中国科学院希望高级电脑技术公司

TURBO PASCAL5.50 技术参考大全

- 集成环境：热键、菜单、联机帮助的编程、调试、运行一体化环境。
- 基本概念：程序、数据类型、算术和逻辑运算、控制结构、指针和动态分配等。
- 一般编程：字符串、递归和文件、合并、排序和查找。
- 高级编程：与DOS、BIOS接口，与汇编接口、文本显示、图形显示、中断、RS—232通讯和内存驻留，程序优化，面向对象(OOP)编程调试、Debugger调试技巧等。
- 参考大全：最新数据库工具箱、图形工具箱、编辑工具箱、数值处理工具箱、过程和函数参考等。

陈世录　张华凯
谭军安　付单兵
盘　虹　许雄斌
编　译

Turbo Pascal 技术参考大全

——5.0 5.5 版初级 中级 高级编程 调试技巧

陈世录 张华凯 谭军安 付单兵
盘 虹 许雄斌 等译

- 集成环境 热键、菜单、联机帮助的编程、调试、运行一体化环境
- 基本概念：程序结构 数据类型 算术和逻辑运算 控制结构
  指针和动态分配内存 文件操作
- 一般编程：字符串、递归和文件 合并、排序和查找
- 高级编程：与*DOS*和*BIOS*接口 与汇编接口 文本显示 图形显示
  中断、*RS-232*通讯和内存驻留 程序的优化
  缓冲字符串*I/O* 大串处理 快速算术运算
  文件加密
- 高级编程：面向对象(*OOP*)编程调试 内部符号调试
  Turbo*Pascal*的*Turbo Debugger*调试技巧
- 参考大全：最新数据库工具箱 图形工具箱
  编辑工具箱 数值处理工具箱过程和函数参考
- 参考大全：标准过程和函数参考 错误信息一览表
  PC ASCII码 PC键盘 保留字

中国科学院希望高级电脑技术公司
一九九一年元月

目　录

第一章　编程快速入门

- 一个简单的 Turbo Pascal 程序
- 在程序中加入变量
- 变量和输入
- 简单的 Turbo Pascal 算术运算
- 使用循环重复语句
- 使用磁盘文件

这一章是专门为初学 Turbo Pascal 的用户编写的。通过这一章的学习，可以了解 Turbo Pascal 的基本要点，学习编写和运行用户的第一个程序。不必太注重于这一章所给出的全部内容的理解，因为既使深刻理解简单的程序概念也需要时间。本章要旨是只花费一点时间熟悉系统；试验一下范例程序，用户亲自动手做做实验。

§ 1.1 一个简单的 Turbo Pascal 程序

一个最好的开始是写你自己的第一段程序。启动 Turbo Pascal 时，首先要在当前驱动器的当前目录中有 TURBO.EXE 文件。在 DOS 提示符下，键入 Turbo，并回车。

在屏幕上，可以看到 Turbo Pascal 的集成开发环境。屏幕的顶端是主菜单，通过它可以使用全部的 Turbo Pascal 功能。在主菜单下是编辑窗口(Edit Window)，用于编制用户程序。编辑窗口下是监视窗口(Watch Window)，用于调试程序。要编制第一个程序，按下 F10 键激活主菜单，然后按 E(Edit)键，此后光标位于编辑窗口，此时可敲入下面的 Turbo Pascal 程序。这段程序用于在屏幕上显示一行文本：

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

如果在输入过程中按错了键，使用数字键盘上的箭头键，移动光标到错误字符，按 DEL 键删除，然后敲入正确的字符。

输完程序后，按 F10 键激活主菜单，按 R 选择 Run 菜单，再次按 R 运行程序。Turbo Pascal 将运行刚才编写的程序，在监视器上显示：

```
This is my first program.
```

按回车返回到集成开发环境。

虽然这段程序很小，但包含了所有 Turbo Pascal 程序都有的元素。有一个程序头 Program Prog1;标识程序。还有一个以 Begin 开始，以 End 结束程序块，如图 1.1 所示。

Turbo Pascal 程序总是从主程序块的第一个 Begin 语句开始执行，直到最后一个 End 语句(程序也可能在遇到 Halt 命令和致命错误时停止，但这些都是例外情况)。

