Turbo Pascal大全

晓 境 编译

(5.0—5.5)

PASCAL

- 功能齐全的选项菜单
- 更加完善的嵌入单元
- 面向对象的程序设计

北京科海培训中心

TURBO PASCAL 大全

(5.0—5.5)

晓 琪 编译

科 海 培 训 中 心
一九九〇年九月

前　言

众所周知，一种计算机语言受用户喜爱的程度主要视其功效和结构。Turbo Pascal 语言之所以深受广大用户欢迎，主要是因为它功能齐全且鲜明地体现了当代程序设计语言的特点—易读、易写、易移植。Turbo Pascal 除保留了标准 Pascal 语言鲜明的结构外，还进一步发展了这种语言的处理能力。它以应用为对象，向程序设计者提供了功能齐全的各种程序设计手段。自推出 Turbo Pascal 5.0 后，最近，美国 Borland 公司又推出了 Turbo Pascal 5.5。这是目前关于 Pascal 的最新版本。除在 Turbo Pascal 5.0 的基础上进一步丰富了功能菜单外，Turbo Pascal 5.5 引入了目前最流行的程序设计思想—面向对象的程序设计。可以这么说，Turbo Pascal 5.5 是目前用于微机程序设计最有成效的开发环境和工具。利用它所提供的集成开发环境以及面面俱到的处理能力，您几乎可以用它做任何事情—编写系统软件，开发控制系统以及更多的应用。

本书适用于使用 Turbo Pascal 5.0 和 5.5 版本的程序设计者。作为一本大全，本书提供了有关 Turbo Pascal 最新颖、最全面的参考资料。它几乎涉及到了 Turbo Pascal 的每一个方面—集成开发环境，访问 DOS 和 BIOS 服务，与汇编语言的接口，工具箱的使用，等等。此外，本书向读者提供了大量实用且富有技巧的实例。仔细考察这些例子不但可使您迅速了解 Turbo Pascal 的基本内容，而且您可以从中学到极为宝贵的编程技巧。所有的例子都是上机调试过的，相信它们会为您带来帮助。

本书分为二十四章。第一章介绍了程序设计的基础，这是 Turbo Pascal 程序设计的初学者必不可少的一课。在这一章中，读者将迅速而全面地了解 Turbo Pascal 程序设计的概貌。第二章到第九章论述了 Turbo Pascal 程序设计系统的所有方面，从集成开发环境到指针和动态存储分配。第十章到第十八章对重要的程序设计专题—如 DOS 和 BIOS 功能、使用汇编语言和编写内存驻留程序等，提供了很有价值的深刻见解。第十九章到第二十二章讨论了 Turbo Pascal 的四个工具箱。您可以将其作为对各种工具箱的快速指南。第二十三章描述了如何使用新的集成调试器，以节省程序设计的时间。在第二十四章中，分绍了 Turbo Pascal 的最新增强特性—面向对象的程序设计。此外，在本书的最后，给出了对 Turbo Pascal 函数和过程的参考指南。

本书得以问世，首先要感谢科海培训中心华根娣主任和夏非彼编辑的热情支持和帮助。冶金部北京钢铁设计研究总院的祝光烈同志为本书做了大量的工作，本书的附录就是他的工作成果，他还为本书提供了大量的资料和素材。在此，向他表示由衷的感谢。

本书是利用业余时间编译而成的。由于时间仓促再加上本人的水平有限，因此，虽经过了努力修改，一定仍会存在不少的错误，欢迎读者批评指正，以使本书再版时能有一个水平上的提高。

1990 年 9 月
于北京

目　　录

第一章　Turbo Pascal 程序设计速成

如果是第一次使用 Turbo Pascal，那么本章最适合您阅读。在本章中，您将了解到 Turbo Pascal 系统的基本内容。同时，还可以编制和运行 Turbo Pascal 程序。我们希望不要过分关心本章所提出的每一个新内容，即使最简单的程序设计概念也要花费时间去理解。只需花些时间，在觉得 Turbo Pascal 系统是令人满意的情形下考察一下示范程序并尝试着动手编制自己的 Turbo Pascal 程序就可以了。

