清华松岗系列丛书

Borland C++4 编程指南（第三版）

Ted Faison 著　　沈金发 吴志美 张新宇 译

清华大学出版社

Borland C++4 编程指南

（第三版）

沈金发　吴志美　张新宇　译
李　莉　审校

清华大学出版社

(京)新登字 158 号

Borland C ++4 编程指南

Borland® C ++4 Object-Oriented Programming (Third Edition)

Ted Faison

图书在版编目(CIP)数据

Borland C++ 4 编程指南/(美)Faison, T. 编著；沈金发等译. —北京：清华大学出版社，1995.7

ISBN 7-302-01776-X

Ⅰ.B… Ⅱ.①F… ②沈… Ⅲ.C 语言-程序设计-指南 Ⅳ.TP312C-62

中国版本图书馆 CIP 数据核字(95)第 10943 号

出 版 者：清华大学出版社 (北京清华大学校内，邮编 100084)
责任编辑：张孟青
责任校对：李凤茹
印 刷 者：清华大学印刷厂
发 行 者：新华书店总店北京科技发行所
开 本：787×1092 1/16 印张：42.75 字数：1013 千字
版 次：1995 年 9 月第 1 版 1995 年 9 月第 1 次印刷
书 号：ISBN 7-302-01776-X/TP・784
印 数：0001—3000
定 价：78.00 元

引　言

20世纪80年代，C语言成为最主要、最通用的程序设计语言，使用C语言可以高效地写出可移植到各类计算机上的代码，从而使软件编制得更快，工程总体规模增大。随着工程规模的增大，软件复杂性也增加了，导致软件开发时间延长，如何缩短软件的研制时间，提高软件开发效率是许多公司面临的主要问题。AT&T公司扩充了ANSI C，开发C++语言，试图将面向对象程序设计的许多优点加入C语言，同时保留了C大受欢迎的许多特点，如简洁性、运行高效性等。

C++是为简化程序设计而开发的，因此，该语言本身要比其前身C语言更复杂。C++中增加特点的目标在于减少软件开发的困难程度。很显然，仅仅采用C++并不能保证能开发出更好或更简单的软件。为获得使用C++所带来的好处，还必须采用一种新的程序设计方法，通常称作面向对象的程序设计方法，简称OOP。

为什么使用OOP方法？

几年前，计算机科学研究者发现：不管使用何种语言，程序员都可能书写和调试几乎完全等量的代码。工作量大体相等，但结果却有所不同。编写100行C语言程序，与编写100行汇编语言程序一样困难，但同样长度的C代码能做更多的事情。考虑到这一点，研究者们试图开发一种更高层次的语言，使单个程序员的能力加倍，从而减少项目开发时间和费用。

在70年代，“对象”这一概念在程序设计语言的研究者中间流传开来。一个对象是代码和数据的集合，它可以模拟自然的或抽象的实体，对象作为有效的程序设计项有两个主要原因：它们代表了普遍使用的物体的直接抽象，而且，对用户掩蔽了实现的复杂性。最早研制出的对象，是与计算机联系非常密切的项，如整数、数组、栈等。一些语言（如Smlltalk）是做为传统语言开发出来的，这些语言中的任何成分都定义为对象。

面向对象的程序设计方法把重点放在对象之间的联系，而不是实现的细节上。联系是对象之间的纽带，通常是通过族谱形成的，新的对象类型就是通过族谱由其它对象类型生成的。把一个对象的实现细节掩藏起来，可以使用户把注意力集中在对象与系统其他部分之间的关系上，而不是对象行为的实现。这种区别是很重要的，这表明了与早期的“命令”语言（如C语言）的根本脱离，在那些命令型语言中，函数和函数调用是程序设计的中心。

在C++中，语言本身的对象很少，设计对象的职责和重任由用户完成。Borland C++包含一些对象类型，不过，要实际应用该语言还需要设计更多的对象。设计出一组组相互关联的对象，就能使OOP的作用大大发挥出来。这些组通常称为类的层次，设计类的层次是OOP的中心。

本书结构

本书描述了基于 Borland C++ 4.0，使用 Borland 公司提供的对象类型来设计对象等级的方法。在此之前，首先介绍了 C++的基本特征。C++的主要 OOP 特征在各章分别介绍。

