
type
  man=string[8];
  line=string;
var
```



```
name : man ;
screenline : line ;
```

另一種字符類型的定義方式為把類型說明的變量定義合併在一起。

例如：

```
VAR
  name : STRING[8] ;
  screenline : STRING ;
```

Turbo Pascal 中，一個字符串中的字符可以通過其對應的下標靈活使用。

例如：

```
var
  name : string ;
begin
  readln (name) ;
  for i := 1 to ord (name[0]) do
    writeln (name[i]) ;
  end.
```

語句 `writeln (name[i])` 輸出 `name` 串中第 `i` 個字符。

例 10.2 : 求輸入英文句子單詞的平均長度.

程序如下：

```
program ex10_2;
var
  ch:string;
  s,count,j:integer;
begin
  write('The sentence is :');
  readln(ch);
  s:=0;
  count:=0;
  j:=0;
  repeat
    inc(j);
    if not (ch[j] in [':',' ',';',',','!', '?', '.', ' ']) then inc(s);
    {也可用if (ch[j]>='a') and (ch[j]<='z') or (ch[j]>='A') and (ch[j]<='Z') then inc(s);}
    if ch[j] in ['.', '!', '?'] then inc(count);
  until (j=ord(ch[0])) or (ch[j] in ['.', '!', '?']);
  if not (ch[j] in ['.', '!', '?']) then writeln('It is not a sentence.')
  else writeln('Average length is ', s/count:10:4);
end.
```

分析:程序中，變量 `s` 用於存句子中英文字母的總數，變量 `count` 用於存放句子中單詞的個數，`ch[j]` 表示 `ch` 串中的第 `j` 個位置上的字符，`ord (ch[0])` 為 `ch` 串的串長度。程序充分利用 Turbo Pascal 允許直接通過字符串下標得到串中的字符這一特點，使程序比較簡捷。

二、字符串的操作

(一) 字符串的運算和比較

由字符串的常量、變量和運算符組成的表達式稱為字符串表達式。

字符串運算符包括：

1. 連接運算符 +

例如：'Turbo '+'PASCAL'的結果是'Turbo PASCAL'。

若連接的結果字符串長度超過 255，則被截成 255 個字符。若連接後的字符串存放在定義的字符串變量中，當其長度超過定義的字符串長度時，超過部份字符串被截斷。

例如：

```
var
  str1, str2, str3 : string[8] ;
```

```
begin
  str1:='Turbo ';
  str2:='PASCAL';
  str3:=str1+str2;
end.
```

則 str3 的值為：'Turbo PA'。

2. 關係運算符 =、<、>、<=、>=

兩個字符串的比較規則為，從左到右按照 ASC II 碼值逐個比較，遇到 ASC II 碼不等時，規定 ASC II 碼值大的字符所在的字符串為大。

例如：

```
'AB'<'AC' 結果為真；
'12'<'2' 結果為真；
'PASCAL'='PASCAL' 結果為假；
```

例 10.3：對給定的 10 個國家名，按其字母的順序輸出。

程序如下：

```
program ex10_3;
var i,j,k:integer;
    t:string[20];
    cname:array[1..10] of string[20];
begin
  for i:=1 to 10 do readln(cname[i]);
  for i:=1 to 9 do
  begin
    k:=i;
    for j:=i+1 to 10 do
      if cname[k]>cname[j] then k:=j;
    t:=cname[i];cname[i]:=cname[k];cname[k]:=t;
  end;
  for i:=1 to 10 do writeln(cname[i]);
end.
```

分析：程序中，當執行到 if cname[k]>cname[j]時，自動將 cname[k] 串與 cname[j] 串中的每一個字符逐個比較，直至遇到不等而決定其大小。這種比較方式是計算機中字符串比較的一般方式。

(二) 字符串的函數和過程

Turbo Pascal 提供了八個標準函數和標準過程，見下表，利用這些標準函數與標準過程，一些涉及到字符串的問題可以靈活解決。

函數和過程名	函數/過程	功 能	說 明
copy(s,m,n)	函數	取 s 中第 m 個字符開始的 n 個字符	若 m 大於 s 的長度，則返回空串；否則，若 m+n 大於 s 的長度，則截斷
length(s)	函數	求 s 的動態的長度	返回值為整數
pos(sub,s)	函數	在 s 中找子串 sub	返回值為 sub 在 s 中的位置，為 byte 型
insert(sour,s,m)	過程	在 s 的第 m 個字符位置處插入子串 sour	若返回串超過 255，則截斷
delete(s,m,n)	過程	刪除 s 中第 m 個字符開始的 n 個字符串	若 m 大於 s 的長度，則不刪除；否則，若 m+n 大於 s 的長度，則刪除到結尾
str(x[:w[:d]],s)	過程	將整數或實數 x 轉換成字符串 s	w 和 d 是整型表達式，意義同帶字寬的 write 語句
val(s,x,code)	過程	將字符串 S 轉換成整數或實數 x	若 S 中有非法字符，則 code 存放非法字符在 S 中的下標；否則，code 為零。code 為整型
upcase(ch)	函數	將字母 ch 轉換成大寫字母	若 ch 不為小寫字母，則不轉換

例 10.4：校對輸入日期(以標準英語日期，月/日/年)的正確性，若輸入正確則以年·月·日的方式輸出。

程序如下：

```
program ex10_4;
const
  max:array[1..12] of byte=(31,29,31,30,31,30,31,31,30,31,30,31); {每月的最多天數}
Var
  st:string;
  p,w,y,m,d:integer;
```


```
procedure err;
begin
  write('Input Error!');
  readln;
  halt;
end;

procedure init(var x:integer);
begin
  p:=pos('/',st);
  if (p=0) or (p=1) or (p>3) then err;
  val(copy(st,1,p-1),x,w);
  if w<>0 then err;
  delete(st,1,p);
end;

begin {of main}
  write('The Date is :');
  readln(st);
  init(m);
  init(d);
  val(st,y,w);
  if not (length(st)<>4) or (w<>0) or (m>12) or (d>max[m]) then err;
  if (m=2) and (d=29) then
    if y mod 100=0 then
      begin
        if y mod 400<>0 then err;
      end
    else if y mod 4<>0 then err;
  write('Date : ',y,'.',m,'.',d);
  readln;
end.
```

分析：此題的題意很簡單，但在程序處理時還需考慮以下幾方面的問題。

- (1) 判定輸入的月和日應是 1 位或 2 位的數字，程序中用了一個過程 `inst`，利用串函數 `pos`，求得分隔符/所在的位置而判定輸入的月和日是否為 1 位或 2 位，利用標準過程 `val` 判定輸入的月和日是否為數字；
- (2) 判定月和日是否規定的日期範圍及輸入的年是否正確；
- (3) 若輸入的月是 2 月份，則還需考慮閏年的情況。

 **例 10.5**：對輸入的一句子實現查找且置換的功能。

程序如下：

```
program ex10_5;
var
  s1,s,o:string;
  i:integer;
begin
  write('The text:');
  readln(s1);
  write('Find:');readln(s);
  write('Replace:');readln(o);
  i:=pos(s,s1);
  while i<>0 do
  begin
    delete(s1,i,length(s));
    insert(o,s1,i);
    i:=pos(s,s1);
  end;
  writeln(s1);
  readln;
end.
```

分析：程序中，輸入要查找的字符串及要置換的字符串，充分用上了字符串處理的標準過程 `delete`、`insert` 及標準函數 `pos`。

練習十

第十一課 枚舉、子界、集合及記錄類型

在前面幾章中我們用到了整型、實型、布爾型、字符型的數據。以上數據類型是由 pascal 規定的標準數據類型，只要寫 integer，real，boolean，char，pascal 編譯系統就能識別並按這些類型來處理。pascal 還允許用戶定義一些類型，這是其它一些語言所沒有的，這就使得 pascal 使用靈活、功能豐富。

一、枚舉類型

隨著計算機的不斷普及，程序不僅只用於數值計算，還更廣泛地用於處理非數值的數據。例如，性別、月份、星期幾、顏色、單位名、學歷、職業等，都不是數值數據。

在其它程序設計語言中，一般用一個數值來代表某一狀態，這種處理方法不直觀，易讀性差。如果能在程序中用自然語言中有相應含義的單詞來代表某一狀態，則程序就很容易閱讀和理解。也就是說，事先考慮到某一變量可能取的值，儘量用自然語言中含義清楚的單詞來表示它的每一個值，這種方法稱為枚舉方法，用這種方法定義的類型稱枚舉類型。

枚舉類型是一種很有實用價值的數據類型，它是 pascal 一項重要創新。

(一) 枚舉類型的定義

枚舉類型是一種自定義類型，要使用枚舉類型當然也要先說明枚舉類型。

枚舉類型的一般格式：

(標識符 1，標識符 2，...，標識符 n)

說明：

(1) 括號中的每一個標識符都稱為枚舉元素或枚舉常量。

(2) 定義枚舉類型時列出的所有枚舉元素構成了這種枚舉類型的值域（取值範圍），也就是說，該類型的變量所有可能的取值都列出了。

例如，下列類型定義是合法的：

```
type
  days=(sun,mon,tue,wed,thu,fri,sat);
  colors=(red,yellow,blue,white,black,green);
```

而下列類型定義是錯誤的(因為枚舉元素非標識符)：

```
type
  colortype=('red','yellow','blue','white'); { '不可用作標識符' }
  numbers=(1,3,5,7,9); { 標識符不可以數字開頭 }
  ty=(for,do,while); { 保留字不可用作標識符 }
```

(二) 枚舉類型變量

定義了枚舉類型，就可以把某些變量說明成該類型。如：

```
var
  holiday,workday:days;
  incolor:colors;
```

也可以把變量的說明與類型的定義合併在一起，如：

```
var
  holiday,workday:(sun,mon,tue,wed,thu,fri,sat);
  incolor:(red,yellow,blue,white,black,green);
```

(三) 枚舉類型的性質

1. 枚舉類型屬於順序類型

根據定義類型時各枚舉元素的排列順序確定它們的序號，第一個枚舉元素的序號為 0。例如：設有定義：

```
type days=(sun,mon,tue,wed,thu,fri,sat);
```

則：

```
ord(sun)=0,ord(mon)=1,ord(sat)=6;succ(sun)=mon,succ(mon)=tue,
succ(fri)=sat;pred(mon)=sun,pred(tue)=mon,pred(sat)=fri。
```

應注意的是：枚舉類型中的第一個元素無前趨，最後一個元素無後繼。

2．對枚舉類型只能進行賦值運算和關係運算

一旦定義了枚舉類型及這種類型的變量，則在語句部分只能對枚舉類型變量賦值，或進行關係運算，不能進行算術運算和邏輯運算。

在枚舉元素比較時，實際上是對其序號的比較。當然，賦值或比較時，應注意類型一致。

例如，設程序有如下說明：

```
type
  days=(sun,mon,tue,wed,thu,fri,sat);
  colors=(red,yellow,blue,white,black,green);
var
  color:colors;
  weekday:days;
```

則下列比較或語句是合法的：

```
weekday:=mon;
if weekday=sun then write('rest');
```

而下面比較或語句是不合法的：

```
mon:=weekday;
mon:=1;
if weekday=sun or sat then write('rest');
if sun>red then weekday:=red;
if weekday<>color then weekday:=color;
```

3．枚舉變量的值只能用賦值語句來獲得

也就是說，不能用 read (或 readln) 讀一個枚舉型的值。同樣，也不能用 write (或 writeln) 輸出一個枚舉型的值。如 write(red) 是非法的，會發生編譯錯誤。千萬不要誤認為，該語句的結果是輸出 'red' 三個字符。

但對枚舉數據的輸入與輸出可通過間接方式進行。輸入時，一般可輸入一個代碼，通過程序進行轉換，輸出時，也只是打印出與枚舉元素相對應的字符串。這在後面的例題中將有使用示例。


4．同一個枚舉元素不能出現在兩個或兩個以上的枚舉類型定義中。

如：

```
type
  color1=(red,yellow,white);
  color2=(blue,red,black);
```

是不允許的，因為 red 屬於兩個枚舉類型。

(四) 枚舉類型應用舉例

 **例 11.1**：一周七天用 sun, mon, tue, wed, thu, fri, sat 表示，要求利用枚舉類型編程：當輸入星期幾的數字，能輸出它的後一天是星期幾 (也用英文表示)。

源程序如下：