§1.1 一个简单的 Turbo Pascal 程序

开始学习编程的最好方法是编制自己的程序。为了启动 Turbo Pascal，要保证已注册进入指定的驱动器中并进入驻留 Turbo.exe 的目录内。在 DOS 提示符下，键入 Turbo 并按 ENTER 键。

在屏幕上，您将看到 Turbo Pascal 集成开发环境。屏幕的顶端是主菜单，它将使您能够进入并利用所有 Turbo Pascal 的特性。在主菜单下是编辑窗口。在编辑窗口中，可以键入程序。在编辑窗口下面是用于调试程序的监视窗口。

为了编写第一个程序，按 F10 键以激活主菜单。然后按 E(用于编辑)。现在，光标将出现在编辑窗口中，请准备好键入下面的 Turbo Pascal 程序，该程序将在屏幕上显示一行文字。

```
Program Prog1;
Begin
WriteLn('This is my first program. );
ReadLn;
End.
```

如果键入有错，使用数字键盘区中的箭头键将光标定位在错误处，按 DEL 消除错误，再键入正确字母。

一旦完整地键入了这个程序，再按 F10 激活主菜单，按 R 以运行程序。Turbo Pascal 将执行刚刚编写的程序，监示器显示这样的信息：

This is my first program.

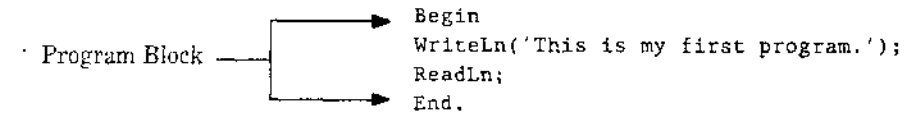
这个程序虽然小，却包含了所有 Turbo Pascal 程序公共的元素。如图 1-1 所示，它有一个程序头----Program Prog1，还有一个以 Begin 开始以 End 结束的程序块(Program Block)。

一个 Turbo Pascal 程序总是在主程序的第一个 Begin 语句处开始执行，直到到达最后的 End 语句为止(当一个程序遇到 Halt 命令或一个致命错误时，它也符停止，但这些都是例外)。

上例程序的程序块仅包含两个语句：

```
WriteLn('This is my first program.');
ReadLn;
```

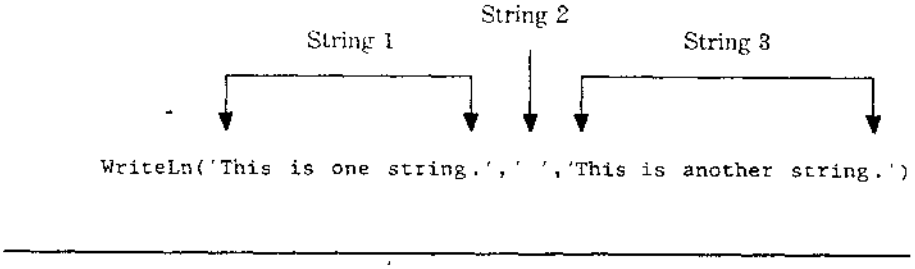
WriteLn 语句显示字符串，ReadLn 语句使计算机等待键入 ENTER。如果是在集成开



一个短的程序块

发环境(IDE)中运行一个程序，则可以使用 **ReadLn** 语句在屏幕回到编辑器屏幕之前停止一个程序的运行。在 IDE 中，可以通过按 **ALT-F5** 看到输出屏幕。

WriteLn 是一个标准 **Turbo Pascal** 过程，它将数字和字符串显示到屏幕上，把它们在打印机上打印出来或将它们写到一个磁盘文件中去。它还将两个专门符号--回车和换行加到行末。这两个专门符号通常在程序设计中简写为 **CR/LF(ASCII** 码为 13 和 10)加以引用，它指示出到这一行正文的末尾和任何增加的正文将在下一行开始。在 **WriteLn** 语句的括号内是将要被打印的内容。在本例的情形下，**WriteLn** 过程仅打印一行，事实上，它能够一次打印多于一个的内容，图 1--2 给出了一个打印三个不同串的 **WriteLn** 语句的例。



一个带有多个串的 **WriteLn** 语句

一个串是单引号括起来的字符任意组合。在图 1--2 所示的例子中，用逗号将一个串与其它串隔开。注意到第二个串由两个空格字符组成。这样，就在第一个串和第三个串之间建立了一个空隙。当这个语句被执行后，结果如下：

This is one string. This is another string.

