北京希望电脑公司高级程序设计丛书

Turbo Pascal 6.0
高级程序设计技术

亦鸥　萧逸　等编著

海洋出版社

北京希望电脑公司高级程序设计丛书

Turbo Pascal 6.0
高级程序设计技术

亦 鸥 萧 逸 等编著

海洋出版社
1992·北京

内 容 简 介

本书详细介绍了最新版本的 Turbo Pascal 6.0 的各种高级程序设计技术．重点讲述了面向对象程序设计和图形程序设计技术．叙述通俗易懂，示例丰富，强调 Turbo Pascal 程序设计的特点，注重方法与技巧的训练，有助于读者良好程序设计风格的培养．

本书既可作为高等院校计算机专业的学生学习 Pascal 语言的教学参考书，亦可供其他专业的学生以及从事计算机系统软件及应用软件开发的工程技术人员参考．

需要本书的读者请与北京 8721 信箱联系，电话：2562329，邮政编码 10080．

(京) 新登字第 087 号

Turbo Pascal 6.0 高级程序设计技术

亦 鸥　萧 逸　等编著

*

海洋出版社出版（北京市复兴门外大街 1 号）

新华书店北京发行所发行　双青印刷厂印刷

开本：787×1092 1／16　印张：23.25　字数：504 千字

1992 年 9 月第一版　1992 年 9 月第一次印刷

印数 1—3000 册

*

ISBN 7-5027-2799-X／TP・110　定价：15.00 元

前　言

Turbo Pascal 是美国 Borland 国际公司推出的产品，它编译速度、运行效率高，可在 PC DOS、MS DOS 或 CP／M80、CP／M86、OS／2 等操作系统支持的 IBM PC／XT、AT、286、386、PS／2 及其兼容的个人计算机上运行．应用非常广泛．

它的主要特点有:

一、与国际标准基本兼容，并作了若干扩充

Turbo Pascal 实现了 ISO7185-1983 标准的 0 级遵从，还扩充了许多其他功能，诸如与计算机硬件有关的绝对地址变量、机器字位及字节操作、中断处理、内存管理、外部子程序调用、程序内机器代码、内部数据格式、嵌入文件、整数逻辑操作和结构常量等等．另外，针对广泛的 IBM 微机用户，还提供了图形及色彩、窗口和声音等特殊功能，通常它们都是通过需求函数和过程来引用的．这些手段都充分发挥了微机自身的特点和功能，使得 Turbo Pascal 语言的表达能力更强、更实用．

二、用户界面好，编程效率高

Turbo Pascal 具有工具箱式的环境，有统一的界面．提供了集高性能文件管理、编辑、编译、调试运行为一体的集成开发环境（IDE）．自 4.0 版本以后，这个界面是彩色多窗口的．现在的 6.0 版本，更是为之一新，给程序员带来编程效益的提高，它支持:

- 多重叠窗口;
- 鼠标、菜单、对话窗;
- 多文件编辑器，可编辑的文件长度可达 1Mb;
- 增强性调试工具;
- 完整的工作面保存和恢复功能.

三、查错功能强，错误定位准

Turbo Pascal 的 IDE 交互性能好，对文件可以统一处理，直接支持编辑、编译、连接装配、运行和调试等各阶段工作.

值得特别指出的是，Turbo Pascal 在编译时的查错功能甚强．一旦发现有错，立即进入编辑状态进行修改，修改后直接转编译状态编译源程序．它的错误定位及错误性质的指示相当准确．这一点对程序员而言无疑是很受欢迎的．在 Turbo Pascal 中仅当修改了当前找到的第一个错误之后，才能继续编译下去．这一方面说明其编译算法设计得好，发挥了微机的特点，同时反映了 Pascal 语言的定义严谨.

四、单元可分别编译，有利于大型软件的开发

Turbo Pascal 中的一个单元(Unit)是常数、数据类型、变量、过程及函数的集合，每个单元很像一个独立的 Pascal 程序，它有必要的初始化代码和程序体，是允许程序分块独立编译的说明库.