该书分为两部分。第一部分，“C++”，从总体上描述了 C++语言，侧重于 Borland C++和使之成为面向对象语言的特点。本书并无意作为 Borland C++的完整参考，而是想告诉读者怎样从面向对象的角度来运用语言的特点。

第二部分，“OWL”。该部分从第 10 章开始，展开了怎样使用 OWL，本书自始至终给出了大量的程序实例，读者可从本书所附的软盘中得到所有例子的源代码。每一个例子都调试过，可以编译和立即执行。

该书的结构有些不寻常。尽管表面上是直叙的，实际上有时需要查阅后来的章节。这是因为本书选择系统地描述 C++，而不是循序渐进的方法。这使得在本书中查阅信息很方便。例如，第 2 章有关析构函数的部分包括了有关析构函数的所有信息，也提到了虚析构函数，尽管虚析构函数在第 5 章才详细描述，这种表达顺序与大多其他 C++书不同，这样做利一定大于弊。

本书的描述

本书从总体上论述 C++编程方法，侧重于 Borland C++ 4.0。当某个具体方面与被推荐的 ANSI C++标准通用时，常常指的是 C++，而不是 Borland C++。因为 Borland C++实质上是 AT&T C++公开版本 2.1 的超集。声明这一区别是很有必要的。

第 1 章，“基础知识”，总结了 Borland C++的主要结构而没有作任何形式的定义或说明，着重描述 C++与 ANSI C 不同的方面。尽管 C 程序可用 C++编译器编译，C++并不能使用与 C 完全相同的技术。本章还指出 C++废弃了的一些 C 的特征。

第 2 章，“对象和类”，是 ANSI C 的面向对象扩展的真正开始。它介绍了对象和类的新概念。本章描述了怎样使用代码和数据来建立对象，对象是怎样使用的，有什么性质。

第 3 章，“继承性”，说明了对象是怎样从其他对象建立起来的，而不是乱构造的。继承使对象能够继承父类的性质，从而减少了完成任务所需的编码量和调试量。继承性还使得类可以象黑匣子一样重复使用，提高了程序员的编程效率。本章讨论了继承和多重继承两种情况。

第 4 章，“重载”，讲述了函数和操作符重载。有经验的程序员看到这儿也许会打呵欠，不过，即便如此也不能“跳”过这一章，因为重载允许不同的类使用统一的符号形式表示概念上相似的动作，这是一个很重要的性质，是 C++提供给程序员的一个简化措施，帮助他们更好地管理大型工程。

第 5 章，“多态性”，是 C++最大力推荐的一个特点。本章详述了多态性，说明了通过虚函数的应用，简化程序设计的具体方法。本章还讲述了虚函数的优缺点及其运行特点。

第 6 章,“例外处理”,讨论了 ANSI C++建议中的多态性所新增的特点。它容许程序以很直接的方式处理例外的情况,这就可使复杂的程序作了相当大的简化。

第 7 章,“流”,讲述输入和输出(I/O)。任何程序必须能输出结果才会有用。因此必须有输出信息的方式。总的说来,程序既需输入也需输出。本章从新的 C++结构——流的角度描述了输入和输出。I/O 流对文件和硬设备是一致的。该概念也适用于对内存的操作。

第 8 章,“基于对象的包容类库”介绍了 Borland C++ 2.0 库中的每一个类。类库是基于类似 Smalltalk 层次的,它是以类 Object 作为根的。

第 9 章,“基于模板的包容类库”,也称为 BIDS(Borland International Data Structure)库,这是模板类库,它们是 Borland C++ 4.0 新的包容的基础。使用基于模板的包容,用户能够处理在包容中的标量和类变量。

第 10 章,“ObjectWindows 库类”,包括了许多 OWL 程序例子,这些在 Borland 文档中是不介绍的。

第 11 章,“MDI 应用程序”,介绍了如何得到基本的 OWL MDI 程序以及通过一系例的定制来完善它。如果要使用 MDI,则一定会发现本章是很有用的。

第 12 章,“列表框的变化”,给出了许多种方法,用这些方法可使标准 Windows 列表框完善以满足许多程序设计方案,这是使用了从 TListBox 中派生的类。运用类似的技术,还可以定制出其它的标准 Windows 控制。