```
program ex11_1;
type week=(sun,mon,tue,wed,thu,fri,sat);
var
  i : integer;
  day,sucday : week;
begin
  write('What date is it');readln(i);
  case i of {根據輸入 i 轉換成枚舉型}
    1:day:=mon;
    2:day:=tue;
    3:day:=wed;
    4:day:=thu;
    5:day:=fri;
    6:day:=sat;
    7:day:=sun;
```

```
end;
{計算明天 sucday}
if (day=sat) then sucday:=sun
else sucday:=succ(day);
{輸出明天是星期幾}
write('The next day is ');
case sucday of
    sun:writeln('sunday');
    mon:writeln('monday');
    tue:writeln('tuesday');
    wed:writeln('wednesday');
    thu:writeln('thursday');
    fri:writeln('friday');
    sat:writeln('saturday');
end;
end.
```

評注：程序中變量 day、sucday 分別表示今天、明天。

二、子界類型

如果我們定義一個變量 i 為 integer 類型，那麼 i 的值在微型機系統的 pascal 中，使用 2 字節的定義表示法，取值範圍為-32768～32767。而事實上，每個程序中所用的變量的值都有一個確定的範圍。

例如，人的年齡一般不超過 150，一個班級的學生不超過 100 人，一年中的月數不超過 12，一月中的天數不超過 31，等等。

如果我們能在程序中對所用的變量的值域作具體規定的話，就便於檢查出那些不合法的數據，這就能更好地保證程序運行的正確性。而且在一定程度上還會節省內存空間。

子界類型就很好解決如上問題。此外，在數組的定義中，常用到子界類型，以規定數組下標的範圍，上一章有關數組知識中我們已用到。

(一) 子界類型定義

子界類型的一般格式：

<常量 1>..<常量 2>

說明：

- (1) 其中常量 1 稱為子界的下界，常量 2 稱為子界的上界。
- (2) 下界和上界必須是同一順序類型（該類型稱為子界類型的基類型），且上界的序號必須大於下界的序號。例如，下列說明：

```
type
    age=0.5..150;  {實數不是順序類型}
    letter=0..'z'; {不是同一類型}
    let1='z'..'a'; {上界序號必須大於下界序號}
```

都是錯誤的。

- (3) 可以直接在變量說明中定義子界類型。如：

```
type
    letter='a'..'d';
var ch1,ch2:letter;
```

可以合併成：

```
var ch1,ch2:'a'..'d';
```

當然，將類型定義和變量定義分開，則更為清晰。

(二) 子界類型數據的運算規則

1. 凡可使用基類型的運算規則同樣適用該類型的子界類型。

例如，可以使用整型變量的地方，也可以使用以整型為基類型的子界類型數據。

2. 對基類型的運算規則同樣適用於該類型的子界類型。

例如，div，mod 要求參加運算的數據為整，因而也可以為整型的任何子界類型數據。

3 · 基類型相同的不同子界類型數據可以進行混合運算。

例如：設有如下說明：

```
Type
  a=1..100;
  b=1..1000;
  c=1..500;
var
  x:a;
  y:b;
  t:c;
  z:integer;
```

則下列語句也是合法的：

```
Z:=Sqr(x)+y+t;
```

下列語句：

```
t:=x+y+z;
```

當 $x+y+z$ 的值在 1~500 範圍內時是合法的，否則會出錯。

(三) 子界類型應用舉例

📖 例 11.2：利用子界類型作為情況語句標號，編一個對數字，大小寫字母和特殊字符進行判別的程序。

源程序如下：

```
program ex11_2;
var c:char;
begin
  readln(c);
  case c of
    '0'..'9':writeln('digits');
    'A'..'Z':writeln('UPPER-CASELETTERS');
    'a'..'z':writeln('lower-caseletters');
    esle writeln('special charactors');
  end;
end.
```

📖 例 11.3：使用子界型情況語句，當輸入月、日、年(10 30 1986)，輸出 30 Oct 1986。

源程序如下：

```
program ex11_3;
var
  month:1..12;
  date:1..31;
  year:1900..2999;
begin
  write('Enter date(mm dd yyyy):');
  readln(month,date,year);
  write(date);
  case month of
    1:write('Jan':5);
    2:write('Feb':5);
    3:write('Mar':5);
    4:write('Apr':5);
    5:write('May':5);
    6:write('Jun':5);
    7:write('Jul':5);
    8:write('Aug':5);
    9:write('Sep':5);
    10:write('Oct':5);
    11:write('Nov':5);
    12:write('Dec':5);
  end;
  writeln(year:6);
end.
```

枚舉類型和子界類型均是順序類型，在前面一章數組的定義時，實際上我們已經用到了子界類型，數組中的下標類型確切地講可以是枚舉類型或子界類型，大多數情況下用子界類型。

如有如下說明：

```
type color=(red,yellow,blue,white,black);
var
  a:array[color]of integer;
  b:array[1..100]of color;
```

都是允許的。

三、集合類型

集合是由具有某些共同特徵的元素構成的一個整體。在 pascal 中，一個集合是由具有同一有序類型的一組數據元素所組成，這一有序類型稱為該集合的基類型。

(一) 集合類型的定義和變量的說明

集合類型的一般形式為：

set of <基類型>;

說明：

- (1) 基類型可以是任意順序類型，而不能是實型或其它構造類型。同時，基類型的數據的序號不得超過 255。

例如下列說明是合法的：

```
type
  letters=set of 'A'..'Z';
  numbers=set of 0..9;
  s1=set of char;
  ss=(sun,mon,tue,wed,thu,fri,sat);
  s2=set of ss;
```

- (2) 與其它自定義類型一樣，可以將類型說明與變量說明合併在一起。如：

```
type numbers=set of 0..9;
var s:numbers;
```

與下列語句等價。

```
var s:set of 0..9;
```

(二) 集合的值

集合的值是用 “[” 和 “]” 括起來，中間為用逗號隔開的若干個集合的元素。如：

```
[ ] {空集}
[1,2,3]
['a','e','i','o','u']
```

都是集合。

說明：

- (1) 集合的值放在一對方括號中，各元素之間用逗號隔開。
- (2) 在集合中可以沒有任何元素，這樣的集合稱為空集。
- (3) 在集合中，如果元素的值是連續的，則可用子界型的表示方法表示。例如：

```
[1,2,3,4,5,7,8,9,10,15]
```

可以表示成：

```
[1..5,7..10,15]
```

- (4) 集合的值與方括號內元素出現的次序無關。例如，[1,5,8] 和 [5,1,8] 的值相等。
- (5) 在集合中同一元素的重複出現對集合的值沒有影響。例如，[1,8,5,1,8] 與 [1,5,8] 的值相等。

(6) 每個元素可用基類型所允許的表達式來表示。如 $[1, 1+2, 4]$ 、 $[ch]$ 、 $[succ(ch)]$ 。

(三) 集合的運算

1. 賦值運算

只能通過賦值語句給集合變量賦值，不能通過讀語句 `read`(或 `readln`) 賦值，也不能通過 `write`(或 `writeln`) 語句直接輸出集合變量的值。

2. 集合的並、交、差運算

可以對集合進行並、交、差三種運算，每種運算都只能有一個運算符、兩個運算對象，所得結果仍為集合。三種運算符分別用 “+”、“*”、“-” 表示。注意它們與算術運算的區別。

3. 集合的關係運算

集合可以進行相等或不相等、包含或被包含的關係運算，還能測試一個元素是否在集合中。所用的運算符分別是 “=”、“<>”、“>=”、“<=” 及 “in”，它們都是二目運算，且運算符 “=”、“<>”、“>=” 及 “<=” 的運算對象都是相容的集合類型，運算符 `in` 的右邊為集合，左邊為與集合基類型相同的表達式。

例 11.4：設有如下說明：

```
type
  weekday=(sun,mon,tue,wed,thu,fri,sat);
  week=set of weekday;
  subnum=set of 1..50;
```

寫出下列表達式的值：

```
1 : {sun,sat} + {sun,tue,fri}
2 : {sun,fri} * {mon,tue}
3 : {sun,sat} * {sun..sat}
4 : {sun} - {mon,tue}
5 : {mon} - {mon,tue}
6 : {sun..sat} - {mon,sun,sat}
7 : {1,2,3,5} = {1,5,3,2}
8 : {1,2,3,4} <> {1..4}
9 : {1,2,3,5} >= {1..3}
10: {1..5} <= {1..4}
11: {1,2,3} <= {1..3}
12: 2 in {1..10}
```

答：表達式的值分別是：

```
1 : {sun,sat,tue,fri}
2 : {}
3 : {sun,sat}
4 : {}
5 : {}
6 : {tue..fri}
7 : TRUE
8 : FALSE
9 : TRUE
10: FALSE
11: TRUE
12: TRUE
```

例 11.5：輸入一系列字符，對其中的數字字符、字母字符和其它字符分別計數。輸入 '?' 後結束。

源程序如下：

```
program ex11_5;
var
  id,il,io:integer;
  ch:char;
  letter:set of char;
  digit:set of '0'..'9';
begin
  letter:=['a'..'z','A'..'Z'];
  digit:=['0'..'9'];
  id:=0;il:=0;io:=0;
  repeat
    read(ch);
```

```
    if ch in letter
    then il:=il+1
    else
        if ch in digit then id:=id+1
        else io:=io+1;
    until ch='?';
    writeln('letter:',il,'digit:',id,'Other:',io);
end.
```

四、記錄類型

在程序中對於組織和處理大批量的數據來說，數組是一種十分方便而又靈活的工具，但是數組在使用中有一個基本限制，這就是：一個數組中的所有元素都必須具有相同的類型。但在實際問題中可能會遇到另一類數據，它是由性質各不相同的成份組成的，即它的各個成份可能具有不同的類型。例如，有關一個學生的數據包含下列項目：

學號	字符串類型
姓名	字符串類型
年齡	整型
性別	字符型
成績	實型數組

Pascal 給我們提供了一種叫做記錄的結構類型。在一個記錄中，可以包含不同類型的並且互相相關的一些數據。

(一) 記錄類型的定義

在 pascal 中，記錄由一組稱為“域”的分量組成，每個域可以具有不同的類型。

記錄類型定義的一般形式：

Record

```
<域名 1>:<類型 1>;
<域名 2>:<類型 2>;
:
:
<域名 n>:<類型 n>;
end;
```

說明：

(1) 域名也稱域變量標識符，應符合標識符的語法規則。在同一個記錄中類型中，各個域不能取相同的名，但在兩個不同記錄類型中的域名可以相同。

(2) 記錄類型的定義和記錄變量可以合併為一個定義，如：

```
type
    date=record
        year:1900..1999;
        month:1..12;
        day:1..31
    end;
var x:date;
```

可以合併成：

```
var
    x: record
        year:1900..1999;
        month:1..12;
        day:1..31
    end;
```

(3) 對記錄的操作，除了可以進行整體賦值，也能對記錄的分量——域變量進行。

(4) 域變量的表示方法如下：

記錄變量名.域名

如前面定義的記錄 x，其 3 個分量分別為：x.year，x.month，x.day。

(5) 域變量的使用和一般的變量一樣，即域變量是屬於什麼數據類型，便可以進行那種數據類型所允許的操作。

(二) 記錄的嵌套

當一個記錄類型的某一個域類型也是記錄類型的時候，我們說發生了記錄的嵌套，看下面的例子：

📖 例 11.6：某人事登記表可用一個記錄表示，其中各項數據具有不同的類型，分別命名一個標識符。而其中的“出生年月日”又包括三項數據，還可以用一個嵌套在內層的記錄表示。

具體定義如下：

```
type
  sexs=(male,female);
  date=record
    year:1900..1999;
    month:1..12;
    day:1..31;
  end;
  personal=record
    name:string[15];
    sex:sexs;
    birthdate:date;
    home:string[40];
  end;
```

📖 例 11.7：設計一個函數比較兩個 `dates` 日期類型記錄變量的遲早。

設函數名、形參及函數類型定義為：

```
AearlyB(A,B:dates):boolean;
```

函數的形參為兩個 `dates` 類型的值參數。當函數值為 `true` 時表示日期 A 早於日期 B，否則日期 A 遲於日期 B 或等於日期 B。顯然不能對 A、B 兩個記錄變量直接進行比較，而要依具體的意義逐域處理。

源程序如下：