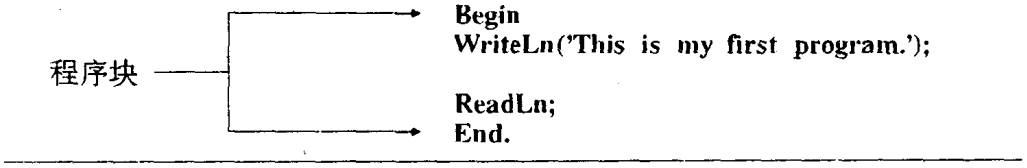

图 1.1 一个小程序块

以上示例中给出的程序块仅包含两条语句：

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

WritLn 语句显示字符串，ReadLn 语句使计算机等待直到按下回车键。如果在集成开发环境下运行程序，可以使用 ReadLn 语句，在屏幕切换回编辑屏幕(Edit Screen)之前让程序停顿。在 IDE (集成开发环境)中可随时按下 ALT-F5 键来观察输出屏幕(Output Screen)。

WriteLn 是 Turbo Pascal 在屏幕显示或在打印机上打印字符串，或把它们写到一个磁盘文件中去的标准过程。它输出时添加两个特殊字符，即回车和换行(ASCII 码 13 和 10)，结束一行。这些特殊字符在编程时通常简写为 CR/LF，它标志着一个文本行结束，所有后加的文本，都从下一行开始。

WriteLn 语句的园括号中的项是要打印的内容。虽然在这种情况下，WriteLn 过程只能打印一行，但是它可以一次打出多项内容。图 1.2 是一个用 WriteLn 语句打印三个独立的字符串的例子。

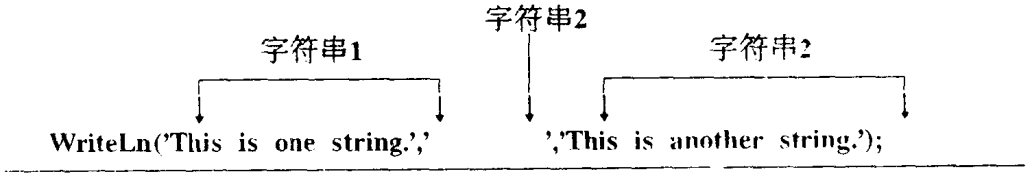

图 1.2 一个多字符串的输出语句

一个字符串是一组包含在单引号中的字符的组合。示例的 WriteLn 过程中用逗号分隔字符串。注意，第二个字符串产生两个空格字符，这个语句执行后会得到下面的结果：

```
This is one string.  This is another string.
```

与 WriteLn 类似，Write 过程也显示字符串或数字；但是 Write 不在该行后附加 CR /LF 符。在使用 Write 时光标将停留在显示的那一行，而 WriteLn 则将光标移到下一行的开始。

§ 1.2 在程序中加入变量

输出信息很少的程序没有什么大用处。一个真正有用的程序必须处理数据，如数字或字符串。而数据需要使用变量，并存放在计算机中存贮数值的内存里。

定义一个变量必须给出变量名和类型，可以变量起一个喜欢的任意名字，但最好起一

个能描述所包含信息的名字。例如给一个存放顾客名的变量起名叫 CustomerName，其定义如下：

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

变量名 CustomerName 也称为变量标识符，因为通过名字标识内存中变量的存放地址。String[50]说明是一个字符串型的变量，而且限定其长度不能超过 50 个字符。

VAR 是 Turbo Pascal 的保留字，表明变量说明的开始。在 Turbo Pascal 中还有 Integer，Begin，End 等许多保留字(附录 D 有一张完整的 Turbo Pascal 保留字表)。保留字在 Turbo Pascal 中很重要，因而不可重定义。下面是一个企图将保留字当作变量标识符使用的例子：

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

整型(Integer)和实型(Real)变量可以存放数字，字符型(Char)变量可以存放单个字符；字符串型(String)变量可以存放一组字符；布尔型(Boolean)变量存放真/假标志。虽然它们是为不同目的设计的变量类型，但有一个共同的特点，就是，在程序中使用赋值语句可改变它们的值。赋值语句置变量为一个特定的值，例如：

