第一章　Turbo Pascal 初阶

1.1　关于 Turbo Pascal

Pascal 是一种计算机通用的高级程序设计语言。它由瑞士 Niklaus Wirth 教授于六十年代末设计并创立。取名 Pascal 是为了纪念十七世纪法国著名哲学家和数学家 Blaise Pascal(1623—1662)。第一个 Pascal 编译程序于 1970 年开始运行。1971 年,Wirth 教授在瑞士的"ETH"杂志上正式发表了 Pascal 程序设计语言的初始报告。1974 年,他和 K. Jensen 联名发表了著名的修改报告和用户手册。这本书描述了标准 Pascal,并提供一些用于说明 Pascal 特性的例子,被 Pascal 的系统实现者和用户们视作基本的指南和权威性著作。

Wirth 教授所设计的 Pascal,是一种优良的程序设计教学语言,整个语言紧凑整齐,概念简洁,数据结构和控制结构比较丰富,程序可读性好,查错能力强,有利于培养学生严谨、清晰的程序设计风格和良好的习惯,并使他们从中领会和理解精细的程序设计技巧。尤其引人注目的是其中引入了结构化程序设计的思想。

在 Pascal 问世以来的二十余年间,先后产生了适合于不同机型的各种各样版本。其中影响最大的莫过于 Turbo Pascal 系列软件。

1983 年 11 月,美国 Borland 公司推出一种适用于微机的崭新 Pascal 编译系统,称为 Turbo Pascal(1.0 版)。Turbo 与 Turbine 同义,意即透平(机)或涡轮(机),似乎表明这一系统将是强力、高效的。不久,1.0 版被更新为 2.0 版。与此同时,又引入了在 Intel 8087 数字协处理器支持下适用的 Turbo-87 Pascal 编译系统,它提高了实数的运算速度并扩大了值域。1985 年,该公司又推出 Turbo Pascal 3.0,同时,能提高实数精度、提供格式化输出形式的 Turbo BCD Pascal 编译系统问世,它特别适合于商业应用。1987 年,Borland 公司开始销售 Turbo Pascal 4.0,并立即得到用户的广泛欢迎。Turbo Pascal 4.0 有如下几个突出的优点:①编译速度快。Turbo Pascal 1.0 和 2.0 的编译速度由通常 Pascal 系统的分钟级降到了秒级,Turbo Pascal 3.0 又比 2.0 提高了一倍。Turbo Pascal 4.0 的编译速度更快,达到每分钟可编译 27000 行源代码(在 8MHZ 的 IBM AT 机上);②操作环境好。Turbo Pascal 4.0 提供了一个设计精巧、易学好用的集成开发环境(Integrated Development Environment,简记 IDE),它集编辑、编译、连接、运行、求助、查错于一身,采用多窗口技术、多级菜单式驱动,上下文敏感式求助,可自由进入操作系统状态并返回,有良好的输入输出陷井控制和完善的交互式编辑功能,使编辑和修改源程序十分方便;③系统功能强。Turbo Pascal 4.0 在前几个版本的基础上又进一步扩充功能,提供了更为丰富的内部预定义标准子程序(约 200 个),它们功能齐全而且使用灵活,从而大大增强了其通用性。特别是 Turbo Pascal 4.0 引入了单元(UNIT)的概念,每个单元就是一段目标代码或称一个模块,它们可单独编译,供多处使用,需要时将它们连接起来。这就给应用程序的开发带来方便,并能有效地缩短开发周期。此外,Turbo Pascal 4.0 还提供了与汇编语言的接口以及直接把汇编指令写入程序的方式,支持多种新的数据类型及各类图形驱动器。

Borland 公司继 1987 年推出 Turbo Pascal 4.0 后,又先后于 1988 年推出 Turbo Pascal-5.0,1989 年推出 Turbo Pascal 5.5。两种新版本展现了若干新颖特色,使它们在微机程序设计中成为更加卓

有成效的开发环境和工具。Turbo Pascal 5.0 和 5.5 主要新增了以下诸功能：①提供了集成调试器，用户可以在集成开发环境下用它调试程序。它可完成对程序的追踪、步进、执行至光标、重新执行、设置断点、中断执行、检查和修改变量及内存地址等功能；②提供了在集成开发环境之外利用 Borland 公司的独立调试器 Turbo Debugger 调试程序的选择；③支持以单元为基础的覆盖模块，提供智能化的覆盖管理程序，使覆盖可以更容易和更快地执行；④支持 Lotus/Intel/MicroSoft 扩展内存规范(Expended Memory Specification，简记 EMS)。一旦加载到 EMS 内存，就关闭覆盖文件，减少后续覆盖的加载，从而加快内存转换；⑤提供了 8087 浮点仿真模式，即使未配置 8087 数字协处理器，亦可使用 IEEE 浮点数据类型，进行优化的浮点运算；⑥Turbo Pascal 5.5 支持面向对象的程序设计(Object-Oriented Programming，简记 OOP)这一新颖的程序设计手段，跟上时代的潮流。

此外，自 Turbo Pascal 5.0 起，增加了一个标准单元 OVERLAY，用以实现强有力的覆盖管理功能；还提供了支持新版本各项功能的若干实用程序和单元；新增了一批(约 30 个)内部预定义标准子程序，使标准子程序的数目达约 230 个之多；对某些编译指示、编译与运行出错编码及信息作了增、删或修改。

从 Turbo Pascal 4.0 开始，Borland 公司就率先使用了集成开发环境 IDE，令用户感到耳目一新，使用起来非常便利。IDE 为用户提供了一个十分友好的界面。Turbo Pascal-5.0 和 5.5 在这方面又有了显著的提高，它反映在用户界面的变化上。主菜单增加了两个可选菜单项——Debug 和 Break/watch，并以 Watch 窗口取代了 4.0 版中的 Output 窗口。各级下拉菜单中亦相应增、改了若干选项，使功能更加丰富、完善。

1990 年，Borland 公司又进一步推出了最新版本 Turbo Pascal 6.0，该系统提供了许多新的特性，主要有：①新一代的 IDE 用户界面，使能支持诸如鼠标器操作，多层覆盖窗口，多文件编辑，扩展的联机求助系统，条件断点等等功能；②Turbo Vision，它是个面向对象的应用框架和库，用它可高效地开发实用的、具有连贯一致用户界面的应用程序；③允许将汇编语言直接写入 Turbo Pascal 源程序中；④其编译器能使用扩充内存以建造大型应用程序。