软硬件需求

学编程方法可以不需要计算机,不过,有一台计算机将大有帮助。为掌握本书的内容,不仅需要研究不同例子的源代码,还得试着做些改动,并编译。由此,需要下面一些软硬件:

- IBM PC AT 机或其兼容机
- MS-DOS 3.31 或更高版本
- 与 Microsoft 兼容的鼠标
- OWL 2.0, VGA 或更好的图形适配器
- Borland C++ 4.0 版本
- OWL 2.0(用于第 10 至 12 章)
- Windows 3.1 或更高版本

第 10 章到第 12 章范例程序的项目文件是用 Borland C++ 4.0 版本编写的。如果用以前的版本,也许不能成功地使用这些项目文件。

微软公司用于 Windows 软件开发的工具(SDK)并不一定需要。如果有 Turbo C++ 或 Borland C++ 4.0 以前的版本,仍可以编译 1 至 7 章的大部分代码,不过第 8 或第 9 章的包容类将不能用——除非改变代码。

第 10 章到第 12 章的 Windows 代码需要 Borland C++ 4.0。如果用 Borland C++ 3.0 和 Windows3.1,需要对源代码或项目文件做修改。

目　　录

第一部分　C++

第1章　基础知识……(3)
1.1　Borland C++项目文件的结构 ……(3)
1.1.1　头文件 ……(3)
1.1.2　一个完整的样本程序 ……(5)
1.2　变量 ……(9)
1.2.1　作用域 ……(9)
1.2.2　类型……(11)
1.2.3　存储类……(11)
1.2.4　const 限定符 ……(12)
1.2.5　Volatile 限定符 ……(14)
1.3　语句……(14)
1.3.1　表达式语句……(15)
1.3.2　if 语句 ……(16)
1.3.3　switch 语句……(17)
1.3.4　标号语句……(19)
1.3.5　while 语句 ……(19)
1.3.6　do while 语句 ……(20)
1.3.7　for 语句 ……(20)
1.3.8　break 语句 ……(21)
1.3.9　continue 语句 ……(22)
1.3.10　GOTO 语句 ……(23)
1.3.11　return 语句 ……(23)
1.4　函数……(24)
1.4.1　参数传递……(24)
1.4.2　const 参数传递 ……(25)
1.4.3　使用缺省参数……(25)
1.4.4　函数返回值……(26)
1.4.5　返回 const 项 ……(27)
1.4.6　返回值方面的问题……(27)

1.4.7 使用函数修饰符 …… (28)
1.5 指针与引用 …… (30)
1.6 指针、引用与 const 连用 …… (32)
1.7 提高部分 …… (33)
1.7.1 使用嵌入式汇编语言 …… (33)
1.7.2 名字分裂 …… (35)
1.7.3 C 和 C++合用 …… (35)
1.7.4 传值方式返回结构 …… (36)
1.7.5 使用枚举类型 …… (38)
1.7.6 使用内存管理 …… (40)
1.7.7 理解 C 参数传递顺序 …… (42)
1.7.8 理解 pascal 参数传递顺序 …… (43)
1.7.9 使用 Interrupt 处理函数 …… (43)
1.7.10 使用参数数目可变的函数 …… (44)

第 2 章 对象和类 …… (47)
2.1 定义类 …… (47)
2.1.1 类标识符 …… (48)
2.1.2 类体 …… (48)
2.2 使用类 …… (49)
2.2.1 封装 …… (50)
2.2.2 类存取控制 …… (50)
2.2.3 类私有成员 …… (51)
2.2.4 类公有成员 …… (52)
2.2.5 类保护成员 …… (53)
2.2.6 类对象的存储类 …… (54)
2.2.7 类作用域 …… (55)
2.2.8 空类 …… (55)
2.2.9 类嵌套 …… (55)
2.2.10 类的实例化 …… (58)
2.2.11 不完全的类声明 …… (58)
2.3 使用数据成员 …… (58)
2.3.1 静态数据成员 …… (59)
2.3.2 private static 数据成员 …… (61)
2.3.3 类对象用作数据成员 …… (62)
2.3.4 数据成员的引用 …… (64)
2.3.5 指针数据成员 …… (65)
2.3.6 指向类数据成员的指针 …… (65)