```
program ex11_7;
type
  dates=record
    year:1900..1999;
    month:1..12;
    day:1..31
  end;
var x,y:dates;

function AearlyB(A,B:dates):boolean;
var earln:boolean;
begin
  early:=false;
  if (A.year<B.year) then early:=true;
  if (A.year=B.year) and (A.month<B.month)
  then early:=true;
  if (A.year=B.year) and (A.month=B.month) and (A.day<B.day)
  then early:=true;
  AearlyB:=early;
end; {of AearlyB}

begin
  write('Input DATE X(mm-dd-yy):'); readln(X.month,X.day,X.year);
  write('Input DATE Y(mm-dd-yy):'); readln(Y.month,Y.day,Y.year);
  if AearlyB(X,Y) then writeln('Date X early!') else writeln('Date X not early!');
end.
```

(三) 開域語句

在程序中對記錄進行處理時，經常要引用同一記錄中不同的域，每次都按 11.4.1 所述的格式引用，非常乏味。為此 Pascal 提供了一個 `with` 語句，可以提供引用域的簡單形式。

開域語句一般形式：

```
with <記錄變量名表> do
<語句>;
```

功能：在 do 後的語句中使用 with 後的記錄的域時，只要直接寫出域名即可，即可以省略記錄變量名和“.”。

說明：

- (1) 一般在 with 後只使用一個記錄變量名。如：

```
write('Input year:');
readln(x.year);
write('Input month:');
readln(x.month);
write('Input day:');
readln(x.day);
```

可以改寫成：

```
with x do
begin
  write('Input year:');readln(year);
  write('Input month:');readln(month);
  write('Input day:');readln(day);
end;
```

- (2) 設 x, y 是相同類型的記錄變量，下列語句是非法的：

```
with x,y do...;
```

- (3) with 後接若干個記錄名時，應是嵌套的關係。如有記錄說明：


```
var
  x:record
    i:integer;
    y:record
      j:0..5;
      k:real;
    end;
    m:real
  end;
```

可以使用：

```
with x do
begin
  read(i);
  with y do
    read(j,k);
  readln(m);
end;
```

或簡寫為：

```
with x,y do
readln(i,j,k,m);
```

 **例 11.8**：讀入 10 個日期，再對每個日期輸出第二天的日期。輸入日期的格式是月、日、年，如 9□30□1993，輸出的格式為 10/1/1993。


分析：可用一個記錄變量 today 表示日期。知道一個日期後要更新為第二天的日期，應判斷輸入的日期是否為當月的最後一天，或當年的最後一天。

源程序如下：

```
program ex11_8;
type
  date=record
    month:1..12;
    day:1..31;
    year:1900..1999;
  end;
var
  today:array[1..10]of date;
  i:integer;
  maxdays:28..31;
```

```
begin
  for i:=1 to 10 do {輸入 10 個日期}
    with today[i] do
      readln(month,day,year);
  for i:=1 to 10 do
    with today[i] do {求第 i 個日期中月份最後一天 maxdays}
      begin
        case month of
          1,3,5,7,8,10,12:maxdays:=31;
          4,6,9,11 :maxdays:=30;
          2:if(year mod 400=0) or( year mod 4=0) and(year mod 100<>0) then
            maxdays:=29
            else maxdays:=28;
          end;
          if day=maxdays then
            begin
              day:=1;
              if month=12 then
                begin
                  month:=1;year:=year+1;
                end
                else month:=month+1;
              end
              else day:=day+1;
              writeln(month,'/',day,'/',year);
            end;
          end.
end.
```

五、應用實例

 **例 11.9**：編制用篩法求 $1-n$ ($n \leq 200$) 以內素數的程序。

分析：由希臘著名數學家埃拉托色尼提出的所謂“篩法”，步驟如下：

- (1) 將所有候選數放入篩中；
- (2) 找篩中最小數（必為素數）next，放入集合 primes 中；
- (3) 將 next 的所有倍數從篩中篩去；
- (4) 重複 (2) ~ (4) 直到篩空。

編程時，用集合變量 sieve 表示篩子，用集合 primes 存放所有素數。

源程序如下：

```
program ex11_9;
const n=200;
var
  sieve,primes:set of 2..n;
  next,j:integer;
begin
  sieve:=[2..n]; {將所有候選數放入篩中}
  primes:=[]; {素數集合置空}
  next:=2;
  repeat
    {找篩 sieve 中最小一個數}
    while not(next in sieve) and(next<=n)do
      next:=succ(next);
    primes:=primes+[next]; {將最小數放入素數集合中}
    {將這個素數的倍數從篩中刪去}
    j:=next;
    while j<=n do
      begin
        sieve:=sieve-[j];
        j:=j+next;
      end
    until sieve=[];
    j:=0;
  for next:=2 to n do {打印出所有素數}
    if next in primes then
      begin
```

```
    write(next:5);  
    j:=j+1;  
    if j mod 10=0 then writeln;  
    end;  
    writeln;  
end.
```

練習十一

1、一家水果店出售四種水果，每公斤的價格是：蘋果 1.50 元，桔子 1.80 元，香蕉 2.0，菠蘿 1.60 元。編一個程序，使售貨員只要從鍵盤輸入貨物的代碼及重量，計算機便能顯示貨物的名稱、單價、重量及總價。

2、輸入一個英語句子，以句號，為結束標誌，統計句子中元音字母出現的次數，把句子所有輔音字母組成一個集合，並把這些輔音字母打印出來。

3、編程序建立某班 25 人的數學課程成績表，要求用數組類型和記錄類型，其成績表格式如下：

姓名	性別	平時成績	期中考試	期終考試	總評成績
張良	男	90	85	92	?
王心	男	70	82	71	?
.....					
李英	女	82	84	75	?

其中總評成績=平時成績×20%+期中考試×30%+期終考試×50%。

4、輸入五個學生的出生日期(月/日/年)和當天的日期，然後用計算機計算出每個人到當天為止的年齡各是多少?(如某人 1975 年 10 月 1 日出生，今天是 94 年 12 月 1 日，則他的年齡應為 19 歲，而另一人的出生日期為 76 年 12 月 30 日，則他的年齡為 17 歲。)

第十二課 過程與函數

前面我們曾經學習了程序設計中的三種基本控制結構（順序、分支、循環）。用它們可以組成任何程序。但在應用中，還經常用到子程序結構。

通常，在程序設計中，我們會發現一些程序段在程序的不同地方反復出現，此時可以將這些程序段作為相對獨立的整體，用一個標識符給它起一個名字，凡是程序中出現該程序段的地方，只要簡單地寫上其標識符即可。這樣的程序段，我們稱之為子程序。

子程序的使用不僅縮短了程序，節省了內存空間及減少了程序的編譯時間，而且有利於結構化程序設計。因為一個複雜的問題總可將其分解成若干個子問題來解決，如果子問題依然很複雜，還可以將它繼續分解，直到每個子問題都是一個具有獨立任務的模塊。這樣編制的程序結構清晰，邏輯關係明確，無論是編寫、閱讀、調試還是修改，都會帶來極大的好處。

在一個程序中可以有主程序而沒有子程序（本章以前都是如此），但不能沒有主程序，也就是說不能單獨執行子程序。pascal 中子程序有兩種形式：函數和過程。

一、函數

在此之前，我們曾經介紹並使用了 pascal 提供的各種標準函數，如 ABS，SUCC 等等，這些函數為我們編寫程序提供了很大的方便。但這些函數只是常用的基本函數，編程時經常需要自定義一些函數。

（一）函數的說明

在 pascal 中，函數也遵循先說明後使用的規則，在程序中，函數的說明放在調用該函數的程序（主程序或其它子程序）的說明部分。函數的結構主程序的結構很相似。

函數定義的一般格式：

```
function <函數名> (<形式參數表>):<類型>; {函數首部}
<說明部分>
begin
    <語句>;
    :
    :
    <語句>;
end;
```

} 函數體

說明：

- (1) 函數由首部與函數體兩部分組成。
- (2) 函數首部以關鍵字 function 開頭。
- (3) 函數名是用戶自定義的標識符。
- (4) 函數的類型也就是函數值的類型，所求得的函數值通過函數名傳回調用它的程序。可見，函數的作用一般是為了求得一個值。
- (5) 形式參數簡稱形參，形參即函數的自變量。自變量的初值來源於函數調用。在函數中，形參一般格式如下：

變量名表 1：類型標識符 1；變量名表 2：類型標識符 2；...；變量名表 n：類型標識符 n

可見形參表相當於變量說明，對函數自變量進行說明，但應特別注意：此處只能使用類型標識符，而不能直接使用類型。

- (6) 當缺省形參表（當然要同時省去一對括號）時，稱為無參函數。
- (7) 函數體與程序體基本相似，由說明部分和執行部分組成。
- (8) 函數體中的說明部分用來對本函數使用的標號、常量、類型、變量、子程序加以說明，這些量只在本函數內有效。
- (9) 函數體的執行部分由 begin 開頭，end 結束，中間有若干用分號隔開的語句，只是 end 後應跟分號，不能像程序那樣用句號“.”。
- (10) 在函數體的執行部分，至少應該給函數名賦一次值，以使在函數執行結束後把函數值帶回調用程序。

(二) 函數的調用

我們可以在任何與函數值類型兼容的表達式中調用函數，或者說，函數調用只能出現在允許表達式出現的地方，或作為表達式的一個因子。

函數調用方式與標準函數的調用方式相同。

函數調用的一般格式：

<函數名>

或

<函數名> (實在參數表)

說明：

- (1) 實在參數簡稱實參。實參的個數必須與函數說明中形參的個數一致，實參的類型與形參的類型應當一一對應。
- (2) 調用函數時，一般的，實參必須有確定的值。
- (3) 函數調用的步驟為：計算實參的值，“賦給”對應的形參；

(三) 函數的應用舉例

📖 例 12.1：求正整數 A 和 B 之間的完全數 (A<B)。

分析：所謂完全數是指它的小於該數本身的因子之和等於它本身，如 $6=1+2+3$ ，6 即是一個完全數。因此我們可定義一個布爾型函數 perfect(x)，若 x 是完全數，其值為 TRUE，否則為 FALSE。整個程序算法如下：

```
1 for i:=A to B do
2 if perfect(i) then writeln(i);
```

源程序如下：

```
program ex12_1;
var
  i,a,b : integer;

function perfect(x:integer):boolean;
var
  k,sum : integer;
begin
  {累加 x 所有小於本身的因數}
  sum:=1;
  for k:=2 to x div 2 do
    if x mod k=0 then sum:=sum+k;
  {判斷 x 是否是完全數}
  perfect:=x=sum; {將結果賦值給函數名}
end;{end of perfect}

begin{主程序開始}
  write('Input a,b:');
  repeat {輸入 0<a<b}
    readln(a,b);
  until (a>0) and (b>0) and (a<b);
  writeln('List of all perfect numbers:');
  {從 a 到 b 逐個判斷,是完全數則打印出來}
  for i:=a to b do
    if perfect(i) then writeln(i);
end.
```

自定義函數只是主程序的說明部分，若主程序中沒有調用函數，則系統不會執行函數子程序。當主程序調用一次函數時，則將實在參數的值傳給函數的形式參數，控制轉向函數子程序去執行，子程序執行完畢後自動返回調用處。

二、過程

在 pascal 中，自定義過程與自定義函數一樣，都需要先定義後調用。函數一般用於求值，而過程一般實現某些操作。

(一) 過程的說明

過程說明的一般格式為：

```
procedure <過程名> (<形式參數表>); {過程首部}
<說明部分>
begin
    <語句>;
    ⋮
    <語句>;
end;
```

} 過程體

說明：

- (1) 過程首部以關鍵字 `procedure` 開頭。
- (2) 過程名是用戶自定義的標識符，只用來標識一個過程，不能代表任何數據，因此不能說明“過程的類型”。
- (3) 形參表缺省（當然要同時省去一對括號）時，稱為無參過程。
- (4) 形參表的一般格式形式如下：

[var] 變量名表：類型；...；[var] 變量名表：類型。

其中帶 `var` 的稱為變量形參，不帶 `var` 的稱為值形參。在函數中，形參一般都是值形參，很少用變量形參（但可以使用）。例如，下列形參表中：