我们相信，Turbo Pascal 系列软件作为开发系统软件与应用软件以及实施科学计算和教学的有力工具，正在发挥其越来越大的作用。它们实际上已成为微机上 Pascal 语言的主流，是目前在国内外最受欢迎的 Pascal 系统之一。

本书内容适用于 Turbo Pascal 5.5，部分内容引用了 Turbo Pascal 6.0 新增特性。为简化计，以下将 Turbo Pascal 作为 Turbo Pascal 5.5 的代名词。对于其它版本，则注明版本号以示区别，如 Turbo Pascal 4.0，5.0 或 6.0 等。

1.2 Turbo Pascal 系统安装

Turbo Pascal 系统软件存储在 4 张容量各为 360K 字节的 $5\frac{1}{4}$英寸双面双密度软盘上。它包括了运行集成开发环境和命令行编译版本所需要的一切文件和程序。为了便于将 Turbo Pascal 系统软件安装到 IBM PC 系列微机或其兼容机上，在该系统软件中装有一个专门的安装文件，其文件名为 INSTALL.EXE。INSTALL.EXE 将引导你逐步完成整个安装过程，你只要仔细阅读显示屏上对每一步骤所给出的指示信息并遵循执行之就行了。

运行 INSTALL.EXE 文件可实现三个功能：一是可将 Turbo Pascal 系统软件安装到硬盘上，此时，硬盘必须留有足够存储空间；二是可将 Turbo Pascal 系统软件安装到软盘上，此时，必须先准备

好 4 张容量各为 360K 字节的空软盘；三是可将硬盘上的 Turbo Pascal-4.0 或 5.0 版升级成 5.5 版，此时，需使用 INSTALL 中的 Upgrade(升级)选项。

现在假定有一台带有硬盘的 IBM PC 机。我们希望将 Turbo Pascal 系统软件安装到硬盘的当前目录上。可按如下步骤执行：①把 Turbo Pascal 系统软件中标记为 Installation Disk 的软盘插入 A 驱动器中；②键入 A：↵，这里↵表示按回车键；③键入 INSTALL↵，并遵循显示屏上给出的指示信息作出响应，直至完成整个安装过程为止。这台 IBM PC 机上就配有 Turbo Pascal 系统软件了。

Turbo Pascal 6.0 系统软件存储在 3 张容量各为 1.2 兆字节的 $5\frac{1}{4}$英寸高密软盘上。其安装方法与 Turbo Pascal 5.5 相似。

1.3 使用集成开发环境

本节描述 Turbo Pascal 用户界面并简单介绍如何使用 IDE，关于 IDE 的详情可参见附录一“Turbo Pascal 集成开发环境”。那里给出的是最新的 Turbo Pascal 6.0 集成开发环境。

Turbo Pascal 之所以深受广大用户欢迎，其主要原因之一就是它率先提出 IDE 概念并付诸应用，经过不断改进，为用户提供了一个十分友好的界面。

为启动 Turbo Pascal，在 Dos 操作系统状态提示符下键入 TURBO↵，Turbo Pascal 就进入了 IDE，显示程序第一屏包括主菜单屏幕和产品版本信息，可按任一键使版本信息消失。此时，显示屏呈现如图 1-1 所示的主菜单屏幕：

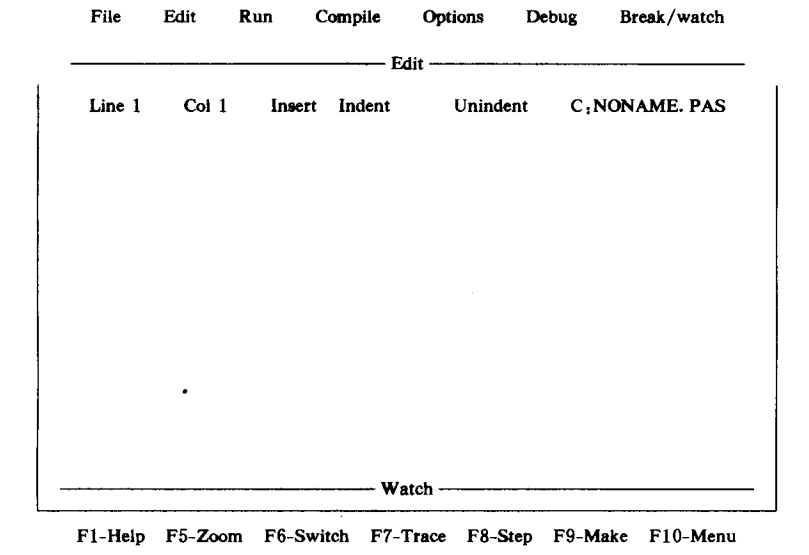

图 1-1 Turbo Pascal 主菜单屏幕

由图 1-1 可见，IDE 呈现的主菜单屏幕把显示屏分成四个部分：主菜单，Edit 窗口，Watch 窗口，以及底部状态行。

显示屏顶端是主菜单，它包含 7 个可选菜单项：File，Edit，Run，Compile，Options，Debug 以及 Break/watch。除了 Edit 选项外，其余 6 个均含有其相应的下拉式可选菜单项，而某些下拉式菜单项本身又含有再下一级的可选菜单项，从而构成了 IDE 独特的多级菜单结构形式。

主菜单中 7 个可选菜单项的功能简述如下：

File	处理与文件有关的操作，如文件的装入、选择、建立、存储、写至新文件等，还有目录列表、修改，切换至 Dos 操作系统状态，以及从 Turbo Pascal 出口退出 IDE。
Edit	进入 Edit 窗口，并激活 Turbo Pascal 编辑器。此时可编辑源文件。
Run	在 IDE 下用集成调试器运行或调试一个程序。
Compile	对当前源文件进行编译，提供对重编译的选择，设置目标代码的去向，发现运行错误，选定主文件，以及获得当前源文件的有关信息等。
Options	确定在 IDE 下工作的配置。这些配置将对编译器选择项，单元文件、目标文件和包含文件的目录，以及程序的运行参数等产生影响。
Debug	可检查和修改变量、表达式，显示当前调用堆栈，在源程序中快速定位过程和函数，打开或关闭集成调试程序，管理显示屏的切换等。它还提供了允许在 IDE 外使用 Borland 公司的独立调试器 Turbo Debugger 调试程序的状态开关。
Break/watch	允许添加、删去或编辑位于 Watch 窗口中的变量，以及在源程序中设置、清除、移动或观察断点。