2.3.7 指向对象数据成员的指针 …… (67)
2.4 使用成员函数 …… (67)
2.4.1 简单成员函数 …… (68)
2.4.2 静态成员函数 …… (69)
2.4.3 Const 成员函数 …… (70)
2.4.4 volatile 成员函数 …… (70)
2.4.5 内联成员函数 …… (71)
2.4.6 带有 const this 的成员函数 …… (72)
2.4.7 带有 volatile this 的成员函数 …… (73)
2.4.8 特殊类函数 …… (75)
2.4.9 构造函数 …… (75)
2.4.10 析构函数 …… (82)
2.4.11 friend(友元)关键字 …… (84)
2.4.12 友元的性质 …… (85)
2.5 提高部分 …… (86)
2.5.1 成员函数指针 …… (86)
2.5.2 数组和类 …… (89)
2.5.3 成员函数调用的分析 …… (94)
2.5.4 类模板 …… (96)
2.5.5 函数模板 …… (103)

第3章 继承性 …… (106)
3.1 可复用性 …… (106)
3.2 继承性 …… (106)
3.3 继承的作用 …… (107)
3.4 C++继承性的局限性 …… (108)
3.5 关于继承的不同观察角度 …… (108)
3.6 单一继承 …… (109)
3.6.1 何时继承 …… (109)
3.6.2 不能被继承的成分 …… (109)
3.6.3 存取基类的限定符 …… (110)
3.6.4 可被继承的类 …… (110)
3.6.5 传递给基类的参数 …… (112)
3.6.6 构造函数的调用顺序 …… (113)
3.6.7 析构函数的调用顺序 …… (114)
3.6.8 种子类 …… (114)
3.6.9 派生类的类型转换 …… (116)
3.6.10 作用域的分辨 …… (117)

3.6.11 性质扩展 …… (120)
3.6.12 性质约束 …… (122)
3.6.13 使用单一继承的例子 …… (124)
3.6.14 函数闭包 …… (126)
3.7 多重继承 …… (131)
3.7.1 声明多基类继承的类 …… (133)
3.7.2 调用基类构造函数 …… (133)
3.7.3 使用虚基类 …… (134)
3.7.4 混合使用虚基类和非虚基类 …… (135)
3.7.5 调用析构函数 …… (136)
3.7.6 使用类型转换 …… (136)
3.7.7 保持基类函数的正确性 …… (137)
3.7.8 多继承中作用域分辨的应用 …… (139)
3.7.9 跟踪内存 …… (140)
3.8 提高部分 …… (141)
3.8.1 运行时刻的考虑 …… (141)
3.8.2 进入对象内部 …… (142)
3.8.3 被继承的 Debugger 类 …… (144)

第 4 章 重载 …… (149)
4.1 重载的原因 …… (149)
4.2 函数重载 …… (150)
4.2.1 非成员重载函数 …… (150)
4.2.2 重载成员函数 …… (152)
4.2.3 类等级中的重载函数 …… (152)
4.2.4 重载不是覆盖 …… (154)
4.2.5 作用域分辨 …… (154)
4.2.6 参数匹配 …… (155)
4.2.7 重载构造函数 …… (155)
4.2.8 一些特殊情况 …… (157)
4.2.9 通过重载定义用户转换规则 …… (158)
4.2.10 重载静态成员函数 …… (161)
4.3 操作符重载 …… (162)
4.3.1 操作符用作函数调用 …… (163)
4.3.2 重载操作符用作成员函数 …… (164)
4.3.3 操作符成员函数的几个注意点 …… (166)
4.3.4 重载操作符用作友元函数 …… (166)
4.3.5 赋值操作符 …… (168)

4.3.6 函数调用操作符() …… (170)
4.3.7 下标操作符 …… (172)
4.3.8 操作符重载限制 …… (173)
4.3.9 操作符的作用域分辨 …… (174)
4.4 提高部分 …… (175)
4.4.1 名字分裂的规则 …… (175)
4.4.2 重载 new 和 delete …… (177)
4.4.3 前缀和后缀操作符 …… (180)