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

取一组字符“John Doe”，把它们存到字符串变量 CustomerName 中。注意赋值语句使用的“:=”操作符，称为赋值操作符。

§1.3 变量和输入

赋值语句是给变量赋值的一种方法；ReadLn 过程是另一种方法。但是与赋值语句不同的是 ReadLn 从程序外界得到值，如键盘或者磁盘文件。一个程序遇到 ReadLn 语句就会停下来等待，直到使用者键入数据并且按下回车，随后 ReadLn 取得输入，把它赋给 ReadLn 语句中的变量。例如一条 Pascal 输入语句 ReadLn(CustomerName)，等待用户敲入一个字符串，接收字符串后，把它存到 CustomerName 变量中。

下面一个简单的程序，说明 ReadLn 是如何取得输入，并且把它存入变量的：

```
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.
```

现在返回 Turbo 编辑器，键入上面的程序。注意这段程序比第一段程序稍微复杂一点，这是因为程序中包含了下面的声明：

```
Uses CRT;
```

CRT 是 Turbo Pascal 的标准单元。单元中包含数据声明，为特殊目的设计的过程和函数。在 CRT 中的例程主要应用于视屏显示。如需使用单元中在过程或函数，必须在程序开始处使用 Uses 语句声明。

敲入程序后，就可象刚才启动第一个程序一样启动它（选择 Run 菜单中在 Run 选择项）。程序启动后，CRT 单元里的 ClrScr 命令清屏，程序显示“Enter a number:”到屏幕上。键入 9 然后回车，程序显示“Your number is 9”，跳过一行显示“Enter a string:”；键入 ABC 并回车，程序现在显示“Your string is ABC.”。这一程序通过 ReadLn 语句从用户获得一个数字和一个字符串。

Prog2 使用了两个变量：整型 i，字符串型 s。整型值是没有小数的数字，范围限制在 -32,768 到 32,767。字符串变量用 String[20]定义，表示能接收 20 个字符(一个字符串的最大字符数目限制在 255 内)。

在作为例子的程序中，仅给出了通用、或者说很原始的方法接收用户数据。首先程序向要用户提示输入一个数。提示的“Enter a number:”用 Write 过程显示，而不用 WriteLn，这是因为 Write 使光标停在提示信息后面，这一提示告诉用户，程序在等待输入。

下一条语句 ReadLn(i)，等待用户敲入一个有效的整型数和回车。程序把用户敲入的值赋给整型变量 i。最后一条语句显示一条信息和 i 的值用以证实输入。

如果 ReadLn 语句测试到输入错误，会通知用户，同时停止程序。例如，重新运行 Prog2，但当请求数时，敲入 ABC 然后回车。Turbo Pascal 检测到一个输入/输出错误，显示信息：

```
Runtime error 106 at 0000:0041
```

106 号错误表示一个数字格式无效，换句话就是 Turbo Pascal 希望得到一个数字值，但是却得到另外一些东西(ABC)，并不是一个有效的整型数。值 0000:0041 是程序中错误发生的位置。如果你是在 IDE 环境，Turbo Pascal 会自动定位错误在源码中的位置，并且把这一段程序送到编辑器。

Prog2 演示了不带参数的 WriteLn 语句的使用。执行这条语句时，只是简单的写一个 CR/LF 到计算机的监示器，作用是使光标移到下一行的第一个位置。

§1.4 简单的 Turbo Pascal 算术运算

下面的例程演示在 Turbo Pascal 中是如何进行算术运算的，同时介绍另一种数据类型 Real(实型)。与整型变量一样，Real 变量也是数字，与整型不同的是实型可以存小数位。它的值的范围也远比整型大，整型数的最大值是 32767，而一个实型数的最大值是 1.0E38，也就是 1 后面有 38 个零。

```
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.
```

Prog3 请求用户输入两个数字，分配给两个实型变量 Number1 和 Number2，然后用于四则算术运算：加、减、乘、除。计算完成之后，Prog3 用科学记数法和十进制两种不同的方式显示结果。

科学记数法仅用于实型变量，是一种表示大数的简便方法。例如以下计算的结果：

5342168903247 * 24729234798734

用科学记数法记为 1.3210774914E+26；第一部分数字(1.3210774914)是存放数字；第二部分(E+26)是第一部分要乘上的 10 的幂。换句话，1.3210774914E+26 表示 1.3210774914 乘以 10 的 26 次方。