在主菜单行下面是 Edit 窗口，即编辑窗口。程序源代码在 Edit 窗口中进行输入及编辑加工，它使用了 Turbo Pascal 的编辑器中各项功能。编辑器类似于一个简单的字处理器，可用它来方便地编写和修改程序。

Edit 窗口顶端有一个报告有关编辑器信息的状态行。状态行各项元素的含义如下：

Line n	显示当前源文件中光标所在行的行号。
Col n	显示当前源文件中光标所在列的列号。
Insert	当出现该指示符，表示为插入方式，此时所键入的任何字符均被插入当前源文件中光标所在位置。若指示符消失，则表示为覆盖方式，此时所键入的任何字符均替代了当前源文件中光标所在位置的内容。
Indent	当出现该指示符，表示为自动缩进，此时按↓键可使光标返至下一行中与上行开始处对齐的位置。若指示符消失，表示取消自动缩进，此时按↓键光标返至下一行第一列处。
Tab	表明已激活了 Tab 方式，即制表键(Tab 键)有效。在 Tab 方式下，按 Tab 键产生一个制表符(^I)并将光标右移一定数目的空格至某位置。图 1-1 所示，表示未激活 Tab 方式。
Fill	仅当 Tab 方式下工作。Fill 特性使得 Turbo Pascal 每行均以制表符与空格符的组合开始，从而减少了所使用的总字符数，优化了编码。图 1-1 所示，表示未启动 Fill 特性。
Unindent	当出现该指示符，表示为自动退格，此时按回退键可使光标回退至与当前行之上的第一个非缩进行对齐的位置。若指示符消失，表示取消自动退格，此时按回退键光标返至上一字符位置。

最后一项元素 C:NONAME. PAS 表示驱动器 C 中正在编辑的文件名 NONAME 和扩展名 PAS。

在 Edit 窗口之下有 Watch 窗口，即监视窗口。一旦将源程序输入 Edit 窗口，就可以在程序执行时使用 Watch 窗口来检验程序。在 Watch 窗口中可以显示、跟踪或改变程序中变量或表达式的值。

Turbo Pascal 主菜单屏幕中的最底下部分是底部状态行。它显示了 7 个功能键(称作热键)及相应功能,即:

F1-Help	F1 键调用求助功能,它打开 Help 窗口,并显示有关求助信息。
F5-Zoom	F5 键可将当前活动窗口扩大至整屏,或者将整屏缩小至原先大小。这是个跟斗式状态开关。
F6-Switch	F6 键可切换当前活动窗口,即可以从 Edit 窗口和 Watch 窗口之间进行跟斗式切换。因此,这也是个跟斗式状态开关。
F7-Trace	F7 键可用来单步跟踪所执行的程序或子程序。每按一次 F7 键,执行程序或子程序中的一步。
F8-Step	F8 键的功能与 F7 键相似,但它对子程序的调用视作一步执行。
F9-Make	F9 键用于重新编译当前源文件,但它仅编译为更新整个源文件所必须的最少数量模块。
F10-Menu	F10 键用于激活主菜单。

由于 IDE 中有许多频繁使用的菜单命令,IDE 为用户提供了 30 多个热键,这些热键用于直接执行某种功能,相当大一部分热键还与某些菜单命令等价。上述主菜单屏幕的底部状态行就显示了其中 7 个热键及相应功能。但底部状态行的内容并非固定不变。比如,按 ALT 键并保持数秒,则底部状态行显示由 ALT 键与其它几个功能键组合而成的一些热键及相应功能:

Alt: F1-Last help F3-Pick F6-Swap F9-Compile X-Exit

下面以回答问题的形式介绍 IDE 的一些基本操作。要真正熟悉 IDE 并能灵活使用,尚需大量上机实践及悉心体验。

①如何选择主菜单中某一选项? ——若在如图 1-1 所示的主屏幕状态,则可键入所需选项中高亮度显示的首字母(大、小写均可),如 F 表示选 File 选项,E 表示选 Edit 选项等。亦可使用光标键←或→在各选项间移动光条,当移至所需选项时,按↘键即可。另外,不论当前活动窗口在何处,均可使用同时键入 Alt 键与某选项首字母,达到上述同一目的。例如,用 Alt-R 组合键(同时按 Alt 键与 R 字母键)即可选择主菜单中 Run 选项。

②如何装入、保存 Turbo Pascal 源文件? ——简单的办法是按 F3 键装入源文件,此时应回答 IDE 询问的文件名;按 F2 键保存当前在 Edit 窗口的源文件。也可使用相应的菜单命令,如组合键 Alt-F-L 与 F3 键等价,Alt-F-S 与 F2 键等价。

③怎样获得联机求助信息? ——IDE 提供了上下文敏感式联机求助功能,并包含大量有关信息。可在除了运行程序以外的任何时刻按 F1 键,以得到光标当前位置条目的求助信息,它显示在一个窗口中,本身还包含一个或多个可以得到更详细信息的关键字,它们以高亮度显示,可移动箭头键至某一关键字并按↘键。在求助系统中再按 F1 键,可得到求助索引。顺着求助索引进行选择,亦可获得各方面的信息。此外,当使用编辑器时还可以按组合键 Ctrl-F1,以得到光标所在位置的标准单元、子程序、变量、常量、类型等有关信息。

④怎样扩大或缩小当前活动窗口? ——可以使用 F5 键,它是个跟斗式状态开关。

⑤怎样切换当前活动窗口? ——可以使用 F6 键,它亦是个跟斗式状态开关,使当前活动窗口为 Edit 窗口或者 Watch 窗口。

⑥如何返至主菜单? ——不管在什么窗口,按 F10 键即可返回主菜单。但 F10 键亦是个跟斗式状态开关,它在主菜单与当前活动窗口之间切换。

⑦怎样离开某一菜单? ——IDE 采用多窗口技术、多级菜单式驱动。为离开某一菜单,可按 Esc

键，此时返至上一活动窗口。当然亦可使用①中所示方法直接返至主菜单中某一选项。

⑧如何退出 Turbo Pascal 的 IDE，返至 Dos 操作系统状态？——简便的用法是按 Alt-X 组合键。亦可使用菜单命令组合键 Alt-F-Q.