第5章 多态性 …… (182)
5.1 先期和迟后联编 …… (182)
5.2 C++是一种混合语言 …… (183)
5.3 虚函数 …… (183)
5.3.1 函数覆盖 …… (184)
5.3.2 改善了的类用户界面 …… (186)
5.3.3 抽象类 …… (186)
5.3.4 虚函数的局限性 …… (189)
5.3.5 虚友元 …… (189)
5.3.6 虚操作符 …… (190)
5.4 多态性的例子 …… (193)
5.5 作用域分辨使多态性失效 …… (196)
5.6 虚函数与非虚函数连用 …… (197)
5.7 vptr 和 vtab 结构的内存布局 …… (198)
5.8 虚函数可以不被覆盖 …… (198)
5.9 确定是否使用虚函数 …… (200)
5.10 私有虚函数 …… (201)
5.11 提高部分 …… (202)
5.11.1 多态性机制 …… (202)
5.11.2 单一继承中的多态性 …… (203)
5.11.3 多重继承中的多态性 …… (207)
5.11.4 内联虚函数 …… (210)
5.11.5 基类中调用多态函数 …… (213)
5.11.6 虚函数和分类等级 …… (215)
5.11.7 构造函数中调用虚函数 …… (217)

第6章 例外处理 …… (219)
6.1 处理例外的旧方法 …… (220)
6.2 处理例外的 OOP 方法 …… (221)

6.3 扔掉例外 …… (221)
6.4 扔掉初始化过的对象 …… (222)
6.5 捕获例外 …… (224)
6.6 捕获无类型的例外 …… (225)
6.7 使用多个 catch 块 …… (226)
6.8 使用 catch/throw 取代 setjmp/longjmp …… (227)
6.9 寻找合适的例外处理程序 …… (227)
6.10 清除栈 …… (227)
6.11 把例外当作类对象 …… (229)
6.12 通过引用处理例外 …… (229)
6.13 例外类型的层次 …… (230)
6.14 用例外处理常见错误 …… (232)
6.15 例外和资源获取 …… (232)
6.16 使用函数闭包 …… (233)
6.17 例外和构造函数 …… (235)
6.18 含子对象的对象中的例外 …… (236)
6.19 标准 C++例外 …… (237)
6.20 OWL 中的错误处理 …… (240)
6.21 TXOwl 例外 …… (241)
6.22 TXCompatibility 例外 …… (241)
6.23 TXGdi 例外 …… (242)
6.24 TXInvalidMainWindow 例外 …… (242)
6.25 TXInvalidModule 例外 …… (243)
6.26 TXMenu 例外 …… (243)
6.27 TXOutOfMemory 例外 …… (243)
6.28 TXPrinter 例外 …… (245)
6.29 TXValidator 例外 …… (245)
6.30 TXWindow 例外 …… (246)
6.30.1 在 Windows 编程中的一些声明 …… (247)
6.30.2 例外处理和回调函数 …… (247)
6.31 多线程程序中的例外处理 …… (248)
6.32 例外接口规范 …… (248)
6.33 函数 unexpected() …… (251)
6.34 terminate()函数 …… (252)
6.35 跟踪对 terminate()的调用 …… (253)

第 7 章 流 …… (255)
7.1 stdio 方法的缺点 …… (255)