```
(x,y:real;n:integer;var w:real;var k:integer;b:real)
```

`x、y、n、b` 為值形參，而 `w、k` 為變量形參。

調用過程時，通過值形參給過程提供原始數據，通過變量形參將值帶回調用程序。因此，可以說，值形參是過程的輸入參數，變量形參是過程的輸出參數。有關變參，這在後面內容具體敘述。

- (5) 過程體與程序、函數體類似。與函數體不同的是：函數體的執行部分至少有一個語句給函數名賦值，而過程體的執行部分不能給過程名賦值，因為過程名不能代表任何數據。
- (6) 過程體的說明部分可以定義只在本過程有效的標號、常量、類型、變量、子程序等。

(二) 過程的調用

過程調用是通過一條獨立的過程調用語句來實現的，它與函數調用完全不同。過程調用與調用標準過程（如 `write`，`read` 等）的方式相同。調用的一般格式為：

<過程名>;

或

<過程名> (實在參數表);

說明：

- (1) 實參的個數、類型必須與形參一一對應。
- (2) 對應於值形參的實參可以是表達式，對應於變量形參的實參只能是變量。
- (3) 過程調用的步驟為：計算實參的值；將值或變量的“地址”傳送給對應的形參；執行過程體；返回調用處。

過程與函數有下列主要區別：

- 過程的首部與函數的首部不同；
- 函數通常是為了求一個函數值，而過程可以得到若干個運算結果，也可用來完成一系列的數據處理，或用來完成與計算無關的各種操作；

- 調用方式不同。函數的調用出現在表達式中，而過程調用是一個獨立的語句。

(三) 過程的應用舉例

📖 例 12.2 : 輸出以下一個圖形:

```
*
**
***
****
*****
*****
```

分析：我們前面學習可用的二重循環打印出上圖形，現我們設置一個過程打印出 N 個連續的“*”號。
源程序如下：

```
program ex12_2;
var i:integer;

procedure draw_a_line(n:integer); {該過程打印出連續 n 個星號，並換行}
var j:integer;
begin
  for j:=1 to n do
    write('*');
  writeln;
end;

begin
  for i:=1 to 6 do
    draw_a_line(i); {調用過程，第 i 行打印 i 個連續星號}
  end.
```

三、過程、函數的數據傳遞

在程序調用子程序時，調用程序將數據傳遞給被調用的過程或函數，而當子程序運行結束後，結果又可以通過函數名、變參。當然也可以用全局變量等形式實現數據的傳遞。這一節我們，就來研究參數傳遞與局部變量、全局變量等問題。

(一) 數值參數和變量參數

前面已經講過，pascal 子程序中形式參數有數值形參（簡稱值參）和變量形參（變參）兩種。事實上，還有函數形參和過程形參兩種，只是應用並不太多，我們不作深入地研究。

1. 值形參

值參的一般格式如一(一)(5)所示。應該強調的是：

- (1) 形參表中只能使用類型標識符，而不能使用類型。
- (2) 值形參和對應的實參必須一一對應，包括個數和類型。
- (3) 實參和值形參之間數據傳遞是單向的，只能由實參傳送給形參，相當賦值運算。
- (4) 一個特殊情況是，當值形參是實型變量名時，對應的實參可以是整型表達式。
- (5) 值形參作為子程序的局部量，當控制返回程序後，值形參的存儲單元釋放。

2. 變量形參

變量形參的一般格式如二(一)(4)所示，必須在形參前加關鍵字 var。

應該注意的是：

- (1) 與變量形參對應的實參只能是變量名，而不能是表達式。
- (2) 與變量形參對應的實參可以根據需要決定是否事先有值。
- (3) 變量形參與對應的實參的類型必須完全相同。
- (4) 對變量形參，運行時不另外開闢存儲單元，而是與對應的實參使用相同的存儲單元。也就是說，調用子程序時，是將實參的地址傳送給對應的變量形參。
- (5) 當控制返回到調用程序後，變量形參的存儲單元不釋放，但變量形參本身無定義，即不得再使用。

(6) 選用形式參時，到底是使用值形參還是變量形參，應慎重考慮。值形參需要另開闢存儲空間，而變量形參會帶來一些副作用。一般在函數中使用值形參，而在過程中才使用變量形參，但也有例外。

例 12.3：寫出下列兩個程序的運行結果。

```
program ex1;
var a,b:integer;

procedure swap(x,y:integer);
var t:integer;
begin
  t:=x;x:=y;y:=t;
end;

begin
  a:=1;b:=2;
  writeln(a:3,b:3);
  swap(a,b);
  writeln(a:3,b:3);
end.

program ex2;
var a,b:integer;

procedure swap(var x,y:integer);
var t:integer;
begin
  t:=x;x:=y;y:=t;
end;

begin
  a:=1;b:=2;
  writeln(a:3,b:3);
  swap(a,b);
  writeln(a:3,b:3);
end.
```

分析：這兩個程序唯一的區別是 ex1 中將 x, y 作為值形參，而 ex2 中將 x, y 作為變量形參，因此在 ex2 中對 x, y 的修改實際上是對調用該過程時與它們對應的變量 a, b 的修改，故最後，a, b 的值為 2, 1。而 ex1 中調用 swap 過程時，只是將 a, b 的值傳遞給 x, y，之後在過程中的操作與 a, b 無關。

答：ex1 的運行結果為：

```
1 2
1 2
```

ex2 的運行結果為：

```
1 2
2 1
```

(二) 全程變量、局部變量及它們的作用域

在主程序的說明部分和子程序的說明部分均可以說明變量，但它們的作用範圍是特定的。

1. 局部量及其作用域

在介紹過程和函數的說明時，我們曾指出，凡是在子程序內部作用的變量，應該在本子程序內加以說明。這種在子程序內部說明的變量稱為局部變量。形式參數也只是在該子程序中有效，因此也屬於局部變量。

一個變量的作用域是指在程序中能對此變量進行存取的程序範圍。因此，局部變量的作用域就是其所在的子程序。實際上，局部變量只是當其所在的子程序被調用時才具有確定的存儲單元，當控制從子程序返回到調用程序後，局部變量的存儲單元就被釋放，從而變得無定義。

事實上，在子程序內定義的標號、符號常量、類型、子程序也與局部變量具有相同的作用域。

2. 全程量及其作用域

全程量是指在主程序的說明部分中說明的量。全程量的作用域分兩種情況：

- (1) 當全程量和局部量不同名時，其作用域是整個程序範圍（自定義起直到主程序結束）。
- (2) 當全程量和局部量同名時，全程量的作用域不包含局部量的作用域。

例 12.4：寫出下列程序的運行結果：

```
program ex12_4;
var x,y:integer;

procedure a;
var x:integer;
begin
  x:=2;
  writeln('#',x,'#');
  writeln('#',y,'#');
end; {of a}

begin {main program}
  x:=1;y:=2;
  writeln('*',x,'*',y);
  a;
  writeln('***',x,'***',y);
end.
```

PASCAL 教程

分析：程序中 x ， y 是全局變量，但在過程 a 中也有變量 x ，故全程變量 x 的作用域為除過程 a 外的任何地方。而 y 的作用域包含了子程序 a ，即整個程序。

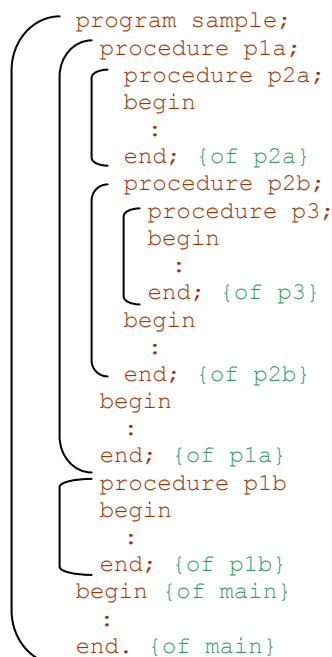
答：運行結果如下：

```
*1*2
#2#
#2#
***1***2
```

評注：變量作用域內對變量的操作都是對同一存儲單元中的量進行的。

四、過程和函數的嵌套

Pascal 語言中，使用過程和函數，能使程序設計簡短，便於閱讀，節省存貯單元和編譯時間。程序往往設計成分層結構，由一個主程序和若干個過程及函數組成。在過程或函數中，還可以說明另一些過程或函數，即過程或函數可以分層嵌套。在同一層中亦可說明幾個並列的過程或函數。例如：



```
program sample;
  procedure p1a;
    procedure p2a;
      begin
        :
      end; {of p2a}
    procedure p2b;
      procedure p3;
        begin
          :
        end; {of p3}
      begin
        :
      end; {of p2b}
    begin
      :
    end; {of p1a}
  procedure p1b
  begin
    :
  end; {of p1b}
  begin {of main}
    :
  end. {of main}
```

上例過程的分層嵌套關係如下：0 層主程序 $sample$ 內並列兩個 1 層過程 $p1a$ 和 $p1b$ 。過程 $p1a$ 又嵌套兩個 2 層過程 $p2a$ 和 $p2b$ ，2 層的第二過程 $p2b$ 又嵌套過程 $p3$ ， $p3$ 就是第 3 層。其中 $p1b$ ， $p2a$ 和 $p3$ 不再嵌套別的過程，稱為基本過程。這種分層結構的程序設計，特別要注意局部變量的使用範圍和過程調用的要求。

在主程序 $sample$ 中定義的變量，可以在所有的過程中使用，主程序可調用 $p1a$ 和 $p1b$ 兩個過程。過程 $p1a$ 中定義的變量，只能在 $p2a$ ， $p2b$ 和 $p3$ 中使用。它能調用 $p2a$ ， $p2b$ 兩個過程，而不能調用 $p3$ 和 $p1b$ 。在過程 $p1b$ 中定義的變量，只能在 $p1b$ 中使用，它只能調用過程 $p1a$ 。過程 $p2a$ 不能調用任何過程。過程 $p2b$ 可以調用並列過程 $p2a$ 和 $p3$ ，而過程 $p3$ 可以調用 $p2a$ 過程。

過程調用是有條件的，過程定義在先，調用在後。同一層過程，後說明的過程可以調用先說明的過程。如果要調用在它後面定義的過程（或函數），可使用〈向前引用〉FORWARD 這個擴充標識符。要注意的是〈向前引用〉過程（或函數）首部中形式參數表寫一次即可，不必重複。如：

```
procedure extend(var a,b:integer);
forward;
```

表示過程 $extend$ 〈向前引用〉。因此，過程 $extend$ 的說明部分只須如下書寫：

```
procedure extend;
<說明部分>
begin
:
end;
```

五、子程序（模塊化）結構的程序設計

例 12.5：對 6 到 60 的偶數驗證哥德巴赫猜想：不小於 6 的偶數可分解成兩個素數之和。

分析：用布爾型函數 $prime(x)$ 判斷 x 是否是素數，若是，函數值為真，否則，函數值為假。算法如下所示。

```
1. t:=6
2. while t≤60 do
3. t1←1;
4. repeat
5. t1←t1+2; /* 找下一個素數 a */
6. until prime(t1) and prime(t-t1); /*直到 a, b 都是素數*/
7. writeln(i, '=', t1, '+', t-t1);
8. t←t+2;
9. endwhile
```

源程序如下：

```
program ex12_5;
var t,t1:integer;

function prime(x:integer):boolean;
var i:integer;
begin
  if x=1 then prime:=false
  else if x=2
    then prime:=true
    else
      begin
        prime:=true;
        i:=2;
        while (i<=round(sqrt(x))) and (x mod i<>0) do
          i:=i+1;
        if i<=round(sqrt(x)) then prime:=false;
        end;
      end;
end; {of prime}

begin
  t:=6;
  while t<=60 do
    begin
      t1:=1;
      repeat
        t1:=t1+2;
      until prime(t1) and prime(t-t1);
      writeln(t, '=', t1, '+', t-t1);
      t:=t+2;
    end;
  end.
```

例 12.6 編寫一個給一個分數約分的程序。

源程序如下：