⑨能否在使用 IDE 时，不退出 IDE 而执行某一 Dos 操作系统命令？——能。使用菜单命令组合键 Alt-F-O 可暂离 Turbo Pascal 系统，回到 Dos 提示符下。此时可键入 Dos 操作系统命令并执行。若需返至 IDE，键入 EXIT ↵ 即可。

1.4 编辑、编译和运行一个 Turbo Pascal 程序

下面给出一个简单的 Turbo Pascal 程序：

```
program Hello;
begin
  WriteLn('Hello!');
  WriteLn('This is your first Turbo Pascal program.');
  WriteLn('You are going to be a great programmer.');
end.
```

我们希望在 IBM PC 机上利用所安装的 Turbo Pascal 系统软件来编辑、编译和运行这个程序。可以分以下几个步骤来进行：

①开启机器，并进入包含 Turbo Pascal 系统软件的子目录。若设该子目录为 C:\TP ，则可键入 CD TP ↵，以将当前目录移至该子目录下。

②键入 TURBO ↵，进入 Turbo Pascal 的 IDE，显示主菜单屏幕和产品版本信息。按任一键使版本信息消失。此时，显示屏上呈现如图 1-1 所示的主菜单屏幕。

③按 F3 键以装入源文件。IDE 显示一个窗口，要求用户提供源文件名。不妨取上述程序（称作源程序）对应的源文件名为 HELLO. PAS，故可键入 HELLO. PAS ↵，此时，图 1-1 Edit 窗口顶部状态行最末一项改成 * C:HELLO. PAS，表示这是当前在 Edit 窗口中所编辑的源文件名。光标落在 Edit 窗口的第一行第一列处，等待输入源程序。

④逐行输入上述源程序。注意在每行末尾键入 ↵ 键，程序有缩进格式，分号和句点不可混淆。在这里可以使用 Turbo Pascal 的编辑器对源程序进行各种编辑加工，如插入、删除字符，插入、删除一行或若干行等等。有关编辑器的功能不在这里阐述。

⑤编译、运行所键入的源程序。可使用菜单命令组合键 Alt-R-R 实现之。若源程序正确无误，则将在 Watch 窗口显示运行该程序的结果，即三行文字信息：

```
Hello!
This is your first Turbo Pascal program.
You are going to be a great programmer.
```

⑥若欲保存正确的源程序，可键入 F2 键。此时硬盘中就存有名为 HELLO. PAS 的源文件。

如果还想再试一个 Turbo Pascal 程序，可以重复步骤③—⑥。倘若希望退出 IDE，返至 Dos 操作系统状态，则可键入组合键 Alt-X。

1.5 程序调试

Turbo Pascal 为用户提供了一个良好的集成开发环境，但用户所编写的程序本身，或者在键入

程序时，难免会有一些错误。排除和纠正计算机程序中的错误称作程序调试。因此，程序调试是上机实践中必不可少的环节。

一个源程序常见的错误大致有三类，即语法错误、语义错误和逻辑错误。

对于语法错误，诸如拼写有误、左右括号不配对等等，Turbo Pascal 的编译器将负责对源程序的语法进行严格检查，一旦发现语法错误，或称编译错误，就停止编译，进入编辑源程序状态，将光标定位在出错位置，并在 Edit 窗口中显示一条编译出错信息。待用户改正错误后，可重新进行编译。只有将源程序中所有的语法错误全部清除后，才能通过编译。

源程序通过编译并非表示它已正确无误。有些错误在编译阶段无法发现。例如，在程序运行中出现非法操作如零作除数，而在编译时作为除数的变量其当前值无法预知。因此，只有当运行源程序时，才会导致出错，称这种错误为语义错误或运行错误。对于此类错误，Turbo Pascal 将终止程序的执行，在显示屏上指示出错信息，并等待用户予以改正。

源程序中还可能有逻辑错误。有时即使源程序顺利地通过了编译，运行中也显示了结果信息，但却与预想的结果不相吻合。这就说明很可能在程序的某处隐藏了微妙的、不易觉察到的逻辑错误。这往往是最难查出的错误。Turbo Pascal 在其 IDE 中，为用户设计了一个功能强、灵活性高的源程序级集成调试器，用于快速、简便地发现并消除此类错误。主菜单中的三个可选菜单项 Run，Debug 以及 Break/watch 就是为此目的而设立的。利用 IDE 的集成调试器可完成对源程序的追踪、步进、执行至光标、重新执行、设置断点、中断执行、检查和修改变量与表达式以及内存地址等功能。此外，还设置了一些热键用于替代选择菜单的调试命令。详情可参见附录一。

1.6 编译指示简介

Turbo Pascal 借助于一系列编译指示来控制整个编译过程。这是 Turbo Pascal 编译器的一大特点。系统已规定好每一种编译指示的缺省值，藉以缩短代码长度，加快执行速度。同时，系统也允许用户根据需要和习惯来改置某个或某些编译指示的缺省值，或者在程序中按要求设置特定的编译指示。

在 Turbo Pascal 的 IDE 下，键入 Alt-O-C 可进入 Options 主菜单项中的 Compiler 选项，它将显示所有的编译指示（高亮度显示的字母为编译指示字母）以及它们的缺省值。可以对这些缺省值进行修改。一旦认可之后，便可作为系统新的编译指示缺省值，将影响后续对源程序的编译。

现在来看如何在程序中设置编译指示，以及编译指示的形式及含义。附录二给出了编译指示一览表。

编译指示写成具有特殊语法的注释形式。它以左花括号开头，紧跟一个美元符号 $ ，后跟有关信息，而以右花括号结尾。程序中凡是可以使用注释的地方，均可以出现编译指示。

Turbo Pascal 共有三种类型的编译指示：

①开关编译指示——通过在指示字母后面指定＋或-来打开或关闭某种编译性能。

②参数编译指示——指定影响编译的参数如文件名、单元名或内存设置等。

③条件编译指示——根据用户定义的条件符对部分源程序进行条件编译。

开关编译指示的一般形式是

{ $ 指示字母△，指示字母△，... }

其中指示字母有 A，B，D，E，F，I，L，N，O，R，S，V 等，△为字符＋或－。＋表示相应的编译性能是活动的，或称是打开的；-表示相应的编译性能是不活动的，或称是关闭的。指示字母各有其特定的含