科学记数法是 Turbo Pascal 实型变量的缺省格式。也可以用十进制格式写实型数。例如：

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

变量 AddResult 后跟格式说明:10:3，告诉 Turbo Pascal 显示的实型变量限制在 10 个字符位置以内，允许有三位小数。如果结果等于 5，其显示如图 1.3 所示。

打印位置	1	2	3	4	5	6	7	8	9	10
输出						5	.	0	0	0

图 1-3 格式化的数值输出

如果数字显示需要多于允许的 10 个字符，程序将跟据实际需要显示全部的数字。

§ 1.5 使用循环、重复语句

循环是重复一条或一组语句的一种机制。Turbo Pascal 提供了几种产生循环的途径。在下面的程序中演示了两种循环：For-Do 循环和 Repeat-Until 循环。

```
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要素：一个起始点，一个终点，还有一个用于计数的整型变量。在 For-Do 循环定义

```
For i := 1 To 5 Do
```

中，i 是计数变量，1 是起点，5 是终点。循环开始时，程序置 i 为 1。每次循环重复时 i 的值加 1，5 次循环后 i 的值等于 6，大于 Fro-Do 循环终点 5，循环结束，程序执行 For-Do 循环块后的第一条语句。

例程演示了称为 Repeat-Until 循环的第二种循环。与 For-Do 循环相比较，Repeat-Until 循环需要作的工作多一点，需程序员而不是 Turbo Pascal 初始化计数变量的值，增加它的值，并且测试结束循环的值。但就是这些工作，使 Repeat-Until 循环自具其优点。首先在写循环语句之前不必知道循环要执行多少次。Repeat-Until 循环重复直到满足在 Until 语句行说明的条件。可以按需增加计数变量。还有一个优点就是可以同时测试多个条件，如下面的例子：

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

只要 Until 语句中的条件有一个测试后为真，程序就跳出 Repeat-Until 循环。循环结构是所有程序设计的基础。随着编程技术的提高，会发现它的更多用处。

§1.6 使用磁盘文件

编写程序最终总会使用磁盘文件存放和提供数据。Turbo Pascal 使磁盘文件的使用简单化，如下面的程序所示：

```
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 建立一个文件，然后顺序填入 20 个正整数。该程序重新读出这一文件，并把值写到屏幕。

Prog5 使用了 Turbo Pascal 的保留字 Text，一种主要存放单词或句子的磁盘文件。Text(文本)文件也可以存放数字。Prog5 声明文件标识符 f 为 Text 型，这个文件标识符将用在已经熟悉的 ReadLn 和 WriteLn 语句中，用于直接从磁盘文件而不是从屏幕中输入和输出。

在使用文件型变量前，先要用一个磁盘文件名给它赋值。可使用 Assign 命令：

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

Assign 命令连接文件变量 f 和物理文件 SQUARES.DAT。Rewrite 语句使文件准备接收数据。如果物理文件 SQUARES.DAT 不存在，Rewrite 命令就创建它。如果文件存在，Rewrite 就销毁文件内容。一个文件被声明重写之后就随时可以接收输出。在 Prog5 中输出到文件 f 由如下语句描述：

```
WriteLn(f,Sqr(i):10);
```

这和前面用过的 WriteLn 过程一样，但第一个参数是文件变量 f，指示这条语句输出的所有内容写到文件 f 中去。

Rewrite 让文件准备写，Reset 让文件准备读。如果用 Reset 声明文件，ReadLn 语句就可以从文件中读取数据，如下所示：

```
ReadLn(f,j);
```

这条语句从 Text 文件 f 中读出一个整型数并存入 j。Turbo Pascal 支持多种文件类型，文本文件是初学者最易使用的一种文件，因为它可以同键盘和监视器一样进行输入和输出。

使用文件的最后要求是关闭文件。关闭文件要做两件事。首先如果一个文件用作输出，缓冲区的数据都将送入磁盘。如果一个程序结束但没有关闭文件，就可能将尚未写入磁盘的缓冲区中的数据丢失。其次，关闭文件将释放一个打开的文件句柄，启动系统时，DOS 服务程序填写打开文件句柄的数量，其缺省值是 8。通过在 CONFIG.SYS 中加入一行

```
FILES=20
```

