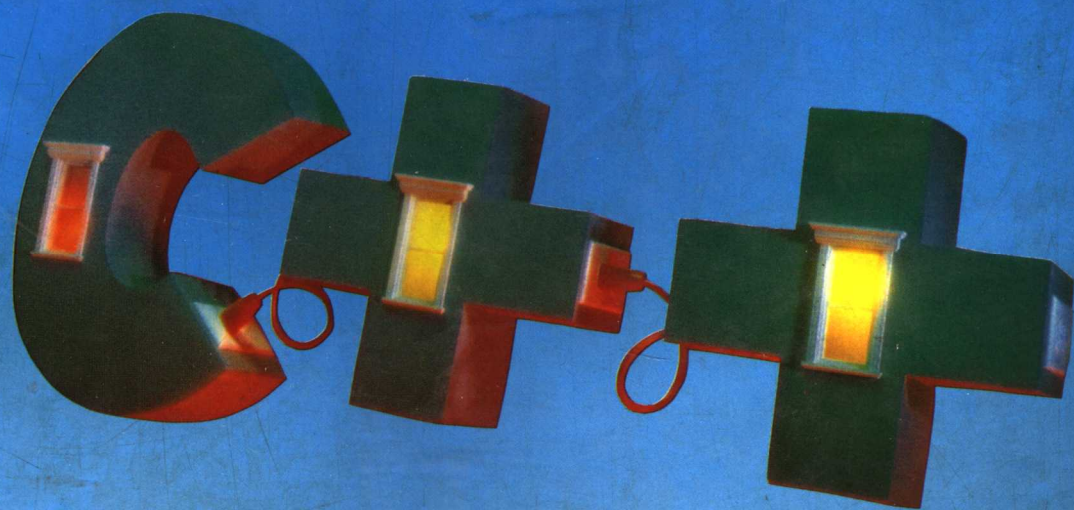


Borland C++3.1

编程指南

[美] Ted Faison 著
蒋维杜 吴志美
张新宇 李景淑 译
李 莉 审校



清华大学出版社

Borland C++ 3.1 编程指南

[美] Ted Faison 著

蒋维杜 吴志美

张新宇 李景淑 译

李 莉 审校

清华大学出版社

Borland C++ 3.1 Object-Oriented Programming

(Second Edition)

Ted Faison

Authorized translation from the English language edition published by Sams.

Copyright ©1992 by Sams.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission in writing from the Publisher.

Chinese language edition published by Tsinghua University Press.

Copyright ©1993 by Tsinghua University Press.

本书英文版由 Prentice Hall 出版社属下的 Sams 计算机图书出版公司于 1992 年出版。版权为 Sams 所有。Sams 将本书的中文版专有出版权授予清华大学出版社和北京科海培训中心。未经出版者书面允许不得以任何方式复制或抄袭本书内容。

(京)新登字 158 号

Borland C++ 3.1 编程指南

[美] Ted Faison 著

蒋维杜 吴志美

张新宇 李景淑 译

李 莉 审校

☆

清华大学出版社出版

北京 清华园

门头沟胶印厂印刷

新华书店总店科技发行所发行

☆

开本: 787×1092 1/16 印张: 50 字数: 1216 千字

1993 年 12 月第 1 版 1993 年 12 月第 1 次印刷

印数: 0001~5000 册

ISBN 7-302-01442-6/TP·565

定价: 65.00 元

译者序

面向对象的编程法(OOP)是当前的热门课题。对许多人来讲,C++本身就是OOP的同义词,就象许多人下意识地将LISP编程法与人工智能联系起来一样。事实上,OOP并不是这种或那种语言的结果,而是其所用的特殊方法的结果。完全可能用Pascal、Ada、BASIC、汇编语言之类的语言开发OOP应用程序,尽管会更困难些。

本书以特殊的开发工具——Borland C++阐述了C++语言的OOP特点。由于当前Microsoft Windows编程方法正受到越来越多的关注,本书也涉及到这方面的有关问题,但只限于OOP框架内。读者将学会用OOP思想理解和思考Windows应用程序,使用与Microsoft Windows软件开发工具这种传统方法不同的开发方法。