义，用大、小写具有相同效果。例如

{$R+}　　表示进行下标范围检查；

{$D-,I-,S-}　　表示不产生调试信息，不检查 I/O 错误，不检查栈空间域是否溢出

开关编译指示分全程的和局部的两种。全程编译指示必须置于程序或单元的说明部分之前，它将影响所包含整个内容的编译；而局部编译指示可出现在程序或单元的任何地方，它只影响开关状态改变所包含内容的编译，例如

{$I-} Reset(F); {$I+}

其中{$I-}与{$I+}均是局部开关编译指示，这表示在执行过程调用 Reset 之前先关闭对 I/O 的出错检查，等执行完 Reset 之后又立即再打开对 I/O 的出错检查。

参数编译指示有如下三种形式：

{$ 指示字母(I 或 L) 文件名}

或

{$O 单元名}

或

{$M 栈大小，堆最小值，堆最大值}

请注意，在参数编译指示中，指示字母与其后的参数之间至少应有一个空格符。例如

{$I TYPES. INC} 表示在该编译指示所在位置把文件 TYPES. INC 的源代码嵌入正在编译的正文中；

{$M 65520, 8192, 655360} 表示指定栈大小为 65520 字节，堆最小值和最大值分别为 8192 和 655360 字节。

参数编译指示亦分全程的和局部的两种，其含义及用法与前述相同。

条件编译指示根据条件符号允许从同一源程序中产生不同的代码。由于本书很少使用此类编译指示，故介绍从略。

在以下各章中，对于源程序示例中用到的具体编译指示，还将作必要的说明，以利于用户逐步增进对如何合理使用编译指示的了解。

练　习

1-1.　请参考 1.4 节所给形式试写你的第一个 Turbo Pascal 程序，使得运行后能够显示匈牙利著名诗人裴多菲的如下一首诗

Life is dear indeed,

love is priceless too;

But for freedom's sake

I may part with the two.

(意即：生命诚可贵，爱情价更高；若为自由故，两者皆可抛。)

注意：Turbo Pascal 的语法规则规定，在使用 WriteLn 过程输出文字信息时，若文字信息本身包含单引号，则必须书写两次，即用两个单引号来产生一个单引号的输出。

第二章　基本元素与特性

2.1　程序基本结构

进行一项程序设计，一般应给定两组信息：首先，需精细地定义和说明将处理的数据，其次，应规定对这批数据所进行的操作。用 Turbo Pascal 进行程序设计，需要编写 Turbo Pascal 程序。一个 Turbo Pascal 程序，就是数据描述和一组必须由计算机一步一步地执行的“语句”或规则的汇集。它由程序首部、说明部分和执行部分所组成，形如

```
program 程序名;
  说明部分
begin
  语句 1;
  语句 2;
  ...
end.
```

第一行称为程序首部，它包含一个 Turbo Pascal 专用词汇 program，接着是程序员给该程序取的名字，最后以分号表示程序首部结束，下面将是程序主体的开始。程序首部在一个 Turbo Pascal 程序中并非必须出现，即它是可选的。写上它仅起文档作用。

说明部分用于定义和说明程序中要用到的数据，如常数、变量等。随着本书逐渐展开地描述，你将会看到说明部分的内容可以非常丰富。它除了可包含常数定义、变量说明外，还可有类型定义、标号说明、子程序（过程或函数）说明；若程序要用到已知的单元，则还应书写相应的 uses 子句；等等。但是，一个简单的 Turbo Pascal 程序亦可不包含说明部分，即说明部分也是可选的。1.4 节中给出的程序即是如此。

执行部分描述了程序要执行的操作。它必须以一个 Turbo Pascal 专用词汇 begin 开始，以另一专用词汇 end 后跟句点结束，其间是一些执行具体操作的语句，以分号作为语句间的分隔符。begin 与 end 应配对出现，这是每一个 Turbo Pascal 程序必须有的。紧跟 end 后的句点既表示执行部分的结束，亦表示整个程序的结束。此后若还有任何语句编码，Turbo Pascal 编译器将不予理会。另外，Turbo Pascal 规定：紧随 end 之前出现的分号允许省略。

说明部分和执行部分合在一起称为程序块。因此，我们说 Turbo Pascal 程序是一个块结构的程序。

Turbo Pascal 使用计算机能够识别和检验（通过 Turbo Pascal 编译器）的、类似于日常英语的专用词汇、术语及规则。它们构成了 Turbo Pascal 语言的语法。程序员使用它们编写 Turbo Pascal 程序时，可不必去管它们究竟是如何在计算机内转换并实现的，而只需致力于正确理解和使用这些词汇、术语并严格遵循既定的规则。但我们并不企图一开始就去完整地、从头至尾地介绍各种定义、词汇、规则、条文、限制等等，因为这往往会使初学者不得要领，从而对这种语言望而生畏。我们将采用符合人们理解力的习惯做法，即从最基本、最易于被接受的开始，然后由简到繁，逐步深入，陆续引出一些新的特性、规则，以解决新的问题。由于 Turbo Pascal 中的结构清晰，语句形式直观，因此易

于阅读与理解。下面就是一个简单而完整的 Turbo Pascal 程序，它要求从键盘键入一个正整数作为圆半径，计算圆的直径、周长及面积，并将结果显示输出。

```
program CircleDemo;
{ calculate diameter, circumference and area of a circle }
const
  Pai = 3.14;
var
  Radius : integer;
  Diameter, Area : real;
begin
  Write('Enter radius of a circle: ');      { whole number }
  ReadLn(Radius);
  Diameter := 2 * Radius;
  Area := Pai * Radius * Radius;
  WriteLn('Diameter of the circle is: ', Diameter:12:5);
  WriteLn('Circumference is: ', 2 * Pai * Radius:12:5);
  WriteLn('Area of the circle is: ', Area:12:5);
end.
```

程序首部中程序名取为 CircleDemo，以示这是有关圆的演示程序，Demo 是 Demons tration（示范、演示）的略称。

第二行至第七行是说明部分。其中，第二行为注释，它以花括号括起一段说明该程序功能的文字信息。第三、四行是常数定义，第五至七行是变量说明。const 和 var 是 Turbo Pascal 的两个专用词汇，分别取自 constant（常数）和 variable（变量）两个英文单词，表示以下是常数定义开始及变量说明开始。Pai，Radius，Diameter，Area 等含义自明。