同 **WriteLn** 一样，**Write** 也显示串和数字。但 **Write** 没有将 **CR/LF** 符号加到行末。当您使用 **Write** 时，光标仍然停留在信息写入的同一行上，而 **WriteLn** 将光标移到下一行的开始处。

1.2 在程序中应用变量

仅打印出文字信息的程序并无多大意义。真正有用的是，程序必须处理数据并需要使用变量，数据和变量被放在计算机的内存中。

为了定义一个变量，必须给出其名字和类型。几乎可以为变量任意起一个名字。但最好选择一个能够描述变量存放信息的名字。例如，可以为一个包含了顾客名的变量起名为 **CustomerName** 并定义如下：

```
Var
    CustomerName: String[50];
```

变量名 CustomerName 也称为变量标识符，因为它通过名字标识了在内存中的存储单元。String[50]将变量说明为一个串并指明该串的长度不得超过 50 个字符。

Var 是一个指示变量声明开始的 Turbo Pascal 保留字，在 Turbo Pascal 中还有许多其它的保留字，如 Integer，Begin，End 等等。对 Turbo Pascal 来讲，保留字是重要的，因此，不能对其进行再定义。下面的例子试图使用保留字作为变量标识符，这是非法的。

```
Var
    Begin: Integer;
    Real: String[50];
```

Integer 和 Real 变量用来存放数字；Char 变量用来存放单个字符；String 变量用来存放字符串；Boolean 变量包含了 True/false 指示符。虽然这些类型用于不同的目的，但它们的变量却具有一个公共的特性----即在程序中通过使用赋值语句改变它们的值。

一个赋值语句将变量设置成特定的值。例如，语句

```
CustomerName:='John Doe';
```

将字符串'John Doe'接收下来并存储在串变量 CustomerName 中。注意!赋值语句使用我们所熟知的赋值运算符:=。

§1.3 变量和输入

赋值语句是设置变量值的一个方法，ReadLn 是另外一个设置变量值的方法。与赋值语句不同的是，ReadLn 从程序之外获得值----如使用程序的人或磁盘文件。当程序遇到一个 ReadLn 语句时，它停下来等待用户键人数据并按 ENTER。然后，ReadLn 获得输人并将其赋给 ReadLn 语句中的变量。例如，Turbo Pascal 输人语句 ReadLn(CustomerName)等待用户键人一个字符串，接收一个串并将其存放在变量 CustomerName 中。

下面的例子向我们示范了 ReadLn 是如何获得输人并将其存放在变量之中的。

现在，让我们回到 Turbo Pascal 编辑器中并将上面的程序输人进去。你已看到，这个程序比起第一个程序来稍稍有些复杂。首先，程序包括了下面的声明:

```
Program Prog2;

Uses CRT;

Var
  i : Integer;
  s : String[20];

Begin
ClrScr;

Write('Enter a number: ');
ReadLn(i);
WriteLn('Your number is ',i);

Write('Enter a string: ');
ReadLn(s);
WriteLn('Your string is ',s);

ReadLn;
End.
```

Uses CRT;

CRT 是 Turbo Pascal 的一个标准单元。单元包括数据声明、过程和设计用于专门应用的函数。例如，CRT 单元中的例行程序主要应用于监视器显示屏幕。如果要使用一个单元中的过程和函数，就必须在程序的开头包含 Uses 语句。

在输入完程序之后，象启动第一个程序那样启动它(在 Run 菜单中选择 Run)。当程序启动后，(来自 CRT 单元中的)Clr Scr 命令清屏，程序在监示器上显示 Enter a number:键入数值 9 并按 ENTER，程序显示 Your number is 9 并在屏幕上跳过一行，显示 Enter a string:。键入 ABC 并按 ENTER。程序显示 Your string is ABC。程序值用了 ReadLn 语句从用户那里获得一个数和一个串。