可改变这个值，使打开文件句柄达到 20。Turbo Pascal 总是定义前五个文件是它的标准

输入/输出设备。也就是说缺省时程序只能打开三个文件。每个打开文件占用一个句柄，只能同时打开三个文件。如果程序中使用多个文件，要注意及时关闭不用的文件，以免超过打开文件的限制。

读完这本书后，读者会加深对这一章介绍的程序设计的概念的理解。这同许多高深的学术论文介绍的知识一样有效。**Turbo Pascal** 提供了许多强有力的功能，需要一定的时间才能完全掌握。阅读、练习、培养兴趣，这样在不知不觉中就能够成为一名合格的程序设计师。

第二章　Turbo Pascal 编程系统

- 启动
- File 菜单
- Run 菜单
- Compile 菜单
- Options 菜单
- Debug 菜单
- Break/Watch 菜单

Turbo Pascal 大受欢迎的原因之一是它的集成开发环境(IDE)。在 Turbo Pascal 的 IDE 下，可以编辑、编译、运行和调试程序，而不必返回 DOS。Borland 首先提出 IDE 的概念，经过精炼加工推出了目前最有效的编程系统。

在 Turbo Pascal 5.5 中，IDE 提供了更多的选择项，其中包括集成的调试功能。一个受欢迎的增加是监视窗口，可以在程序运行时观察变量的值。这一章详细介绍了 Turbo Pascal 5.5 的用户接口以及如何使用各项功能。

§2.1 启动

在 DOS 提示符下键入 TURBO 并回车，就进入 Turbo Pascal 的集成开发环境，如图 2-1 所示。屏幕顶行是主菜单，有 7 个选择项，File，Edit，Run，Compile，Options，Debug 和 Break/Watch。进行选择时使用 F10 和箭头键使主菜单某一选择项成为高亮度区，然后按回车。也可以敲某选择的第一个字母(如按 F 代表 File)。

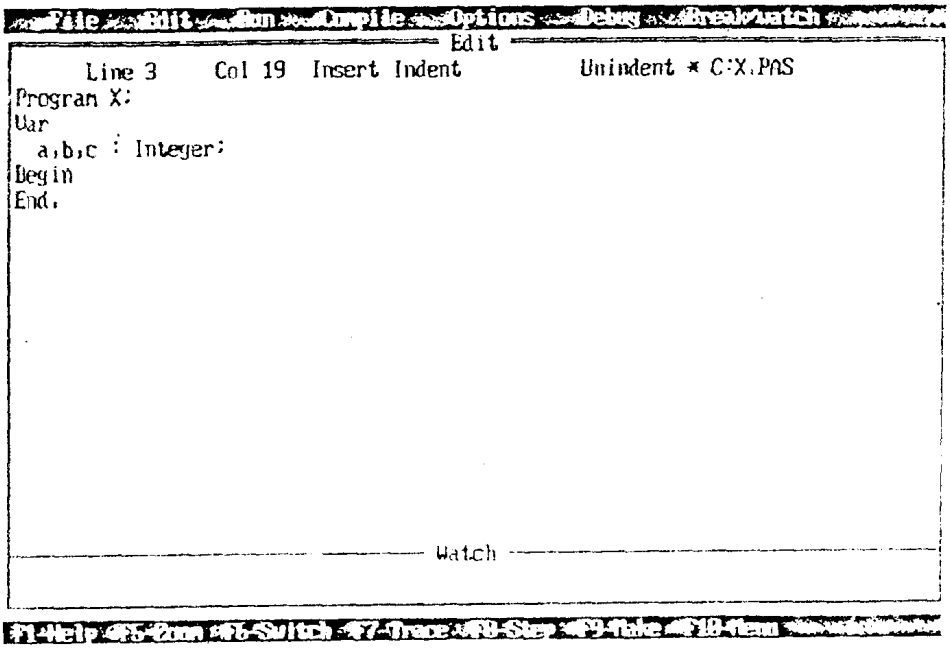

图 2-1　Turbo Pascal 5.5 主菜单