执行部分共有七个将执行具体操作的语句，它们是：

①输出信息到显示屏的输出语句——Write 语句；

②等待从键盘键入圆半径数值的输入语句——ReadLn 语句；

③计算圆直径的赋值语句；

④计算圆面积的赋值语句；

⑤输出有关所求圆直径的信息及数值的输出语句——WriteLn 语句；

⑥输出有关所求圆周长的信息及数值（以表达式计值求得）的输出语句—— WriteLn 语句。

⑦输出有关所求圆面积的信息及数值的输出语句——WriteLn 语句。

虽然程序中有些具体细节尚未阐述，例如 Write 语句和 WriteLn 语句有什么区别，赋值语句的一般表示形式是什么，最后三个 WriteLn 语句中的":12:5"是什么含义，等等。然而，执行此程序所要完成的主要操作以及它们的执行顺序，还是比较清晰易辨的。

在计算机上藉助键盘键入上述程序，经 Turbo Pascal 编译器编译通过后，便可运行。下面是运行示例。为直观起见，在程序运行中由程序员所键入的内容下面加一横线，其余的都是由计算机显示输出的内容。

```
Enter radius of a circle: 10↵
Diameter of the circle is:  20.00000
Circumference is:  62.80000
Area of the circle is:  314.00000
```

2.2 基本字符集与标识符

2.2.1 基本字符集

Turbo Pascal 的基本字符集由基本字符(字母、数字) 和专用字符组成。它们均属于 ASCII 字符集(参见附录七)中的字符。

字母:英文大写字母 A 至 Z,小写字母 a 至 z

十进制数字:阿拉伯数字 0 至 9

十六进制数字:阿拉伯数字 0 至 9,英文字母 A 至 F,或 a 至 f

专用字符:+ - * / = < > [ ] . , ( ) : ; ^ @ { } $ #

下列符号均由两个专用字符组成,亦视作专用字符:

:= <> >= <= .. (* *) (. .)

其中:=表示赋值运算符;<>,>=,<=均为关系运算符,分别表示不等于,大于等于,小于等于;..表示子界分隔符;(*,*),(.,.)均为替代符号,它们分别等价于专用字符{,},[,].

此外,常用到的字符还有空格符,单引号字符,下划线字符,以及 ASCII 控制符(ASCII 码从 0 到 31)等。

2.2.2 标识符

标识符用以标记名字。程序中要用到许多名字,有些名字在 Turbo Pascal 中具有特定含义,称为保留字,如 program,const,var,begin,end 等;有些名字也是由 Turbo Pascal 规定的,用来标识预定义的类型、常数、变量、过程、函数、单元等,称为标准标识符,如 integer,real,Write,ReadLn,WriteLn 等;其余的则是应由程序员根据程序要求自行选定的名字,如对程序、常数、变量、标号、类型、过程、函数、记录字段、单元等等,均需取定相应的名字。所有上述这些名字都是用标识符来标记的。

在 Turbo Pascal 中,标识符的形成规则是:它必须由一个英文字母或下划线开头,后面可跟英文字母、数字和下划线的任意组合。标识符中的英文字母大、小写不予区分。一个标识符的字符长度仅由行的长度即 127 个字符所限制,但只有前 63 个字符有效。

下面是一些符合规则的标识符示例:

X begin CircleDemo U235

PROGRAMMING Department _ of _ Defense

S1 _ 000 _ 000 _ 000 _ 000

this _ identifier _ is _ very _ long _ but _ is _ still _ quite _ legal

TURBO Turbo turbo 三者认为是同一个标识符

非法的标识符示例:

1st 不能以非字母开头

Two Words 不能包含空格符

What _ is _ this? 问号符不允许

under-weight 减号符不允许

虽然 Turbo Pascal 不区分英文大、小写字母,但 GetDriverName 比 GETDRIVERNAME 更易读。本书中所用到的标识符,除了保留字以及几个常用的预定义数据类型全用英文小写字母外,一般都

使用这种混合形式，以增强可读性。

Turbo Pascal 共有保留字 52 个，其中打 * 号的是 Turbo Pascal 5.5 新扩充的保留字。不能把保留字用作用户说明的标识符，否则，Turbo Pascal 编译器将指示出错信息。以下是 Turbo Pascal 保留字表。

absolute	external	mod	shr
and	file	nil	string
array	for	not	then
begin	forward	* object	to
case	function	of	type
const	goto	or	unit
* constructor	if	packed	until
* destructor	implementation	procedure	uses
div	in	program	var
do	inline	record	* virtual
downto	interface	repeat	while
else	interrupt	set	with
end	label	shl	xor

除保留字外，Turbo Pascal 还规定了相当数量的系统预定义名字即标准标识符。与保留字不同的是，允许将它们重新定义，用作用户说明的标识符。但这样做将失去它们原定的特殊功能，从而容易引起混淆。本书中均使用它们的原有含义。

2.3 标准标量类型

Turbo Pascal 规定程序中所用到的变量需在说明部分加以说明。除了以某标识符作为变量的名字之外，还必须指明该变量所属的数据类型。一个数据类型定义了变量可接受值的集合以及对它所能执行的操作。Turbo Pascal 向程序设计者提供了一个丰富的数据类型集合，它们各用于专门的目的，但却都是由各种简单的、非构造型的数据类型所构成。本节所涉及的即是 Turbo Pascal 中最为基本的几种数据类型：整型、实型、布尔型和字符型。它们都是系统预定义的简单数据类型，称为标准标量类型。其对应的类型名字如 ShortInt，integer，LongInt，byte，word，real，single，double，extended，comp，boolean，char 等均是标准标识符。

2.3.1 整型

一个整型数据用来存放整数。Turbo Pascal 支持五种预定义整型，它们是短整型(ShortInt)，整型(integer)，长整型(LongInt)，字节型(byte)和字类型(word)．每一种类型规定了相应的整数取值范围以及所占内存字节数(一个字节为 8 个二进位)：

类型	取值范围	占字节数	格式
ShortInt	−128..127	1	带符号 8 位
integer	−32768..32767	2	带符号 16 位

类型	取值范围	占字节数	格式
LongInt	−2147483648..2147483647	4	带符号 32 位
byte	0..255	1	无符号 8 位
word	0..65535	2	无符号 16 位

Turbo Pascal 有两个预定义整型常数标识符 MaxInt 和 MaxLongInt，它们各表示确定常数值，供程序中直接引用：

整型常数标识符	值	类型
MaxInt	32767	integer
MaxLongInt	2147483647	LongInt

Turbo Pascal 还允许使用十六进制的整型数据。为规定一个整型值是十六进制，需在它之前加一个字符 $。例如 $38 表示一个十六进制整数，其值等于十进制的 56，又 $FE 的值等于十进制的 254。这里规定字母 A,B,C,D,E,F 分别表示十进制数 10,11,12,13,14,15。