五、带面向对象的扩充，支持面向对象程序设计(OOP)

Turbo Pascal 自 5.5 版本起，扩充了支持面向对象程序设计的设施。在 Turbo Pascal 6.0 中又得到了进一步的改进和充实，使得 Turbo Pascal 也跻身于 OOP 的行列，成为面向对象程序设计的有力工具。

六、其他

(1) 它规定了无类型变量，对于书写系统软件这是绝对必要的；

(2) 说明部分的各个说明段次序可随意，十分灵活方便；

(3) 自带的编辑器与 WordStar 命令几乎相同，广大用户不必专门为了掌握编辑程序而花很多功夫；

此外，Turbo Pascal 6.0 还有以下一些新功能：

(4) 面向对象的应用工具 Turbo Vision，可直接应用于用户程序(它提供了用以书写 IDE 的工具)；

(5) 功能完备的嵌入式汇编器；

(6) 对象说明中的私有域和私有方法；

(7) 扩展的语法指令($X)，可将函数与过程等同对待(忽略函数的返回值)；

(8) 286 代码生成；

(9) 类型常量中的地址引用；

(10) 远过程和近过程指令($G)；

(11) 从目标文件中链接初始化数据($L)；

(12) 一种新的堆管理器，快速而且可以减小零碎的堆单元；

(13) 增强的联机帮助工具，每个库函数和过程均附有示例代码。

本书详细介绍了最新版本的 Turbo Pascal 6.0 的各种高级程序设计技术。叙述通俗易懂，示例丰富，强调 Turbo Pascal 程序设计的特点，注重方法与技巧的训练，有助于读者良好程序设计风格的培养。

本书既可作为高等院校计算机专业的学生学习 Pascal 语言的教学参考书，亦可供其他专业的学生以及从事计算机系统软件及应用软件开发的工程技术人员参考。

由于水平、时间所限，不妥之处在所难免，欢迎读者批评指正。

编著者

1992 年 9 月

目 录

第一篇 高级程序设计基础

第二篇 面向对象程序设计技术

第三篇 高级程序设计进阶

第四篇 高级图形程序设计

第一篇　高级程序设计基础

第一章　程序和单元

1.1 程　　序

Turbo Pascal 的程序除了程序头和可选的 uses 子句外，采用过程说明的形式。

程序头指出程序名和参数。

uses 子句说明所有被程序直接和间接使用的单元。

程序自动使用 System 单元，System 单元实现了文件 I/O、字符串处理、浮点运算、动态内存分配等低层例程。

除了 System 单元，Turbo Pascal 还实现了如 Printer、Dos、Crt 等许多标准单元，这些单元不能被自动使用，必须用 uses 子句说明包括它们，如：

```
uses Dos,Crt;                                    {使用 Dos 和 Crt 中的例程}
```

在 uses 子句中列出的单元顺序决定了它们被初始化的顺序。

1.2 单　　元

单元是 Turbo Pascal 模块编程的基础，它用来创建能由许多程序引用而不需要源程序的库，以便把大程序分成逻辑相关的模块。

1.3 什么是单元

Turbo Pascal 提供了大量的预定义常量、数据类型、变量、过程和函数，让用户程序存取。这些定义中，一些是 Turbo Pascal 所独有的，另外一些是 IBM PC 及其兼容机或 MS-DOS 所独有的。单个程序不可能用到所有的说明，所以可把相关的说明分成组，称为单元。用户程序只使用所需要的单元即可。

单元是常量、数据类型、变量、过程和函数的集合。每个单元像一个独立的 Pascal 程序：有必要的初始化代码和程序体，程序可在首部调用它。简单地说，单元指说明库，可将说明置于程序中，且允许程序分块独立编译。

单元中的说明通常互有联系。比如，Crt 单元含有 PC 屏幕子程序的所有说明。

Turbo Pascal 提供 8 个标准单元，其中有 6 个 —— System，Overlay，Graph，Dos，Crt 和 Printer —— 支持通常的 Turbo Pascal 程序，它们都存放在 TURBO.TPL 中。另二个 —— Turbo3 和 Graph3 单元 —— 设计用来帮助保持 3.0 版本的程序和数据