7.2 C++流 …… (256)
7.3 广义的流 …… (257)
7.4 内部数据类型的标准I/O流 …… (258)
7.4.1 char 和 char * 类型的I/O操作 …… (259)
7.4.2 int 和 long 类型的I/O操作 …… (260)
7.4.3 float 和 double 类型的I/O操作 …… (261)
7.4.4 用户类的I/O操作 …… (262)
7.5 操作函数 …… (264)
7.5.1 使用数制(基)操作函数 …… (266)
7.5.2 设置和清除格式标志 …… (267)
7.5.3 改变域宽及填充 …… (267)
7.5.4 使用格式操作函数 …… (268)
7.5.5 用户定义的操作函数 …… (270)
7.6 文件I/O的流实观 …… (279)
7.6.1 文本文件输入 …… (279)
7.6.2 流的错误检测 …… (280)
7.6.3 文本文件输出 …… (282)
7.6.4 二进制文件输入 …… (284)
7.6.5 二进制文件输出 …… (286)
7.6.6 拷贝文件 …… (289)
7.7 内存格式化 …… (291)
7.8 将打印机看作流 …… (293)
7.9 提高部分 …… (294)
7.10 streambuf 等级 …… (295)
7.11 ios 等级 …… (296)
7.11.1 类 filebuf …… (296)
7.11.2 类 fstream …… (301)
7.11.3 类 fstreambase …… (304)
7.11.4 类 ifstream …… (307)
7.11.5 类 ios …… (310)
7.11.6 类 iostream …… (322)
7.11.7 类 iostream_withassign …… (325)
7.11.8 类 istream …… (325)
7.11.9 类 istream_withassign …… (332)
7.11.10 类 istrstream …… (333)
7.11.11 类 ofstream …… (335)
7.11.12 类 ostream …… (338)
7.11.13 类 ostream_withassign …… (342)

7.11.14 类 ostrstream …… (344)
7.11.15 类 streambuf …… (349)
7.11.16 从类 streambuf 中派生类 …… (356)
7.11.17 类 strstream …… (359)
7.11.18 使用类 strstream …… (360)
7.11.19 类 strstreambase …… (362)
7.11.20 类 strstreambuf …… (364)

第 8 章 基于对象的包容类库 …… (371)
8.1 类的分类 …… (371)
8.2 AbstractArray(抽象数组)类 …… (373)
8.3 Array 类 …… (378)
8.4 Array 类的使用 …… (379)
8.5 数组中某一存储位置的再次使用 …… (380)
8.6 Association(关联)类 …… (381)
8.7 关联所用到的对象的定义 …… (383)
8.8 Association 类的使用 …… (384)
8.9 由 Association 派生出类 …… (386)
8.10 Bag(包)类 …… (388)
8.11 BaseDate(基日期)类 …… (391)
8.12 BaseTime(时基)类 …… (394)
8.13 Btree(B 树)类 …… (397)
8.13.1 树的基本概念 …… (397)
8.13.2 二叉树 …… (398)
8.13.3 二叉树的性能评价 …… (399)
8.13.4 B 树 …… (399)
8.14 Collection(集)类 …… (412)
8.15 Container(包容)类 …… (414)
8.16 Deque(双端队列)类 …… (419)
8.17 Dictionary(字典)类 …… (422)
8.17.1 一个 Dictionary 的例子 …… (423)
8.17.2 用外部循环量遍历 Dictionary 包容 …… (426)
8.18 DoubleList(双向链表)类 …… (426)
8.19 Error 类 …… (431)
8.20 HashTable(哈希表)类 …… (434)
8.21 List(链表)类 …… (442)
8.22 Object 类 …… (446)
8.23 PriorityQueue(优先队列)类 …… (450)

8.23.1 PriorityQueue 类的使用 …… (452)
8.23.2 把优先队列转换成 GIFO 队列 …… (455)
8.24 Queue(队列)类 …… (456)
8.24.1 Set(集合)类 …… (459)
8.24.2 处理字符串的 Set 类 …… (460)
8.25 一个更接近数学意义上的集合的 Set 类 …… (462)
8.26 Sortable(排序)类 …… (464)
8.27 SortedArray(排序数组)类 …… (466)
8.28 Stack(栈)类 …… (468)
8.29 String(字符串)类 …… (471)
8.30 String 类的使用 …… (474)
8.30.1 从 String 类派生出新的类 …… (476)
8.30.2 Time(时间)类 …… (478)
8.31 Time 类的使用 …… (479)
8.32 由 Time 类派生出新类 …… (481)
8.33 循环量 …… (484)