2.3.2 实型

一个实型数据用来存放实数。Turbo Pascal 支持五种预定义实型，它们是基本实型(real)，单精度实型(single)，双精度实型(double)，扩展实型(extended) 和装配实型(comp)。每一种类型规定了相应实数取值范围，其所占内存字节数，以及它们所能达到的精度即有效数字位数：

类型	取值范围	占字节数	有效位数
real	2.9×10^{-39} 至 1.7×10^{38}	6	7 至 8
single	1.5×10^{-45} 至 3.4×10^{38}	4	11 至 12
double	5.0×10^{-324} 至 1.7×10^{308}	8	15 至 16
extended	1.9×10^{-4951} 至 1.1×10^{4932}	10	19 至 20
comp	$-2^{63}+1$ 至 $2^{63}-1$	8	19 至 20

上表中前四种类型给出的是其绝对值的取值范围。对于此类实型数据，若其绝对值大于上界，则产生上溢；绝对值小于下界，则产生下溢。下溢导致结果为 0。comp 类型的取值范围是 $-2^{63}+1$ 至 $2^{63}-1$ 之间整数，相当于十进制的 -9.2×10^{18} 至 9.2×10^{18}。由于 comp 类型的数据表示成二进制形式的数，这种类型的变量有时处理起来比较方便，特别对于数值很大的整数间计算，这种数据类型是很有用的。

所有程序都可使用基本实型 real，而对其它四种类型 single,double,extended 和 comp，则要求系统提供数字协处理器——8087,80287 或 80387 芯片，或者由软件模拟的仿真程序。前者的优点是提高对实型数据的处理速度，并减少舍入误差。Turbo Pascal 为后者提供了仿真模式。可以根据具体的计算机配置，使用 IDE 中菜单项选择或在程序中使用编译指示，来确定采用哪种方式。

2.3.3 布尔型

一个布尔型数据用来存放逻辑值，或称布尔值。Turbo Pascal 支持预定义布尔型，以标准标识符 boolean 表示。boolean 一词，系根据十九世纪英国数学家 George Boole(1815——1864)的名字而得，

George Boole 为现代布尔代数之父。类型 boolean 总共只有两个可能的值:False 和 True,它们分别表示逻辑假和逻辑真。False 和 True 都是标准标识符。每个布尔型数据在内存中占 1 个字节。规定 False<True。

2.3.4 字符型

一个字符型数据用来存放 ASCII 字符集内的某个字符。Turbo Pascal 支持扩展 ASCII 码,共包括 256 个不同字符。但其中有一些字符不能在标准显示设备如视频显示器或宽行打印机上显示或打印输出,即所谓非印刷字符。在计算机内部,字符集的元素是以该元素在字符集内的顺序位置来标记的,位置取值范围为 0 至 255,称这些整数为字符在字符集内的序数值或序号。每个字符型数据在内存中占 1 个字节。将字符用单引号括起,即成字符常数,如'a','3','?'。字符常数可按字符的序数值确定大小关系,如'Y'<'Z','A'<'a'。Turbo Pascal 提供表示字符常数的另外两种形式:使用字符 ^ 和 #,前者专为用于表示 ASCII 码表中序号为 0 至 31 的那些字符,它们各有特定含义,称为控制字符。Turbo Pascal 允许以字符 ^ 后跟一相应字母(或字符)来表示某一控制字符,比如 ^ G 表示序号为 7 的控制字符,这是个响铃符,^ M 表示序号为 13 的控制字符,即回车符。后者可用字符 # 后跟一序号表示相应字符,比如 #7 与 ^ G 一样,#65 表示字符'A',#218 表示某一特殊图形字符' '。

2.4 注释、常数定义和变量说明

2.4.1 注释

在 2.1 节程序 CircleDemo 中有一行注释。它以左花括号开始,接着是注释正文,最终以右花括号结尾。在程序中加上注释,有助于阅读和理解程序。Turbo Pascal 编译器对注释的内容予以忽略。

注释与空格符一样,被视作分隔符。所谓分隔符是指分开两个相邻的词(标识符、标号、专用符号、数字、字符串等)的符号。因此,注释可加在程序中任何能使用分隔符的地方。

注释可跨越若干行。例如,程序 CircleDemo 中的那行注释可改写成

```
{ calculate diameter, circumference
    and area of a circle }
```

象大多数编程语言一样,Turbo Pascal 允许插入任意多的注释。注释对于编译后的程序大小以及运行速度没有影响。但应注意,写注释必须花括号配对。如果键入时误将} 打成)或],则编译器会认为下面所有内容(包括程序最后一行 end 和句点在内) 均属于注释正文,因为它未遇到}.

由于 Turbo Pascal 识别(* 与 *)分别作为{与}的替代符号,因此可书写第二种形式的注释,如

```
(* calculate diameter, circumference
    and area of a circle *)
```

两种形式的注释可形成注释嵌套。因为若注释以(* 开始,则它将忽略所有的花括号,直至遇见 *),反之亦然。下面是注释嵌套的示例:

```
(*
Write('Enter radius of a circle: '); { whole number }
*)
```

其效果是把有 Write 语句的这一行注释掉，相当从程序“删去”这一行。注意，该行原有的注释亦一起被注释掉。

被嵌套的注释必须使用不同于嵌套注释的记号，并且同一层次注释的记号必须配对，即{与}或是(*与*)。

我们推荐用花括号对作为通常注释的开始与最终记号。而(*与*)则限于程序调试时临时性“删除”某些语句或程序段使用。调试完毕后，可将(*与*)去掉，以恢复原貌。

如果紧跟{或(*后面的是字符$，然后是一些有关信息，最终以}或*)结尾，那就是一个编译指示，而不是注释(参见 1.6 节)。

2.4.2 常数定义

程序中往往有一些经常使用的常数值，它们具有一定的含义，例如 7 可以表示一周天数，3.14 表示 π 的近似值，2.718 表示 e 的近似值，等等。为使程序简化并易于维护，Turbo Pascal 允许程序员使用常数定义，引入标识符作为常数值的同义词，供程序各处使用。

常数定义出现在说明部分。它以保留字 const 开始，后随一个常数赋值表。该表由若干个常数赋值组成，其间以分号分隔。每个常数赋值是一个标识符后跟等号及常数值。称这里的标识符为常数标识符。