与 6.0 版兼容．在此我们对标准单元就不做详细的解释了，只进行一般性的介绍．

1.4 单元的结构

单元通过支持常量、数据类型、变量，提供过程和函数，从而实现某种功能．单元隐藏了功能实现的两部分(接口部分和实现部分)实际实现的细节．程序使用单元时，就像在程序中定义它们一样．可以使用单元中的所有说明．

单元结构与程序类似，但有一些重要区别．例如，下面是一个单元的形式:

```
Unit <标识符>;
interface
Uses <单元表>;{可选}
{公用说明}
implementation
Uses <单元表>;{可选}
{私有说明}
{过程和函数的实现}
begin
  {初始化代码}
end
```

单元头以保留字 unit 开始，后跟单元名(标识符)，类似程序的开始．单元头后面是关键字 interface，表示单元接口部分的开始．单元接口部分指在使用该单元的其他单元中可见的部分．

单元可用 uses 子句指定使用的其他单元．uses 子句可出现在两个地方．第一个地方是在关键字 interface 后面．此时，在别的单元接口部分说明的常量和数据类型可以在这个单元的接口部分的任何说明中引用．

第二个地方是出现在关键字 implementation 后面，此时，被使用单元中的说明只在这个单元的实现部分才能使用．这种机制允许递归引用单元，本章后面将具体介绍这一点．

1.4.1 接口部分

单元接口部分("公共"部分)从保留字 interface 开始，它出现于单元头之后，保留字 implemention 之前．接口部分决定了在使用该单元的程序和单元中的可见部分，任何使用该单元的程序和单元均可存取这些"可见"项．

在单元接口部分，可说明常量、数据类型、变量、过程和函数．像程序一样，可任意安排它们的顺序，可重复写，如:

```
type ... var ... <procs> ... const ... type ... const ... var
```

在单元中，能被其他单元或程序使用的过程或函数在接口部分说明，但是实际的程序体在实现部分中定义．forward 说明既不必要也不允许．

在接口部分中列出了所有过程和函数头之后，再在实现部分列出常规过程和函数程

序体.

uses 子句可出现在实现部分，跟在关键字 implementation 后面.

1.4.2 实现部分

实现部分("私有"部分)从保留字 implementation 开始. 任何在接口部分说明的对象: 常量、类型、变量、过程和函数，在实现部分是可见的. 不仅如此，实现部分还有本身的说明，但是说明的对象对使用该单元的程序或单元是不可见的. 程序不知道它们的存在，不能引用或调用它们. 但是这些说明的项可以被"可见"的、在接口部分说明的过程和函数使用.

如果在实现部分出现 uses 子句，则必须紧跟在 implementation 关键字之后.

如果过程说明为 external(外部)，那么一个或多个{$I filename} 指令应该出现在源文件中单元的最后一个 end 之前.

在接口部分说明的一般过程和函数(非内部子程序)，必须在实现部分重新出现. 出现在实现部分的过程和函数头，必须与接口部分一致或者用简写的形式. 简写形式是指 procedure 或 function 后面跟子程序名(标识符). 子程序含有所有的局部说明: 标号、常量、类型、变量以及过程和函数，这些说明后面是子程序体. 假如下面说明出现在单元的接口部分:

```
Procedure IsWap(var v1, v2: integer);
function IMax(v1, v2:integer):integer;
```

那么，实现部分可以采取以下形式，

```
procedure IsWap;
var
  Temp: integer;
Begin
  Temp:=v1; v1:=v2; v2:=Temp;
end; {of Pro IsWap}

function IMax(v1, v2:integer): integer;
begin
  if v1 > v2 then
    IMax:=v1
  else IMax:=v2;
end; {of func IMax}
```

局部于实现部分的子程序(亦即，没有在接口部分说明的局部子程序)必须有完整的过程和函数头.

1.4.3 初始化部分

单元的实现部分通常在保留字 implementation 和最后一个 end 之间. 然而，如果把关键字 begin 放在最后一个 end 之前，那么，两个语句之间的复合语句(类似于程序