程序 Prog2 使用了两个变量:整型(integer)变量 i 和串型(string)变量 S。整型数是无小数部分的数，值可在-32768 至 32767 之间变化。串型变量 S 被定义为 String[20]，这意味着它可以存放多达 20 个字符。(一个串所能存放的最大字符数是 255)。

上边给出的示例程序使用了一个从用户那里获得输入的通用方法。首先，程序通过要求一个数来提示用户输入。通过使用 Write 过程将提示 Enter a number 显示出来。与过程 WriteLn 相对，Write 过程把光标直接放在提示信息之后。它告诉用户，程序正在等待输入。

下一个语句 ReadLn(i);等待用户键入一个有效整数并按 ENTER，程序将用户键入的内容赋给整型变量 i，最后一个语句通过显示一条信息和 i 的内容证实了用户的输入。

如果 ReadLn 语句查出在输入中的错误，它将通知您并停止程序的运行。例如，重新运行 Prog2，当它要求一个数时，键入 ABC 并按 ENTER。Turbo Pascal 将查出一个 Input/Output 错误并显示这样的信息:

Runtime error 106 at 0000: 0041

106 错误指示出一个无效数使格式。简单说来，Turbo Pascal 期待一个数而它却得到另外的内容----并非有数整数的 ABC。值 0000:0041 是程序中的存储单元，错误是在该单元中发生的。如果是在 IDE 中运行程序，Turbo Pascal 将自动地在源代码中使错误定位并将程序的该部分带进编辑器。

程序 Prog2 还示范了无参数 WriteLn 语句的用法。当执行它时，这个语句仅向计算机监视器输出一个 CR/LF，使光标放到下一行的第一个位置。

§ 1.4 简单的 Turbo Pascal 运算

下面的示例程序示范了在 Turbo Pascal 程序中如何使用运算，并介绍了另外一个数据类型 Real。同 Integer 型变量相同的是，Real 变量也是数;不同之处在于，Real 变量可以带有小数部分。它们可以比 Integer 型变量的值大许多:一个 Integer 型变量的最大值是 32767，

```
Program Prog3;
Uses CRT;
Var
  Number1,
  Number2,
  AddResult,
  SubResult,
  MultResult,
  DivResult    : Real;
```

```
Begin
ClrScr;

Write('Enter a number: ');
ReadLn(number1);
Write('Enter another number: ');
ReadLn(number2);

AddResult  := Number1 + Number2;
SubResult  := Number1 - Number2;
MultResult := Number1 * Number2;
DivResult  := Number1 / Number2;

WriteLn;
WriteLn('Number1 + Number2 = ',AddResult);
WriteLn('Number1 - Number2 = ',SubResult);
WriteLn('Number1 * Number2 = ',MultResult);
WriteLn('Number1 / Number2 = ',DivResult);

WriteLn;

WriteLn('Number1 + Number2 = ',AddResult:10:3);
WriteLn('Number1 - Number2 = ',SubResult:10:3);
WriteLn('Number1 * Number2 = ',MultResult:10:3);
WriteLn('Number1 / Number2 = ',DivResult:10:3);

WriteLn;
Write('Press ENTER...');
ReadLn;
End.
```

而一个 **Real** 型变量的最大值则是 **1.0E38**，即 **1** 后随 **38** 个零。

程序 **Prog3** 要求用户输入两个数赋给 **Real** 型变量 **Number1** 和 **Number2**，然后在四种运算中使用它们，这四种运算是:加法、减法、乘法和除法。在运算之后，**Prog3** 以两种方式:科学表示法和十进制数表示法输出了结果。

科学表示法仅用于 **Real** 型变量，是表达大数的简短方式。例如，下列运算的结果:

5342168903247*24729234798734

以科学表示法表示将是 **1.3210774914E+26**。该数的第一个部分包含了有意义的数字，第二部分是第一部分 **10** 的幂。换句话说，数 **1.3210774914E+26** 可以表示成 **1.32107 74914** 乘以 **10** 的 **26** 次方。

在 **Turbo Pascal** 中，科学表示法是用于表达 **Real** 型变量值的缺省表示法。然而，也可以用十进制数表示格式输出 **Real** 型的值。例如，在语句

WriteLn('Number1+Number2=',Addresult:10:3);