```
program ex12_6; 變量參數
var a,b:integer;
               ↓
procedure common(var x,y:integer);
var i,j,k:integer;
begin
  {求 x,y 的最大公約數}
  i:=x; j:=y;
  repeat
    k:=i mod j;
    i:=j; j:=k;
  until k=0;
  {對 x,y 進行約分}
  x:=x div i; y:=y div i;
end;

begin
  write('Input a,b='); readln(a,b);
  common(a,b);
  writeln(a,b:5);
end.
```

如輸入：

Input a,b=12 8

則輸出：

3 2

練習十二

1、輸入 5 個正整數求它們的最大公約數。（提示：可用一個數組將 5 個數存放起來，然後求第一個數和第二個數的公約數，再求第三個數與前兩個數公約數的公約數，這樣求得前三個整數最大公約數.....如此類推可求出 5 個整數的最大公約數）

2、給一維數組輸入任意 6 個整數，假設為：

7 4 8 9 1 5

請建立一個具有以下內容的方陣：

7 4 8 9 1 5

4 8 9 1 5 7

8 9 1 5 7 4

9 1 5 7 4 8

1 5 7 4 8 9

5 7 4 8 9 1

（請用子程序編寫）。

3、求兩個正整數的最小公倍數。

4、輸入一個任意位的正整數，將其反向輸出。

5、有五位同學，其各科成績如下：

學號	數學	語文	英語	總分	名次
----	----	----	----	----	----

1	108	97	90		
---	-----	----	----	--	--

2	98	88	100		
---	----	----	-----	--	--

3	100	43	89		
---	-----	----	----	--	--

4	84	63	50		
---	----	----	----	--	--

5	97	87	100		
---	----	----	-----	--	--

(1) 編寫一個過程 enter，輸入每個學生成績並計算各人的總分。

(2) 編寫過程 minci，用以排出每個人的名次。

(3) 按學號順序輸出。

第十三課 動態數據類型

前面介紹的各種簡單類型的數據和構造類型的數據屬於靜態數據。在程序中，這些類型的變量一經說明，就在內存中佔有固定的存儲單元，直到該程序結束。

程序設計中，使用靜態數據結構可以解決不少實際問題，但也有不便之處。如建立一個大小未定的姓名表，隨時要在姓名表中插入或刪除一個或幾個數據。而用新的數據類型——指針類型。通過指針變量，可以在程序的執行過程中動態地建立變量，它的個數不再受限制，可方便地高效地增加或刪除若干數據。

一、指針的定義及操作

(一) 指針類型和指針變量

在 pascal 中，指針變量（也稱動態變量）存放某個存儲單元的地址；也就是說，指針變量指示某個存儲單元。

指針類型的格式為：

^基類型

說明：

(1) 一個指針只能指示某一種類型數據的存儲單元，這種數據類型就是指針的基類型。基類型可以是除指針、文件外的所有類型。例如，下列說明：

```
type pointer=^Integer;
var p1,p2:pointer;
```

定義了兩個指針變量 p1 和 p2，這兩個指針可以指示一個整型存儲單元（即 p1、p2 中存放的是某存儲單元的地址，而該存儲單元恰好能存放一個整型數據）。

(2) 和其它類型變量一樣，也可以在 var 區直接定義指針型變量。例如：

```
var a:^real; b:^boolean;
```

又如：

```
type
  person=record
    name:string[20];
    sex:(male,female);
    age:1..100
  end;
var pts:^person;
```

(3) pascal 規定所有類型都必須先定義後使用，但只有在定義指針類型時可以例外，如下列定義是合法的：

```
type
  pointer=^rec;
  rec=record
    a:integer;
    b:char
  end;
```

(二) 開關和釋放動態存儲單元

1. 開關動態存儲單元

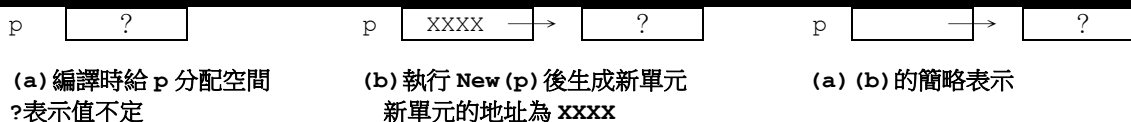
在 pascal 中，指針變量的值一般是通過系統分配的，開關一個動態存儲單元必須調用標準過程 new。new 過程的調用的一般格式：

New (指針變量)

功能：開關一個存儲單元，此單元能存放的數據的類型正好是指針的基類型，並把此存儲單元的地址賦給指針變量。

說明：

(1) 這實際上是給指針變量賦初值的基本方法。例如，設有說明：var p:^Integer; 這只定義了 p 是一個指示整型存儲單元的指針變量，但這個單元尚未開關，或者說 p 中尚未有值（某存儲單元的首地址）。當程序中執行了語句 new(p) 才給 p 賦值，即在內存中開關（分配）一個整型變量存儲單元，並把此單元的地址放在變量 p 中。示意如下圖：



內存單元示意圖

(2) 一個指針變量只能存放一個地址。如再一次執行 $\text{New}(p)$ 語句，將在內存中開闢另外一個新的整型變量存儲單元，並把此新單元的地址放在 p 中，從而丟失了原存儲單元的地址。

(3) 當不再使用 p 當前所指的存儲單元時，可以通過標準過程 Dispose 釋放該存儲單元。

2. 釋放動態存儲單元

dispose 語句的一般格式：

dispose (指針變量)

功能：釋放指針所指向的存儲單元，使指針變量的值無定義。

(三) 動態存儲單元的引用

在給一個指針變量賦以某存儲單元的地址後，就可以使用這個存儲單元。引用動態存儲單元一般格式：

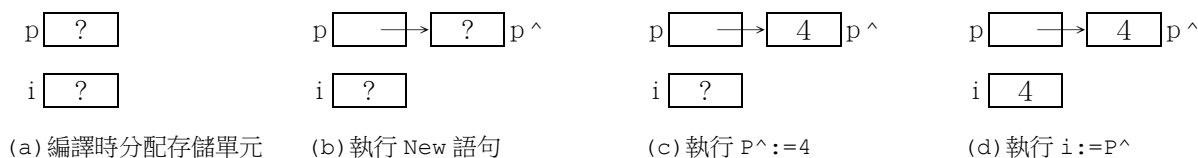
<指針變量>^

說明：

- (1) 在用 New 過程給指針變量開闢了一個它所指向的存儲單元後，要使用此存儲單元的唯一方法是利用該指針。
- (2) 對動態存儲單元所能進行的操作是該類型（指針的基類型）所允許的全部操作。

例 13.1：設有下列說明：
 $\text{var } p:\text{^integer}; i:\text{integer};$
 畫出執行下列操作後的內存示意圖：
 $\text{New}(p); p^:=4; i:=p^;$

解：如下圖所示。



內存單元示意圖

(四) 對指針變量的操作

前已述及，對指針所指向的變量（如 $p^$ ）可以進行指針的基類型所允許的全部操作。對指針變量本身，除可用 New 、 Dispose 過程外，尚允許下列操作：

1. 具有同一基類型的指針變量之間相互賦值

例 13.2：設有下列說明與程序段：

```
var p1,p2,p3:^integer;
begin
  New(P1); New(P2); New(P3);
  P1:=P2; P2:=P3;
end;
```

2. 可以給指針變量賦 nil 值

nil 是 PASCAL 的關鍵字，它表示指針的值為“空”。例如，執行： $p1:=\text{nil}$ 後， $p1$ 的值是有定義的，但 $p1$ 不指向任何存儲單元。

3. 可以對指針變量進行相等或不相等的比較運算

在實際應用中，通常可以在指針變量之間，或指針變量與 nil 之間進行相等（ $=$ ）或不相等（ $<>$ ）的比較，比較的結果為布爾量。

例 13.3：輸入兩個整數，按從小到大打印出來。

分析：不用指針類型可以很方便地編程，但為了示例指針的用法，我們利用指針類型。定義一個過程 swap 用以交換兩個指針的值。

源程序如下：

```
program ex13_3;
type pointer=^integer;
var p1,p2:pointer;

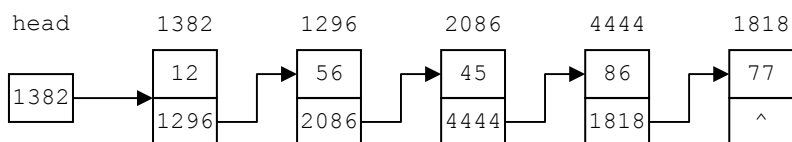
procedure swap(var q1,q2:pointer);
var q:pointer;
begin
  q:=q1;
  q1:=q2;
  q2:=q;
end;

begin
  new(p1);new(p2);
  write('Input 2 data:');readln(p1^,p2^);
  if p1^>p2^ then swap(p1,p2);
  writeln('Output 2 data:',p1^:4,p2^:4);
end.
```

二、鏈表結構

設有一批整數(12, 56, 45, 86, 77,,), 如何存放呢? 當然我們可以選擇以前學過的數組類型。但是, 在使用數組前必須確定數組元素的個數。如果把數組定義得大了, 就會有大量空間存儲單元, 定義得小了, 又會在運行中發生下標越界的錯誤, 這是靜態存儲分配的局限性。

利用本章介紹的指針類型可以構造一個簡單而實用的動態存儲分配結構——鏈表結構。下圖是一個簡單鏈表結構示意圖：



其中：

- 每個框表示鏈表的一個元素，稱為結點。
- 框的頂部表示了該存儲單元的地址(當然，這裏的地址是假想的)。
- 每個結點包含兩個域：一個域存放整數，稱為數據域，另一個域存放下一個結點(稱為該結點的後繼結點，相應地，該結點為後繼結點的前趨結點)的地址。
- 鏈表的第一個結點稱為表頭，最後一個結點表尾，稱為指針域；
- 指向表頭的指針 head 稱為頭指針(當 head 為 nil 時，稱為空鏈表)，在這個指針變量中存放了表頭的地址。
- 在表尾結點中，由指針域不指向任何結點，一般放入 nil。

(一) 鏈表的基本結構

由上圖可以看出：

(1) 鏈表中的每個結點至少應該包含兩個域：一是數據域，一是指針域。因此，每個結點都是一個記錄類型，指針的基類型也正是這個記錄類型。因此，head 可以這樣定義：

```
type
  pointer=^rec;
  rec=record
    data:integer;
    next:pointer;
  end;
var head:pointer;
```


(2) 相鄰結點的地址不一定是連續的。整個鏈表是通過指針來順序訪問的，一旦失去了一個指針值，後面的元素將全部丟失。

(3) 與數組結構相比，使用鏈表結構時；可根據需要採用適當的操作步驟使鏈表加長或縮短，而使存儲分配具有一定的靈活性。這是鏈表結構的優點。

(4) 與數組結構相比，數組元素的引用比較簡單，直接用“數組名[下標]”即可，因為數組元素佔用連續的存儲單元，而引用鏈表元素的操作卻比較複雜。

(二) 單向鏈表的基本操作

上圖所示的鏈表稱為單向鏈表。下面我們通過一些例題來說明對單向鏈表的基本操作，並假設類型說明如前所述。


 **例 13.4**：編寫一個過程，將讀入的一串整數存入鏈表，並統計整數的個數。

分析：過程的輸入為一串整數，這在執行部分用讀語句完成。過程的輸出有兩個：一是鏈表的頭指針，一是整數的個數，這兩個輸出可以用變量形參來實現。

由於不知道整數的個數，我們用一個特殊的 9999 作為結束標記。

過程如下：

```
procedure creat(var h:pointer; var n:integer);
var p,q:pointer;x:integer;
begin
  n:=0;h:=nil; read(x);
  while x<>9999 do
  begin
    New(p);
    n:=n+1;p^.data:=x;
    if n=1 then h:=p
    else q^.next:=p;
    q:=p;read(x)
  end;
  if h<>nil then q^.next:=nil;
  Dispose(p);
end;
```

 **例 13.5**：編一過程打印鏈表 head 中的所有整數，5 個一行。

分析：設置一個工作指針 P，從頭結點順次移到尾結點，每移一次打印一個數據。

過程如下：

```
procedure print(head:pointer);
var p:pointer; n:integer;
begin
  n:=0;p:=head;
  while p<>nil do
  begin
    write(p^.data:8);n:=n+1;
    if n mod 5=0 then writeln;
    p:=p^.next;
  end;
  writeln;
end;
```

(三) 鏈表結點的插入與刪除

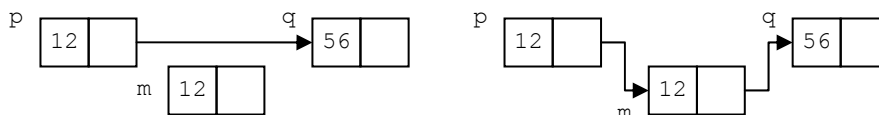
鏈表由於使用指針來連接，因而提供了更多了靈活性，可以插入刪除任何一個成分。

設有如下定義：