常数定义示例：

```
const
  DaysPerWeek = 7;
  Limit = 255;
  Max = 1024;
  Pai = 3.14;
  E = 2.718;
  FirstChar = 'A';
  LastChar = 'Z';
  CommonPort = $3F8;
```

这些常数标识符被分别赋以整型常数、实型常数、字符常数等。常数标识符 CommonPort 被赋以十六进制整型常数 $3F8，等于十进制 1016。

对于下述常数定义

```
const
  C1 = -2;
  C2 = 200;
  C3 = 3679;
  C4 = 47420;
  C5 = 1000000;
```

Turbo Pascal 编译器自动选择最有效的整数类型，即占用内存空间最小的类型，因此 C1,C2,C3,C4,C5 被分别视作 ShortInt,byte,integer,word,LongInt 类型而分配内存空间。

用以上形式定义的常数称作无类型常数，因为在定义时无需指明数据类型。在程序中可到处引用这些常数标识符，但不允许改变其值。

Turbo Pascal 在程序书写格式上是比较自由的。一行上可以书写数个语句，只要语句间以分号隔开；一个空格符与一串空格符没有什么区别。因此，完全可以写

```
const C1 = -2; C2 = 200; C3 = 3679; C4 = 47420; C5 = 1000000;
```

不过就程序设计风格上来讲，还是按前面那样把保留字 const 自成一行，下面的常数赋值缩进对齐为宜。

2.4.3 变量说明

Turbo Pascal 规定程序中所用到的每个变量必须先说明后引用。为此，需在说明部分给出变量说明，以命名变量并指明其所属数据类型。

变量说明以保留字 var 开始，后随一个标识符说明表。该表由若干个标识符说明组成，其间以分号分隔。每个标识符说明包含一个或若干个(以逗号隔开)标识符，然后是冒号和数据类型。称这里的标识符为变量标识符，简称变量。编译器将为所有这些变量按所属数据类型分配一定的内存单元。在程序 CircleDemo 中，标识符 Radius 被说明为 integer 类型变量，而 Diameter，Area 均被说明为 real 类型变量。与常数定义相类似，变量说明的书写格式亦是灵活多变的。比如，可写

```
var
  Radius : integer;
  Diameter : real;
  Area : real;
```

或者

```
var
  Radius : integer; Diameter, Area : real;
```

等等。

变量说明中所出现的数据类型，除了可为系统预定义的那些数据类型外，还可以是在此前由程序员自定义的数据类型，比如枚举类型、子界类型(参见 2.5 节)，等等。

变量标识符与常数标识符有两个主要区别：一是变量标识符在变量说明时没有赋予初始值，如果一开始就引用，将得到不确定的数值，而常数标识符在常数定义时必须给出相应的常数值；二是变量标识符在程序运行中可不断改变其值，当然亦可随时引用，而常数标识符只能被引用，而不允许改变其值。

2.5 枚举类型与子界类型

程序员可根据需要自定义数据类型，这是 Turbo Pascal 强有力的特征之一。就简单数据类型而言，除了 2.3 节介绍的系统预定义标准标量类型之外，还包括可由程序员自定义的两种标量类型：枚举类型与子界类型。这可由类型定义来实现。类型定义出现在说明部分，它告诉编译器需要组织新的数据类型。在定义了一个新的数据类型之后，便可在变量说明中命名属于该数据类型的变量标识符。

Turbo Pascal 中的类型定义是以保留字 type 开始，后随一个类型命名表。该表由若干个类型命名组成，其间以分号分隔。每个类型命名包含一个标识符，然后是等号和数据类型。称这里的标识符为类型标识符。本节仅叙述如何定义枚举类型与子界类型。今后将陆续引出其它类型定义。

2.5.1 枚举类型

枚举类型定义了一个有序的值的集合。这些值是通过枚举描述性标识符的方法来指示的，称作枚举直接量。枚举类型定义的一般形式是

```
type
```

类型标识符 = (标识符 1,标识符 2,...,标识符 n);

其中 type 是保留字,下面可跟随若干个上述形式的类型定义。

例如,下述书写形式

```
type
  Color = (Red, Blue, Green, Yellow, Purple);
  Sex = (Male, Female);
  Suit = (Club, Diamond, Heart, Spade);
  Day = (Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday);
  Switch = (Off, On);
```

定义了五个枚举类型 Color,Sex,Suit,Day,Switch 以及它们各自所能取到的所有值。若接着有变量说明

```
var
  MyFavorColor : Color;
  YourSex : Sex;
  PlayingCard : Suit;
  Today, Yesterday, Tomorrow : Day;
  Status : Switch;
```

则 MyFavorColor,YourSex,PlayingCard,Today,Yesterday,Tomorrow,Status 都是枚举类型变量,它们各自可取相应的枚举直接量作为其值。

Turbo Pascal 允许在一个枚举类型中最多有 65536 个枚举直接量,然而在通常情形下只需定义不到 10 个。Turbo Pascal 编译器接受这些描述性标识符,并将其转换成一个以 0 开始的有序的数列,比如在枚举类型 Color 中,Red 的序数值是 0,Blue 的序数值是 1,依此类推,最后 Purple 的序数值是 4,这同时也就确定了这几个枚举直接量之间的大小顺序,即有

Red < Blue < Green < Yellow < Purple

而如果程序员把类型定义写成

```
type
  Color = (Red, Yellow, Green, Blue, Purple);
```

那么 Yellow 所对应的序数值将是 1 而不是 3,同时有

Red < Yellow < Green < Blue < Purple

因此,这完全取决于程序员在定义枚举类型时所列各描述性标识符的顺序,同时也说明了为什么把象 Color 这样的标量类型称作枚举数据类型并把这些描述性标识符称作枚举直接量的原因。事实上,在经编译后的代码中,这些描述性标识符不再存在,而是以相应的序数值代替。而对于程序员来说,可以忽略这个技术细节,使用这些含义明确的描述性标识符。

上例中的类型定义和变量说明可由一简化形式表示,即不出现类型定义,直接将变量说明为枚举类型变量。比如

```
Var
  MyFavorColor : (Red, Blue, Green, Yellow, Purple);
```

可以想见,Turbo Pascal 预定义的布尔类型具有如下枚举类型定义

```
type
  boolean = (False, True);
```

因此,False 与 True 的序数值分别是 0 与 1,并且有 False<True。