本书是一本实用手册,因为其中有许多程序、工程实例。所有的例子程序和有编号的代码段都可以直接装入并编译,以验证要说明的各种不同的OOP性质。本书很大程度上考虑了功能和效率这些实际问题,读者应该具备C语言编程的经验,而且最少要有两年经验。有经验的用户将在每章尾部发现其感兴趣的资料。

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优点加入 C 语言, 同时保留了使得 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++ 能带来的好处, 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。

0.1 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。一个对象是一组代码和数据的集合, 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 Smalltalk) 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关系, 而不是实现的细节上。关系是对象之间联系的纽带, 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联系上, 而不是对象行为的实现过程上。这种区别是很重要的, 它表明了与早期的“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。设计出一组组相互联系的对象, 就能使 OOP 的作用大大发挥出来。这些组通常称为类的等级。设计类的等级是 OOP 的中心。

0.2 本书结构

本书描述了基于 Borland C++3.1,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,“用 Borland C++进行面向对象的程序设计”,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,只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,“开发 Windows 和 DOS 应用程序”,该部分从第 8 章开始,展示了怎样使用 Borland C++的类库和实用开发框架。90 年代以来,人们认为应该用图形界面代替其他任何界面,而 Windows 是 DOS 机器上最主要的图形用户界面。预见到这点,Borland 公司已为其 C++3.0 编译器开发了两种实用框架,以便于书写 Windows 或 DOS 程序。全书中大量的内容放在例程上(配套磁盘上有源代码)。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。

0.3 本书的描述

本书从总体上论述 C++编程方法,侧重于 Borland C++3.1。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正式的定义或说明,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,增加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”,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,包括对其运行时间特点

的解释。

第6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7章,“包容类库”,是Borland C++3.1独有的,不能适用于其他编译器。本章描述了Borland装配的包容类库,并以应用例子说明。该类库对任一程序设计工程都是基本的,必须象学Borland C++一样学习它。类的重用是使C++多产的一个重要特征。本章不仅说明了怎样直接使用包容类库,还介绍了如何利用它们作为基类来建立自定义的类。本章详细介绍了基于对象和基于模板的包容类的情况。

第8章,“用于Windows程序设计的类”,从C++程序员的角度介绍了微软公司的Windows,并假定读者对作为图形用户界面和编程环境的Windows是熟悉的。本章讲述了在Windows环境下设计和开发类的方法,每个类都有简单的示例实用程序(配套磁盘上有相应的代码)。现有的有关Windows程序设计的文献大多只讲述了对C程序员非常有效的技术,对面向对象的程序设计来说还不够。在Borland C++中,用类来简化工作量,并掩藏Windows程序设计通常会遇到的许多困难。

第9章,“完整的窗口程序”,将前述章节的知识连起来,设计编制了一个小的窗口应用程序。许多Windows的结构都有说明,包括消息框、对话模式、写字板界面,使用打印机,装DLLS(动态连接库)等等。大范围地使用对象可使任务相对简单,甚至更受欢迎。

第10章,“对象窗口类库”,从对象窗口库(OWL)着手,研究了Borland实用框架。本章讲述了重用基本OWL类来按要求改制典型Windows程序不同部分的方法。本章采用的方法是低层次的,重点集中在单个类或窗口对象上,而不是应用。还将涉及传统控制、持久对象、图象闪动等方面。为理解本章和下章的内容,需对OWL有基本了解。

第11章,“OWL应用程序”,与第10章相比,采用了高层次的方法进行OWL编程。从OWL类中派生出新类来支持一些一般应用需要,如状态行、弹出菜单、工具板、编辑窗等。从Windows程序设计的观点来看,第10、11章包含了本书中最有意义的内容。

第12章,“Turbo Vision类”,处理另一类Borland实用框架,Turbo Vision(TV)。为改变TV应用程序的基本部分,从TV已有的类中派生出了一低层次的类,如状态行、菜单条、桌面等。本章末尾详细介绍了固定的TV对象。理解本章和第13章,需对TV有基本了解。

第13章,“Turbo Vision应用程序”,使用第12章设计的类和例子建立了不同的TV应用程序。这些应用程序强调了TV的一些重要特点,包括上下文有关帮助、性质检查和编辑窗口。使用本章的指导,可以开发复杂的应用程序,很好地用上鼠标,可以指定颜色,多层重叠的窗口,热键和集成帮助设施。本章还展示了如何为适应需求,通过派生类来改变TV的基本特征,而不必考虑事件管理、图形方式等内在的细节。