第 9 章 基于模板的包容类库 …… (487)
9.1 FDS 和 ADT 包容 …… (487)
9.2 FDS 包容 …… (487)
9.3 FDS 存储范例 …… (488)
9.4 FDS 包容的使用 …… (488)
9.5 FDS 向量包容 …… (488)
9.5.1 简单直接向量 …… (489)
9.5.2 直接计数向量 …… (491)
9.5.3 直接排序向量 …… (492)
9.5.4 间接简单向量 …… (495)
9.5.5 间接排序向量 …… (497)
9.6 FDS 表包容 …… (499)
9.6.1 直接简单表 …… (499)
9.6.2 直接排序表 …… (501)
9.6.3 间接表 …… (505)
9.6.4 间接排序表 …… (505)
9.7 ADT 包容 …… (506)
9.8 用包容类进行内存管理 …… (507)
9.9 ADT 数组 …… (508)
9.10 ADT 排序数组 …… (512)
9.11 ADT 栈 …… (514)

9.12 ADT 直接栈 …… (514)
9.13 ADT 间接栈 …… (517)
9.14 ADT 队列和双端队列 …… (518)
9.15 ADT 包和集合 …… (522)
9.16 异质 ADT 包容 …… (526)

第二部分 OWL

第 10 章 ObjectWindows(对象窗口)库类 …… (530)
10.1 使用 OWL 的调试版本 …… (530)
10.2 OWL 应用程序的入口点 …… (530)
10.3 定制主窗口 …… (532)
10.4 OWL 对话框 …… (535)
10.4.1 选择对话框风格 …… (535)
10.4.2 灰色 3D 对话框 …… (536)
10.4.3 Borland 风格对话框 …… (537)
10.4.4 对话框中的位图 …… (537)
10.4.5 位图式子控制的 OWL 计数表 …… (539)
10.4.6 读写对话框数据 …… (539)
10.5 OWL 子控制 …… (544)
10.6 数据合法性 …… (545)
10.7 合法检验函数选项 …… (545)
10.7.1 使用 TFilterValidator …… (546)
10.7.2 使用 TRangeValidator …… (547)
10.7.3 使用 TPXPictureValidator …… (547)
10.7.4 使用 TStringLookupValidator …… (548)
10.8 定制控制 …… (549)
10.8.1 使用 BWCC 定制控制 …… (549)
10.8.2 雕像 …… (550)
10.8.3 自画式控制 …… (551)
10.8.4 自画式圆按钮 …… (557)

第 11 章 MDI 应用程序 …… (564)
11.1 窗口层次 …… (566)
11.2 定制 MDI 用户区域 …… (567)
11.3 框架和用户区域刷新消息 …… (567)
11.4 预定义 Windows 类属性的修改 …… (568)
11.5 刷新纯背景颜色 …… (570)

11.6 刷新闪烁的背景颜色 …… (572)
11.7 拖动并安置 …… (574)
11.8 键盘处理 …… (576)
11.9 主消息循环 …… (577)
11.10 函数 TApplication::ProcessAppMessage() …… (578)
11.11 允许键盘处理的窗口 …… (579)
11.12 带子控制的 MDI 子窗口 …… (580)
11.13 改变 MDI 子窗口的背景 …… (580)
11.14 MDI 子窗口中子控制的处理 …… (584)
11.15 子控制的两个构造函数 …… (584)
11.16 两个构造函数间的区别 …… (584)
11.17 一个完整的例子 …… (585)
11.18 用对话框作 MDI 子窗口 …… (590)
11.19 MDI 应用程序中的菜单 …… (595)
11.20 替换主菜单 …… (595)
11.21 合并菜单 …… (599)
11.22 OLE 2.0 菜单合并 …… (599)
11.23 OWL 如何合并菜单 …… (600)
11.24 改变菜单项的状态 …… (604)
11.25 命令使能 …… (605)
11.26 增添菜单命令的检取标记 …… (606)

第 12 章 列表框的变化 …… (608)
12.1 制表点 …… (608)
12.2 设置制表点 …… (609)
12.3 范例 12.1 …… (609)
12.4 制表点越界 …… (612)
12.5 用多列表框模拟制表点 …… (613)
12.5.1 移动列表框选择条 …… (614)
12.5.2 范例 12.2 …… (615)
12.5.3 用户可定制的显示格式 …… (623)
12.5.4 改变列宽 …… (623)
12.5.5 重排 …… (624)
12.5.6 删除多余的列 …… (624)
12.5.7 记录字符串长度 …… (627)
12.5.8 向列表框增加字符串 …… (627)
12.5.9 删除列表框中字符串 …… (627)
12.5.10 清空列表框 …… (628)