```
type
  pointer=^rec;
  rec=record
    data:integer;
    next:pointer
  end;
var head:pointer;
```

1. 結點的插入

如下圖所示，要在 P 結點和 Q 結點之間插入一個結點 m，其操作如下：



只要作如下操作即可：

```
New(m);  
read(m^.data);  
m^.next:=q;  
p^.next:=m;
```

例 13.6：設鏈表 head 中的數據是按從小到大順序存放的，在鏈表中插入一個數，使鏈表仍有序。

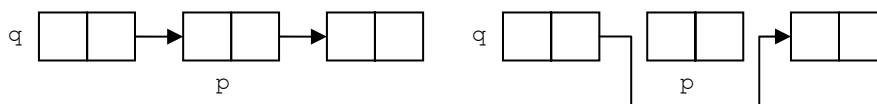
分析：顯然，應分兩步：查找、插入。設 po 指向要插入的結點，若僅知道 po 應插在 p 之前（作為 p 的前趨結點）是無法插入的，應同時知道 p 的前趨結點地址 q。

當然，如果插在鏈表原頭結點這前或原鏈表為空表或插在原尾結點之後，則插入時又必須作特殊處理。過程如下：

```
procedure inserting(var head:pointer; x:integer);  
var po,p,q:pointer;  
begin  
  new(po);po^.data:=x;  
  p:=head;  
  if head=nil then {原表為空表}  
  begin  
    head:=po;po^.next:=nil;  
  end  
  else  
  begin  
    while (p^.data<x) and (p^.next<>nil) do  
      begin  
        q:=p;p:=p^.next  
      end;  
    if p^.data>=x then {不是插在原尾結點之後}  
    begin  
      if head=p then head:=po {插在原頭結點之前}  
      else q^.next:=po;  
        po^.next:=p  
      end  
    else {插在原尾結點之後}  
    begin  
      p^.next:=po;  
      po^.next:=nil  
    end;  
  end;  
end;
```

2. 結點的刪除

如下圖所示，要在刪除結點 P 的操作如下：



要刪除結點 P，則只要將其前趨結點的指針域指向 P 的後繼結點即可。

```
q^.next:=p^.next;  
dispose(p);
```

例 13.7：將鏈表 head 中值為 x 的第一個結點刪除

分析：有三種情況存在：頭結點的值為 x；除頭結點外的某個結點值為 x；無值為 x 的結點。為將前兩種情況統一起來，我們在頭結點之前添加一個值不為 x 的哨兵結點。

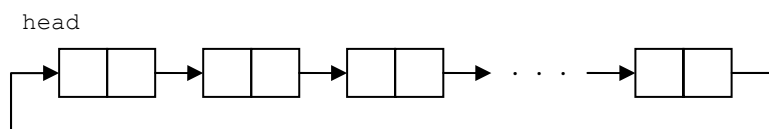
算法分兩步：查找、刪除。過程如下：

```
procedure deleteing(var head:pointer; x:integer);  
var p,q:pointer;  
begin  
  New(p);p^.data:=x-1;p^.next:=head;  
  head:=p; {以上為添加哨兵結點}  
  while (x<>p^.data) and (p^.next<>nil) do  
    begin  
      q:=p;  
      p:=p^.next  
    end;  
  if x=p^.data then {存在值為 x 的結點}
```

```
begin
    q^.next:=p^.next; {刪除 x}
    dispose(p);
end
else writeln('Not found!');
p:=head^.next; {刪除哨兵}
head:=p;
dispose(p)
end;
```

(四) 環形鏈表結構

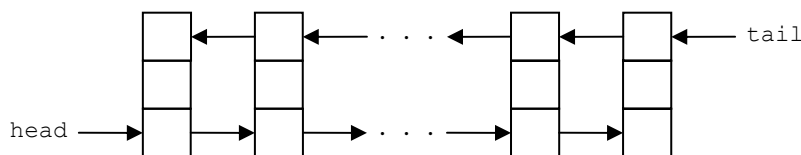
在單向鏈表中，表尾結點的指針為空。如果讓表尾結點的指針域指向表頭結點，則稱為單向環形鏈表，簡稱單鏈環。如圖所示。



單鏈環示意圖

(五) 雙向鏈表結構

單鏈表中，每個結點只有一個指向其後繼結點的指針域。如果每個結點不僅有一個指向其後繼結點的指針域，還有一個指向其前趨的指針域，則這種鏈表稱為雙向鏈表。如圖所示。




雙向鏈表示意圖

可用如下定義一個數據域為整型的雙向鏈表：

```
type
    pointer=^node;
    node=record
        prev:pointer;
        data:integer;
        next:pointer;
    end;
```

對雙向鏈表的插入、刪除特別方便。與單向鏈環相似，我們還可定義雙向鏈環。

三、綜合例析

 **例 13.8**：讀入一串以“#”為結束標誌的字符，統計每個字符出現的次數。

分析：設置一個鏈表存放，每讀入一個字符，就從鏈表的頭結點向後掃描鏈表，如果在鏈表中此字符已存在，則其頻率加 1，否則將該字符的結點作為鏈表的新頭結點，相應頻率為 1。

源程序如下：

```
program ex13_8;
type
    ref=^letters;
    letters=record
        key:char;
        count:integer;
        next:ref;
    end;
var
    k:char;
    sentinel,head:ref;

procedure search(x:char);{統計 x 出現頻率}
var w:ref;
begin
    w:=head;
    sentinel^.key:=x;
    while (w^.key<>x) do w:=w^.next;
    if w<>sentinel then {鏈表中已存在 x }
        w^.count:=w^.count+1
```

```
else
begin {鏈表中未有 x }
  w:=head;new(head);
  with head^ do
  begin
    key:=x;count:=1;next:=w;
  end
end;
end;{of search}

procedure printlist(w:ref);
begin
  while w<>sentinel do
  begin
    writeln(w^.key:2,w^.count:10);
    w:=w^.next;
  end;
end;{of printlist}

begin {main program}
  new(sentinel);
  with sentinel^ do
  begin
    key:='#';count:=0;next:=nil;
  end;
  head:=sentinel;
  read(k);
  while k<>'#' do
  begin
    search(k);read(k);
  end;
  printlist(head);
end.
```

也可編程如下：

```
program ex13_8;
type
  ref=^letters;
  letters=record
    key:char;
    count:integer;
    next:ref;
  end;
var
  k:char;
  head:ref;

procedure search(x:char);{統計 x 出現頻率}
var w,np:ref;
begin
  w:=head;
  while (w^.key<>x) and (w^.next<>nil) do w:=w^.next;
  if w^.key=x then {鏈表中已存在 x }
    w^.count:=w^.count+1
  else
  begin {鏈表中未有 x }
    if w^.key='#' then
      np:=w
    else
    begin
      new(np);
      w^.next:=np;
    end;
    np^.key:=x;
    np^.count:=1;
    np^.next:=nil;
  end;
end;{of search}

procedure printlist(w:ref);
begin
```

```
    if w^.key='#' then exit;
  repeat
    writeln(w^.key:2,w^.count:10);
    w:=w^.next;
  until w=nil;
end;{of printlist}

begin {main program}
  new(head);
  head^.key:='#';
  head^.next:=nil;
  read(k);
  while k<>'#' do
  begin
    search(k);read(k);
  end;
  printlist(head);
  dispose(head);
end.
```

📖 例 13.9：用鏈表重寫篩法求 2~100 之間所有素數程序。

源程序如下：

```
program ex13_9;
uses wincrt;
type
  link=^code;
  code=record
    key:integer;
    next:link;
  end;
var head:link;

procedure printlist(h:link);{打印鏈表h}
var p:link;
begin
  p:=h;
  while p<>nil do
  begin
    write(p^.key,'-->');
    p:=p^.next;
  end;
end;

procedure buildlink;{建立鏈表}
var p,q:link;
    i:integer;
begin
  new(head);
  head^.key:=2;
  p:=head;
  for i:=3 to 100 do
  begin
    new(q);
    q^.key:=i;
    q^.next:=nil;
    p^.next:=q;
    p:=q;
  end;
end;

procedure prime;{篩法將找到的素數的倍數從鏈表中刪除}
var h,p,q:link;
begin
  h:=head;
  while h<>nil do
  begin
    p:=h;q:=p^.next;
    while q<>nil do
      if (q^.key mod h^.key=0) then
        begin
```

```
        p^.next:=q^.next;
        dispose(q);
        q:=p^.next;
    end
    else
    begin
        p:=q;
        q:=q^.next;
    end;
    h:=h^.next;
end;
end;

begin {main program}
    buildlink;
    writeln;
    prime;
    printlist(head);
end.
```

練習十三

- 1、圍繞著山頂有 10 個洞，一隻兔子和一隻狐狸各住一個洞，狐狸總想吃掉兔子。一天兔子對狐狸說，你想吃我有一個條件，你先把洞編號 1 到 10。你從第 10 號洞出發，先到第 1 號洞找我，第二次隔一個洞找我，第三次隔兩個洞找我，以後依次類推，次數不限。若能找到我，你就可以飽餐一頓，在沒找到我之前不能停止。狐狸一想只有 10 個洞，尋找的次數又不限，哪有找不到的道理，就答應了條件。結果就是沒找著。

利用單鏈環編程，假定狐狸找了 1 0 0 0 次，兔子躲在哪個洞裏才安全。

- 2、某醫院病房的訂位管理中，將病人的記錄按姓名的字母順序排成一個鏈表。試編寫程序，從鍵盤上輸入下列字母，就可對病員記錄進行管理：
- (1) i —— 新病人入院（插入一個病員記錄）。
 - (2) d —— 病人出院（刪除一個病員記錄，並顯示該記錄）。
 - (3) s —— 查詢某病員記錄（顯示該記錄或“未找到”）。
 - (4) q —— 在屏幕上列出所有的病員記錄並結束程序。

- 3、編寫一個簡單的大學生新生入校登記表處理程序。

第十四課 文件

在 DOS 操作中，我們所談及的文件稱之為外部文件。外部文件是存儲在外部設備上，如：外存儲器上，可由計算機操作系統進行管理，如用 `dir`、`type` 等命令直接對文件進行操作。

Pascal 所談及的文件，稱之為內部文件。內部文件的特點是文件的實體（實際文件）也是存儲在外存儲器上，成為外部文件的一分子，但在使用時必須在程序內部以一定的語句與實際文件聯繫起來，建立一一對應的關係，用內部文件的邏輯名對實際文件進行操作。內部文件的邏輯名必須符合 Pascal 語言標識符的取名規則。

Pascal 中的文件主要用於存放大量的數據。如：成績管理，原始數據很多，使用文件先將其存入磁盤，通過程序讀出文件中的數據再進行處理，比不使用文件要來得方便、有效。

Pascal 中的一個文件定義為同一類型的元素組成的線性序列。文件中的各個元素按一定順序排列，可以從頭至尾訪問每一個元素，從定義上看，文件與數組相似，但它們之間有著明顯不同的特徵，主要表現在：

（1）文件的每一個元素順序存貯於外部文件設備上（如磁盤上）。因此文件可以在程序進行前由 Pascal 程序或用文字編輯軟件，如 `edit`、`ws`、Turbo Pascal 的 `edit` 命令等產生，或在運行過程中由程序產生，且運行完後，依然存貯在外部設備上。

（2）在系統內部，通過文件指針來管理對文件的訪問。文件指針是一個保存程序在文件中位置蹤跡的計算器，在一固定時刻，程序僅能對文件中的一個元素進行讀或寫的操作，在向文件寫入一個元素或從文件讀取一個元素後，相應的文件指針就前進到下一元素位置。而數組是按下標訪問。

（3）在文件類型定義中無需規定文件的長度即元素的個數，就是說元素的數據可動態改變，一個文件可以非常之大，包含許許多多元素，也可以沒有任何元素，即為一個空文件。而數組的元素個數則是確定的。

使用文件大致有以下幾個步驟：

- （1）說明文件類型，定義文件標識符；
- （2）建立內部文件與外部文件的聯繫；
- （3）打開文件；
- （4）對文件進行操作；
- （5）關閉文件。

Turbo Pascal 將文件分為三類：文本文件（順序）、有類型文件（順序或隨機）和無類型文件（順序或隨機）。下面將介紹這些文件及其操作。

一、文本文件

文本文件又稱為正文文件或行文文件，可供人們直接閱讀，是人機通信的基本數據形式之一。文本文件可用文字編輯程序（如 DOS 的 `edit` 或 Turbo Pascal 的編輯命令 `edit`）直接建立、閱讀和修改，也可以由 P A S C A L 程序在運行過程中建立。

1．文本文件的定義：

文本文件的類型為 TEXT，它是由 ASCII 字符組成的，是 Pascal 提供的標準文件之一。標準文件 TEXT 已由 Pascal 說明如下：