0.4 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
- EGA,VGA 或更好的图形适配器
- Borland C++3.1 版本
- Borland 实用框架(用于第 10 至 13 章)
- Windows3.1 或更高版本

第 8 章到第 13 章的示例工程文件是用 Borland C++3.1 版本编写的。如果用以前的版本,也许不能成功地使用这些工作文件。

微软公司用于 Windows 的软件开发的工具(SDK)并不一定需要。如果有 Turbo C++,或 Borland C++ 3.1 版本以前的版本,仍可以编译 1 至 6 章的代码,不过第 7 章的包容类将不能用——除非改变代码。

第 8 到 11 章的 Windows 代码需要 Borland C++3.1。如果用 Borland C++3.0 和 Windows3.0,需要对源代码或工程文件做修改。

目 录

引言.....	(1)
0.1 为什么使用 OOP 方法?	(1)
0.2 本书结构	(2)
0.3 本书的描述	(2)
0.4 软硬件需求	(3)

第一部分 用 Borland C++ 进行面向对象的程序设计

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	(4)
1.2 变量	(8)
1.2.1 作用域	(8)
1.2.2 类型.....	(10)
1.2.3 存储类.....	(10)
1.2.4 const 限定符	(10)
1.2.5 Volatile 限定符	(12)
1.2.6 语句.....	(13)
1.2.7 函数.....	(21)
1.2.8 指针和引用.....	(26)
1.2.9 指针、引用与 const 连用	(28)
1.2.10 提高部分	(29)
第 2 章 对象和类	(41)
2.1 定义类.....	(41)
2.1.1 类标识符.....	(42)
2.1.2 类体.....	(42)
2.2 使用类.....	(43)
2.2.1 封装.....	(43)
2.2.2 类存取控制.....	(44)
2.2.3 类私有成员.....	(45)
2.2.4 类公有成员.....	(46)
2.2.5 类保护成员.....	(46)

2.2.6	类对象的存储类	(47)
2.2.7	类作用域	(47)
2.2.8	空类	(48)
2.2.9	类嵌套	(48)
2.2.10	类的实例化	(50)
2.2.11	不完全的类声明	(50)
2.3	使用数据成员	(51)
2.3.1	静态数据成员	(51)
2.3.2	private static 数据成员	(53)
2.3.3	类对象用作数据成员	(54)
2.3.4	指针数据成员	(56)
2.3.5	指向类数据成员的指针	(56)
2.3.6	指向对象数据成员的指针	(57)
2.4	使用成员函数	(58)
2.4.1	简单成员函数	(59)
2.4.2	静态成员函数	(59)
2.4.3	Const 成员函数	(60)
2.4.4	volatile 成员函数	(61)
2.4.5	内联成员函数	(61)
2.4.6	带有 const this 的成员函数	(62)
2.4.7	带有 volatile this 的成员函数	(63)
2.4.8	特殊类函数	(64)
2.4.9	构造函数	(65)
2.4.10	析构函数	(70)
2.4.11	友元关键字	(72)
2.4.12	友元的性质	(73)
2.5	提高部分	(74)
2.5.1	成员函数指针	(74)
2.5.2	数组和类	(76)
2.5.3	成员函数调用的分析	(81)
2.5.4	类模板	(82)
2.5.5	函数模板	(89)
第3章	继承性	(92)
3.1	可复用性	(92)
3.2	继承性	(92)
3.3	继承的作用	(93)
3.4	C++继承性的局限性	(93)
3.5	关于继承的不同观察角度	(94)