中，变量 **Addresult** 后随格式说明 **10:3**，这告诉 **Turbo Pascal** 在一个 **10** 位宽并允许 **3** 位小数的地方以右齐格式打印 **Real** 型变量。如果结果数为 **5**，则该数被以图 **1--3** 的格式打印出来。

如果被打印的数需要多于分配的 **10** 个位，程序将尽可能以满足需要的位数把整个的数打印出来。

§ 1.5 带有循环的语句

循环是重复一条或一组语句的机构。**Turbo Pascal** 提供了几种建立循环的方式。下面的示例程序示范了两种循环方式:**For-Do** 循环和 **REPEAT-Until** 循环。

Print position	1	2	3	4	5	6	7	8	9	10
Output						5	.	0	0	0

图 1--3　格式化的数值输出

```
Program Prog4
Uses CRT;
Var
  NumberArray : Array [1..5] Of Integer;
  Average : Real;
  i : Integer;

Begin
ClrScr;

(*********************)
(* The For-Do Loop *)
(*********************)
For i := 1 To 5 Do
  Begin
  Write('Enter a number: ');
  ReadLn(NumberArray[i]);
  End;

Average := 0;
i := 1;

(*************************)
(* The Repeat-Until Loop *)
(*************************)

  Repeat
  Average := Average + NumberArray[i];
  i := i + 1;
  Until i > 5;

Average := Average / 5;
WriteLn('The average is: ',Average:0:2);

ReadLn;
End.
```

写一个循环需要三个元素:起始点、终止点和一个用作计数器的 **Integer** 型变量。在 **For-Do** 循环定义

For i:=1 To 5 Do

中，**i** 是计数器，**1** 是起始点，**5** 是终止点，每次重复循环时，**i** 的值增 **1**，在第五次通过循环后，**i** 的值为 **6**。因为 **6** 大于在 **For-Do** 循环定义中说明的终止点，循环终止。程序继续执行紧随循环块后的第一个语句。

在上例程序中还示范了第二种循环类型----**Repeat-Until** 循环。与 **Foe-Do** 循环相比，

Repeat-Until 循环需要做更多的工作；您，而不是 Turbo Pascal，必须初始化计数器和增量的值并测试终止循环的条件。由于这些工作，使 Repeat-Until 循环具有了较多的优点。在编写一个循环之前，您不必知道它将执行多少次。Repeat-Until 循环一直重复执行直到在 Until 行中所说明的条件满足为止。您还可以对计数器给予增量。Repeat-Until 的另一个优点是可以同时测试多个条件，如下例所示：

```
Repeat
i:=i+1;
j:=j+1;
Until (i>100) Or (j=50);
```

如果在 Until 语句中的任一个测试为真，则程序退出 Repeat-Until 循环。循环结构是计算机程序设计所有方面的基础。随着程序设计实践的增多，您将发现它们众多的使用。

§1.6 使用磁盘文件

实际上，您更需要编写程序来存取和检索磁盘文件上的数据，如下面程序所示范的那样。Turbo Pascal 使得磁盘文件的使用变得更为容易。

```
Program Prog5;
Uses CRT;
Var
  i,j : Integer;
  f : Text;
  r : Real;

Begin
ClrScr;

Assign(f,'SQUARES.DAT');
Rewrite(f);

For i := 1 To 20 Do
  WriteLn(f,Sqr(i):10);

Reset(f);
For i := 1 To 20 Do
  Begin
  ReadLn(f,j);
  WriteLn(i:4,' squared is ',j:4);
  End;

Close(f);

WriteLn;
Write('Press ENTER...');
ReadLn;
End.
```

Prog5 建立了一个正文文件并将前 20 个非负整数的平方写到文件中去。然后，程序重新读这个文件并将值输出到屏幕上。

Prog5 引用了 Turbo Pascal 保留字 Text，这是一个磁盘文件类型，虽然一个正文文件也能存放数值，但它主要用于存放单词和句子。Prog5 声明了文件标识符 f 是 Text 类型的。我们将在所熟悉的 ReadLn 和 WriteLn 语句中使用这个文件标识符。以直接向磁盘而不是屏幕输入和输出。