```
TYPE TEXT=FILE OF CHAR;
```

因此，TEXT 同標準類型 INTEGER、REAL 等一樣可以直接用於變量說明之中，無需再由用戶說明。例如：


```
VAR F1, F2: TEXT;
```

這裏定義了兩個文本文件變量 F1 和 F2。

2．文本文件的建立

文本文件的建立有兩種方法：

- (1) 直接用 Turbo Pascal 的 Edit 建立原始數據文件。

 **例 14.1**：將下表中的數據存入名為 **A.dat** 的文件中。

```
3 4
29 30 50 60
```


80 90 70 75
60 50 70 45

操作步驟：

- 1) 進入 Turbo Pascal 的編輯狀態；
 - 2) 輸入數據；
 - 3) 存盤，文件名取 A.dat。
- 此時，已將數據存入文本文件 A.dat 中。文本文件也可用 DOS 中的 Edit 等軟件建立。

(2) 用程序的方式建立中間數據或結果數據文件。

用程序的方式建立文件操作步驟為：

- 1) 定義文本文件變量；
- 2) 把一外部文件名賦於文本文件變量，使該文本文件與一相應外部文件相關聯；

命令格式：

ASSIGN (f, name)

f 為定義的文本文件變量

name 為實際文件文件名

如：ASSIGN(F1, 'FILEIN.DAT')

或：ASSIGN(F1, 'PAS\FILEIN.RES')

這樣在程序中對文本文件變量 F1 的操作，也就是對外部實際文件 FILEIN.DAT 或 FILEIN.RES 的操作。上例中文件 FILEIN.DAT 是存貯在當前目錄中，而文件 FILEIN.RES 則是存貯在 PAS 子目錄中。

- 3) 打開文本文件，準備寫；

命令格式 1：

REWRITE (f)

功能：創建並打開新文件準備寫，若已有同名文件則刪除再創建

命令格式 2：

APPEND (f)

功能：打開已存在的文件並追加

- 4) 對文件進行寫操作；

命令格式：


WRITE (f, <項目名>) 或 WRITELN (f, <項目名>)

功能：將項目內容寫入文件 f 中

- 5) 文件操作完畢後，關閉文件。

命令格式：

CLOSE (f)

 **例 14.2**：從鍵盤上讀入例 14.1 的表數據，用程序寫入名為 B.dat 的文件中。

```
program ex14_2;
var
  I, j, n, m, x: integer;
  f: text;
begin
  write('n,m='); readln(n,m);
  assign(f, 'b.dat');
```

```
rewrite(f);
writeln(f,n,' ',m);
for i:=1 to n do
begin
  for j:=1 to m do
  begin
    read(x);
    write(f,x,' ');
  end;
  readln; writeln(f);
end;
close(f);
end.
```

3 · 讀取文本文件

文本文件內容讀出操作步驟：

- (1) 定義文本文件變量；
- (2) 用 ASSIGN(f, name) 命令，將內部文件 f 與實際文件 name 聯繫起來；
- (3) 打開文本文件，準備讀；

命令格式：

RESET (f)

功能：打開文件 f 準備讀

READ (f, <變量名表>) READLN (f, <變量名表>)

功能：將文件 f 中指針指向的數據讀到變量中

文本文件提供了另外兩個布爾函數，在文本的操作中很有用處，它們是：

回送行結束符狀態：

EOLN (f)

回送文件結束符狀態：

EOF (f)

- (4) 文件操作完畢，用 CLOSE (f) 命令關閉文件。

📖 **例 14.3**：讀出例 14.1 建立的文本文件，並輸出。

```
program ex14_3;
var
  i,j,n,m,x:integer;
  f:text;
begin
  assign(f,'b.dat');
  reset(f);
  read(f,n,m);
  writeln(n,' ',m);
  for i:=1 to n do
  begin
    for j:=1 to m do
    begin
      read(f,x);
      write(x,' ');
    end;
    writeln;
  end;
  close(f);
end.
```

由於文本文件是以 ASCII 碼的方式存儲，故查看文本文件的內容是極為方便，在 DOS 狀態可使用 DOS 中 TYPE 等命令進行查看，在 Turbo Pascal 中可以象取程序一樣取出文件進行查看。


4 · 文本文件的特點

(1) 行結構

文本文件由若干行組成，行與行之間用行結束標記隔開，文件末尾有一個文件結束標記。由於各行長度可能不同，所以無法計算出給定行在文本文件中的確定位置，從而只能順序地處理文本文件，而且不能對一文本文件同時進行輸入和輸出。

(2) 自動轉換功能

文本文件的每一個元素均為字符型，但在將文件元素讀入到一個變量（整型，實型或字符串型）中時，Pascal 會自動將其轉換為與變量相同的數據類型。與此相反在將一個變量寫入文本文件時，也會自動轉移為字符型。

 **例 14.4：**某學習小組有 10 人，參加某次測驗，考核 6 門功課，統計每人的總分及各門的平均分，將原始數據及結果數據放入文本文件中。

分析：

(1) 利用 Turbo Pascal 的 EDIT 建立原始數據文件 TESTIN.DAT 存貯在磁盤中，其內容如下：

```
10 6
1 78 89 67 90 98 67
2 90 93 86 84 86 93
3 93 85 78 89 78 98
4 67 89 76 67 98 74
5 83 75 92 78 89 74
6 76 57 89 84 73 71
7 81 93 74 76 78 86
8 68 83 91 83 78 89
9 63 71 83 94 78 95
10 78 99 90 80 86 70
```

(2) 程序讀入原始數據文件，求每人的總分及各科的平均分；

(3) 建立結果數據文件，文件名為 TEXTIN.RES


程序：

```
program ex14_4;
var
  f:text;
  i,j,n,m,s:integer;
  data:array [1..10,0..7] of integer; {第 7 欄為每人的各科總分}
  p:array [1..6] of integer; {同一科全班總分}

procedure in_data; {從文本文件中讀取數據}
var i,j:integer;
begin
  assign(f,'textin.dat');
  reset(f);
  read(f,n); readln(f,m);
  for i:=1 to n do
    for j:=0 to m do
      read(f,data[i,j]);
  close(f);
end;

procedure out_data; {將數據寫入文件}
var i,j:integer;
begin
  assign(f,'textin.res');
  rewrite(f);
  for i:=1 to n do
    begin
      for j:=0 to m+1 do
        write(f,data[i,j]:4);
      writeln(f);
    end;
  write(f,' ');
  for j:=1 to m do
    write(f,p[j]:4);
  writeln(f);
  close(f);
end;
```

```
begin {of main}
  in_data;
  fillchar(p,sizeof(p),0); {函數 fillchar 用以將 p 中所有元素置為 0}
  for i:=1 to n do
  begin
    s:=0;
    for j:=1 to m do {求每人總分，各科總分}
    begin
      s:=s+data[i,j];
      p[j]:=p[j]+data[i,j];
    end;
    data[i,m+1]:=s; {第 m+1 欄為每人的各科總分}
  end;
  for i:=1 to n do {顯示輸出數據}
  begin
    for j:=0 to m+1 do
      write(data[i,j]:4);
    writeln;
  end;
  write(' ');
  for j:=1 to m do {求各科平均分}
  begin
    p[j]:=p[j] div n;
    write(p[j]:4);
  end;
  writeln;
  out_data;
end.
```

-  **例 14.5：**讀入一個行長不定的文本文件。排版，建立一個行長固定為 60 個字符的文件，排版要求：
- (1) 當行末不是一個完整單詞時，行最後一個字符位用 '-' 代替，表示與下一行行頭組成完整的單詞；
 - (2) 第一行行頭為兩個空格，其餘各行行頭均不含有空格。

分析

- (1) 建立原始數據文件。
- (2) 程序邊讀入原始數據文件內容，邊排版。
- (3) 每排完一行行長為 60 字符，並符合題中排版條件，寫入目標文件中。

設原始數據 TEXTCOPY.DAT 文件內容如下：

```
Pavel was arrested.
That dat Mother did not light the stove.
Evening came and a cold wind was blowing.
There was a knock at the window.
Then another.
Mother was used to such knocks,but this time she gave a little start of joy.
Throwing a shawl over her shoulders,she opened the door.
```

程序：

```
program ex14_5;
const
  maxlen=60;
var
  f1,f2:text;
  ch,ch1:char;
  s:string[maxlen];
  i,j:integer;
  inname,outname:string[20];
begin
  write('Enter textin filename:');
  readln(inname);
  assign(f1,inname);
  reset(f1);
  write('Enter textout filename:');
  readln(outname);
  assign(f2,outname);
  rewrite(f2);
  j:=0; s:=' ';
```

```
while not eof(f1) do
begin
  while not eoln(f1) do
  begin
    inc(j);
    read(f1,ch); {邊讀文件邊排版}
    if j=maxlen+1 then
      case ch of
        ' ':begin writeln(f2,s); s:=''; j:=0; end;
        '.':begin writeln(f2,s); j:=1; s:=ch; end;
      else
        begin
          chl:=s[60];
          if s[59]=' ' then s[60]:=' ' else s[60]:='-';
          writeln(f2,s); j:=2; s:=chl+ch; {排完一行，寫入另一文件}
        end;
      end
    else s:=s+ch;
  end;
  readln(f1); {讀換行符}
end;
writeln(f2,s);
close(f1);
close(f2);
end.
```

對 TEXTCOPY.DAT 文本文件運行程序得到排版結果文件 TEXTCOPY.RES 內容如下：

```
Pavel was arrested.That dat Mother did not light the stove.
Evening came and a cold wind was blowing.There was a knock
at the window.Then another.Mother was used to such knocks,b-
ut this time she gave a little start of joy.Throwing a shawl
over her shoulders,she opened the door.
```

二、有類型文件

文本文件的元素均為字字符。若要在文件中存貯混合型數據，必須使用有類型文件。

1．有類型文件的定義

有類型文件中的元素可以是混合型的，並以二進制格式存貯，因此有類型文件（除了字符類型文件，因為它實質上是文本文件）不象文本文件那樣可以用編輯軟件等進行閱讀和處理。有類型文件的類型說明的格式為：

類型標識符=File of 基類型；

其中基類型可以是除了文件類型外的任何類型。例如：

```
FILE1=FILE OF INTEGER;
FILE2=FILE OF ARRAY[1..10] OF STRING;
FILE3=FILE OF SET OF CHAR;
FILE4=FILE OF REAL;
FILE5=FILE OF RECORD;
      NAME:STRING;
      COURSE:ARRAY[1..10] OF REAL;
      SUM:REAL;
      END;
```

等等，其中 FILE2，FILE3，FILE5 中的數組、集合、記錄等類型可以先說明再來定義文件變量。

例如：

```
VAR
F1:FILE;
F2,F3:FILE3;
F4:FILE5;
```

與前面所有類型說明和變量定義一樣，文件類型說明和變量定義也可以合併在一起，例如：

```
VAR
F1:FILE OF INTEGER;
F2,F3:FILE OF SET OF CHAR;
F4:FILE OF RECORD
```

```
NAME:STRING;  
COURSE:ARRAY[1..10] OF REAL;  
SUM:REAL;  
END;
```

Turbo Pascal 對有類型文件的訪問既可以順序方式也可以用隨機方式。為了能隨機訪問有類型文件，Turbo Pascal 提供如下幾個命令：

命令格式 1：

seek(f,n)

功能：移動當前指針到指定 f 文件的第 n 個分量，f 為非文本文件，n 為長整型

命令格式 2：

filepos(f)

功能：回送當前文件指針，當前文件指針在文件頭時，返回 0，函數值為長整型

命令格式 3：

filesize(f)

功能：回送文件長度，如文件空，則返回零，函數值為長整型

2·有類型文件的建立


有類型文件的建立只能通過程序的方式進行，其操作步驟與文本文件程序方式建立的步驟相仿，不同之處：

- (1) 有類型文件的定義與文本文件的定義不同；
- (2) 有類型文件可以利用 SEEK 命令指定指針隨機寫入。

3·有類型文件的訪問

有類型文件訪問的操作步驟與文本文件的程序訪問操作步驟相仿，區別之處：

- (1) 有類型文件的定義與文本文件的定義不同；
- (2) 有類型文件可以利用 SEEK 命令訪問文件記錄中的任一記錄與記錄中的任一元素。

 **例 14.6：建立幾個學生的姓名序、座號、六門課程成績總分的有類型文件。**

分析：為簡單起見，這裏假設已有一文本文件 FILEDATA.TXT，其內容如下：