3.6 单一继承	(94)
3.6.1 何时继承	(94)
3.6.2 不能被继承的成分	(95)
3.6.3 基类的存取限定符	(95)
3.6.4 可被继承的类	(96)
3.6.5 传递给基类的参数	(97)
3.6.6 构造函数的调用顺序	(98)
3.6.7 析构函数的调用顺序	(99)
3.6.8 种子类	(99)
3.6.9 派生类的类型转换	(101)
3.6.10 作用域的分辨	(102)
3.6.11 性质扩展	(105)
3.6.12 性质约束	(107)
3.6.13 使用单一继承的例子	(108)
3.6.14 函数闭包	(110)
3.7 多重继承	(114)
3.7.1 声明多基类继承的类	(115)
3.7.2 调用基类构造函数	(116)
3.7.3 使用虚基类	(116)
3.7.4 混合使用虚基类和非虚基类	(118)
3.7.5 调用析构函数	(118)
3.7.6 使用类型转换	(118)
3.7.7 保持基类函数的正确性	(120)
3.7.8 多继承中作用域分辨的应用	(121)
3.7.9 跟踪内存	(122)
3.8 提高部分	(123)
3.8.1 运行时刻的考虑	(123)
3.8.2 进入对象内部	(123)
3.8.3 被继承的 Debugger 类	(126)
第4章 重载	(130)
4.1 重载的原因	(130)
4.2 函数重载	(131)
4.2.1 非成员重载函数	(131)
4.2.2 重载成员函数	(132)
4.2.3 类等级中的重载函数	(133)
4.2.4 重载不是覆盖	(134)
4.2.5 作用域分辨	(134)
4.2.6 参数匹配	(135)

4.2.7	重载构造函数	(136)
4.2.8	一些特殊情况	(137)
4.2.9	通过重载定义用户转换规则	(138)
4.2.10	重载静态成员函数	(141)
4.3	操作符重载	(141)
4.3.1	操作符用作函数调用	(143)
4.3.2	重载操作符用作成员函数	(143)
4.3.3	操作符成员函数的几个注意点	(145)
4.3.4	重载操作符用作友元函数	(145)
4.3.5	赋值操作符	(147)
4.3.6	函数调用操作符()	(149)
4.3.7	下标操作符	(151)
4.3.8	操作符重载限制	(152)
4.3.9	操作符的作用域分辨	(152)
4.4	提高部分	(153)
4.4.1	名字分裂的规则	(153)
4.4.2	重载 new 和 delete	(156)
4.4.3	前缀和后缀操作符	(158)

第5章 多态性..... (161)

5.1	先期和迟后联编	(161)
5.2	C++是一种混合语言	(162)
5.3	虚函数	(162)
5.3.1	函数覆盖	(163)
5.3.2	空的虚函数	(164)
5.3.3	改善了的类用户接口	(165)
5.3.4	抽象类	(166)
5.3.5	虚函数的局限性	(169)
5.3.6	虚友元	(169)
5.3.7	虚操作符	(170)
5.3.8	虚构造函数	(172)
5.3.9	虚析构函数	(172)
5.4	多态性的例子	(172)
5.5	作用域分辨使多态性失效	(175)
5.6	虚函数和非虚函数连用	(176)
5.7	vptr 和 vtab 结构的内存布局	(177)
5.8	虚函数可以不被覆盖	(178)
5.9	确定是否使用虚函数	(179)
5.10	私有虚函数	(180)

5.11 提高部分.....	(182)
5.11.1 多态性机制.....	(182)
5.11.2 单一继承中的多态性.....	(182)
5.11.3 多重继承中的多态性.....	(186)
5.11.4 嵌入式虚函数.....	(189)
5.11.5 基类中调用多态函数.....	(192)
5.11.6 虚函数和分类等级.....	(194)
5.11.7 构造函数中调用虚函数.....	(196)
第6章 流.....	(198)
6.1 Stdio 方法的缺点.....	(198)
6.2 C++流	(199)
6.3 广义的流	(199)
6.4 内部类型的标准 I/O 流	(200)
6.4.1 char 和 char * 类型的 I/O 操作	(202)
6.4.2 int 和 long 类型的 I/O 操作	(203)
6.4.3 float 和 double 类型的 I/O 操作	(204)
6.4.4 用户类的 I/O 操作	(204)
6.5 操作函数	(206)
6.5.1 使用数制(基)操作函数	(208)
6.5.2 设置和清除格式标志	(209)
6.5.3 改变域宽及填充	(209)
6.5.4 使用格式操作函数	(210)
6.6 文件 I/O 的流实现	(211)
6.6.1 文本文件输入	(212)
6.6.2 流的错误检测	(213)
6.6.3 文本文件输出	(214)
6.6.4 二进制文件输入	(216)
6.6.5 二进制文件输出	(217)
6.7 内存格式化	(218)
6.8 将打印机看作流	(220)
6.9 提高部分	(221)
6.10 内部流类型.....	(221)
6.11 streambuf 等级	(221)
6.11.1 类 streambuf	(222)
6.11.2 从类 streambuf 中派生类	(229)
6.11.3 类 strstreambuf	(232)
6.11.4 类 filebuf	(237)
6.12 ios 等级	(242)