```
10  
li hong  
1 89 67 56 98 76 45  
wang ming  
2 99 87 98 96 95 84  
zhang yi hong  
3 78 69 68 69 91 81  
chang hong  
4 81 93 82 93 75 76  
lin xing  
5 78 65 90 79 89 90  
luo ze  
6 96 85 76 68 69 91  
lin jin jin  
7 86 81 72 74 95 96  
wang zheng  
8 92 84 78 89 75 97  
mao ling  
9 84 86 92 86 69 89  
cheng yi  
10 86 94 81 94 86 87
```

第一個數 10 表示有 10 個學生，緊接著是第一個學生的姓名、座號、6 科成績，然後是第二個學生，等等。

從文本文件中讀出數據，求出各人的總分，建立有類型文件，設文件名為 filedata.fil，文件的類型為記錄 studreco，見下例程序。

```
program ex14_6;
const maxstu=100;
type {定義記錄類型}
  studreco=record
    name:string[18];
    numb:string[2];
    course:array [1..6] of integer;
    sum:integer;
  end;
  stud=array [1..maxstu] of studreco;
var
  n:integer;
  student:stud;


procedure in_data(var nn:integer; var data:stud); {從文本文件中讀取數據}
var i,j,s:integer;
  f:text;
begin
  assign(f,'filedata.txt');
  reset(f);
  readln(f,nn); {nn 為學生人數}
  for i:=1 to nn do
    with data[i] do
      begin
        readln(f,name);
        read(f,numb);
        s:=0;
        for j:=1 to 6 do
          begin
            read(f,course[j]); {讀各科成績}
            s:=s+course[j]; {求總分}
          end;
        sum:=s;
        readln(f);
      end;
    close(f);
  end;

procedure out_data(nn:integer; data:stud); {將數據寫入有類型文件中}
var i:integer;
  f:file of studreco;
begin
  assign(f,'filedata.fil');
  rewrite(f);
  for i:=1 to nn do
    write(f,data[i]);
  close(f);
end;

procedure write_data(nn:integer; data:stud); {讀出有類型文件中記錄數據，顯示輸出}
var i,j:integer;
  f:file of studreco;
begin
  assign(f,'filedata.fil');
  reset(f);
  for i:=1 to nn do
    read(f,data[i]);
  close(f);
  for i:=1 to nn do
    with data[i] do
      begin
        write(name:18);write(numb:4);
        for j:=1 to 6 do
          write(course[j]:4);
        writeln(sum:6);
      end;
    end;

begin {of main}
  in_data(n,student);
  out_data(n,student);
  write_data(n,student);
```

end.

 例 14.7：產生數 1~16 的平方、立方、四次方表存入有類型文件中，並用順序的方式訪問一遍，用隨機方式訪問文件中的某個指定的數及相應的平方、立方、四次方值。

分析：建立有類型文件文件名為 BIAO.FIL，文件的類型為實數型。

(1) 產生數 1~16 及其平方、立方、四次方值，寫入 BIAO.FIL，並順序讀出輸出；

(2) 用 SEEK 指針分別指向用戶指定的數 i 所在文件的位置，其位置數為 (i-1)×4（注意文件的第一個位置是 0），讀出其值及相應的平方、立方、四次方值輸出。

```
program ex14_7;
var
  f:file of real; {定義文件類型為實數型}
  i:integer;
  x,y,z:real;
begin {產生數據寫入文件中}
  assign(f,'biao.fil');
  rewrite(f);
  for i:=1 to 16 do
  begin
    x:=i*i; y:=x*i; z:=y*i;
    write(f,i,x,y,z);
  end;
  close(f);
  reset(f); {從文件中順序讀取全部數據}
  writeln('k':8,'k^2':10,'k^3':12,'k^4':14);
  while not eof(f) do
  begin
    read(f,i,x,y,z);
    writeln(i:8,x:10,y:12,z:14:2);
  end;
  write('Please input a number: '); readln(i);
  if (i-1)*4>filesize(f) then
    write('Out of range!')
  else
  begin
    seek(f,(i-1)*4); {指針指向 i 所在的記錄}
    read(f,i,x,y,z);
    writeln('line no ',i);
    writeln(i:8,x:10,y:12,z:14:2);
  end;
  close(f);
end.
```

程序運行結果如下：


k	k^2	k^3	k^4
1.00	1.00	1.00	1.00
2.00	4.00	8.00	16.00
3.00	9.00	27.00	81.00
4.00	16.00	64.00	256.00
5.00	25.00	125.00	625.00
6.00	36.00	216.00	1296.00
7.00	49.00	343.00	2401.00
8.00	64.00	512.00	4096.00
9.00	81.00	729.00	6561.00
10.00	100.00	1000.00	10000.00
11.00	121.00	1331.00	14641.00
12.00	144.00	1728.00	20736.00
13.00	169.00	2197.00	28561.00
14.00	196.00	2744.00	38416.00
15.00	225.00	3375.00	50625.00
16.00	256.00	4096.00	65536.00

Please input a number: 11
line no 11

11.00 121.00 1331.00 14641.00

另外，Turbo Pascal 還提供了第三種形式文件即無類型文件，無類型文件是低層 I/O 通道，如果不考慮有類型文件、文本文件等存在磁盤上字節序列的邏輯解釋，則數據的物理存儲只不過是一些字節序列。這樣它就與內存的物理單元一一對應。無類型文件用 1 2 8 個連續的字節做為一個記錄（或分量）進行輸入輸出操作，數據直接在磁盤文件和變量之間傳輸，省去了文件緩解區，因此比其它文件少占內存，主要用來直接訪問固定長元素的任意磁盤文件。無類型文件的具體操作在這裏就不一一介紹，請參看有關的書籍。

三、綜合例析

 例 14.8：建立城市飛機往返鄰接表。文本文件 CITY.DAT 內容如下：

第一行兩個數字 N 和 V；

N 代表可以被訪問的城市數，N 是正數 < 100；

V 代表下面要列出的直飛航線數，V 是正數 < 100；

接下來 N 行是一個個城市名，可乘飛機訪問這些城市；

接下來 V 行是每行有兩個城市，兩城市中間用空格隔開，表示這兩個城市具有直通航線。

如：CITY1 CITY2 表示乘飛機從 CITY1 到 CITY2 或從 CITY2 到 CITY1。

生成文件 CITY.RES，由 0、1 組成的 N×N 鄰接表。

鄰接表定義為：

$$LJ[i,j] = \begin{cases} 0 & i,j \text{ 兩城市無航線;} \\ 1 & i,j \text{ 兩城市直通航線;} \end{cases}$$

分析

（1）用從文本文件 city.dat 中讀入 N 個城市名存入一些數組 CT 中；

（2）讀入 V 行互通航班城市名，每讀一行，查找兩城市在 CT 中的位置 L、K，建立鄰接關係，lj[l,k]=1 和 lj[k,l]=1；

（3）將生成的鄰接表寫入文本文件 CITY.RES 中。

設 CITY.DAT 內容如下：

```
10 20
fuzhou
beijin
shanghai
wuhan
hongkong
tiangjin
shenyan
nanchan
chansa
guangzhou
fuzhou beijin
fuzhou shanghai
fuzhou guangzhou
beijin shanghai
guangzhou beijin
wuhan fuzhou
shanghai guangzhou
hongkong beijin
fuzhou hongkong
nanchan beijin
nanchan tiangjin
tiangjin beijin
chansa shanghai
guangzhou wuhan
chansa beijin
wuhan beijin
shenyan beijin
shenyan tiangjin
shenyan shanghai
shenyan guangzhou
```

程序：

```
program ex14_8;
var
  f,g:text;
  i,j,n,v,m,l,k,p:integer;
  lj:array [1..100,1..100] of integer;
  ct:array [1..100] of string;
```

```
    c1,c2,c:string;
begin
    assign(f,'city.dat');
    reset(f);
    assign(g,'city.res');
    rewrite(g);
    readln(f,n,v); writeln(g,n:4); l:=0; k:=0;
    fillchar(lj,sizeof(lj),0);
    for i:=1 to n do
    begin
        readln(f,ct[i]); writeln(g,i:4,' ',ct[i]);
    end;
    for i:=1 to v do
    begin
        readln(f,c); m:=pos(' ',c); p:=length(c);
        c1:=copy(c,1,m-1); c2:=copy(c,m+1,p-m);
        for j:=1 to n do
        begin
            if c1=ct[j] then l:=j;
            if c2=ct[j] then k:=j;
        end;
        lj[l,k]:=1; lj[k,l]:=1;
    end;
    for i:=1 to n do
    begin
        for j:=1 to n do write(g,lj[i,j]:3);
        writeln(g);
    end;
    close(f);
    close(g);
end.
```

得到 CITY.RES 文件內容如下：

```
10
1 fuzhou
2 beijin
3 shanghai
4 wuhan
5 hongkong
6 tiangjin
7 shenyan
8 nanchan
9 chansa
10 guangzhou
0 1 1 1 1 0 0 0 0 1
1 0 1 1 1 1 1 1 1 1
1 1 0 0 0 0 1 0 1 1
1 1 0 0 0 0 0 0 0 1
1 1 0 0 0 0 0 0 0 0
0 1 0 0 0 0 1 1 0 0
0 1 1 0 0 1 0 0 0 1
0 1 0 0 0 1 0 0 0 0
0 1 1 0 0 0 0 0 0 0
1 1 1 1 0 0 1 0 0 0
```

例 14.9：對例 14.6 的 FILEDATA.FIL 文件內容按總分的高低順序排序。

分析：文件的排序就是將文本文件的各分量按一定要求排列使文件有序，文件排序有內排序和外排序二種，內排序是指將文件各分量存入一個數組，再對數組排列，最後將該數組存入原來的文件。外排序不同於內排序，它不是將文件分量存入數組，而是對文件直接排序，內排序比外排序速度要快，但當文件很大時，無法調入內存，此時用外排序法較合適。本程序使用過程 SEEK，實現外排序。

程序：

```
program ex14_9;
const maxstu=100;
type
    stureco=record
        name:string[18];
        numb:string[2];
        course:array [1..6] of integer;
        sum:integer;
    end;
```

```
procedure file_sort; {外排序}
var
  i,j,nn:integer;
  f:file of studreco;
  a,b:studreco;
begin
  assign(f,'filedata.fil');
  reset(f);
  nn:=filesize(f);
  for i:=1 to nn-1 do
  begin
    for j:=i+1 to nn do
    begin
      seek(f,i-1); read(f,a);
      seek(f,j-1); read(f,b);
      if a.sum<b.sum then
      begin
        seek(f,i-1); write(f,b);
        seek(f,j-1); write(f,a);
      end;
    end;
  end;
  close(f);
end;

procedure write_data; {讀出排序後文件內容}
var
  i,j,nn:integer;
  f:file of studreco;
  data:studreco;
begin
  assign(f,'filedata.fil');
  reset(f);
  nn:=filesize(f);
  for i:=1 to nn do
  begin
    read(f,data);
    with data do
    begin
      write(name:18); write(numb:4);
      for j:=1 to 6 do
        write(course[j]:4);
      writeln(sum:6);
    end;
  end;
end;

begin {of main}
  file_sort;
  write_data;
end.
```

練習十四

1、編一程序，計算文本文件中行結束標誌的數目。

2、計算文本文件的行長度的平均值、最大值和最小值。

3、一文本文件 `FILE.DAT` 存放 `N` 個學生某學科成績，將成績轉換成直方圖存入 `FILE.RES` 文件中。

如 `FILE.DAT` 內容為：

```
5
78 90 87 73 84
```

得到直方圖文件 `FILE.RES` 內容為：

```
5
*****
*****
*****
*****
*****
```

4、銀行賬目文件含有每一開戶的賬目細節：開戶號、姓名、地址、收支平衡額。寫一程序，讀入每一開戶的賬目，生成銀行賬目文件。

5、通訊錄文件每個記錄內容為：姓名、住址、單位、郵編、電話，編一程序按姓名順序建立通訊錄文件，要求先建立文件，再對文件按姓名順序進行外排序。