6.12.1	类 ios	(243)
6.12.2	类 istream	(254)
6.12.3	类 ostream	(260)
6.12.4	类 iostream	(263)
6.12.5	类 istream.withassign	(266)
6.12.6	类 ostream.withassign	(268)
6.12.7	类 iostream.withassign	(269)
6.12.8	类 fstreambase	(270)
6.12.9	类 strstreambase	(273)
6.12.10	类 ifstream	(275)
6.12.11	类 ofstream	(277)
6.12.12	类 fstream	(280)
6.12.13	类 istrstream	(282)
6.12.14	类 ostrstream	(285)
6.12.15	类 strstream	(289)
6.12.16	流中二进制文件操作和文本文件操作	(291)
6.12.17	用户定义的操作函数	(296)
6.13	流代码的大小	(303)

第7章 包容类库..... (304)

7.1	类的等级结构的优点	(304)
7.2	类层次结构的目标	(306)
7.3	包容类	(306)
7.3.1	类的分类	(306)
7.3.2	程序运行时对类的识别	(308)
7.4	基于 Object 类的包容类	(308)
7.5	类 AbstractArray	(309)
7.6	Array 类	(313)
7.6.1	Array 类的使用	(314)
7.6.2	数组中某一存储位置的再次使用	(315)
7.7	Association(关联)类	(316)
7.7.1	关联所用到的对象的定义	(318)
7.7.2	Association 类的使用	(319)
7.7.3	从 Association 派生类	(321)
7.8	类 Bag	(322)
7.9	类 BaseDate	(326)
7.10	类 BaseTime	(329)
7.11	类 Btree	(331)
7.11.1	树的基本概念	(331)

7.11.2	二叉树	(332)
7.11.3	B 树	(333)
7.12	类 Collection	(345)
7.13	类 Container	(347)
7.14	类 Date	(352)
7.15	类 Deque	(358)
7.16	类 Dictionary	(360)
7.16.1	一个 Dictionary 的例子	(361)
7.16.2	用外部循环量遍历 Dictionary 包容	(363)
7.17	类 DoubleList	(364)
7.18	类 Error	(369)
7.19	类 HashTable	(371)
7.20	类 List	(378)
7.21	类 Object	(382)
7.22	类 PriorityQueue	(385)
7.22.1	类 PriorityQueue 的使用	(387)
7.22.2	把优先队列转换成 GIFO 队列	(390)
7.23	类 Queue	(390)
7.24	类 Set	(394)
7.24.1	处理字符串的类 Set	(394)
7.24.2	一个更接近数学意义上的集合的类 Set	(396)
7.25	类 Sortable	(398)
7.26	类 SortedArray	(400)
7.27	类 Stack	(402)
7.28	类 String	(404)
7.28.1	类 String 的使用	(407)
7.28.2	从类 String 派生出新的类	(408)
7.29	类 Time	(410)
7.29.1	类 Time 的使用	(411)
7.29.2	由类 Time 派生出新类	(413)
7.30	类 Timer	(416)
7.31	类 TShouldDelete	(419)
7.32	循环量	(420)
7.33	构造类库	(422)
7.34	基于模板的包容类	(422)
7.35	FDS 和 ADT 包容	(423)
7.36	FDS 包容	(423)
7.36.1	FDS 存储范例	(423)
7.36.2	FDS 包容	(424)

7.36.3	FDS 向量包容	(424)
7.36.4	简单直接向量	(426)
7.36.5	直接计数向量	(426)
7.36.6	直接排序向量	(427)
7.36.7	间接简单向量	(429)
7.36.8	间接计数向量	(430)
7.36.9	间接排序向量	(431)
7.37	FDS 表包容	(433)
7.37.1	直接简单表	(435)
7.37.2	直接排序表	(435)
7.37.3	间接表	(438)
7.37.4	间接排序表	(439)
7.38	ADT 包容	(439)
7.38.1	ADT 数组	(440)
7.38.2	ADT 排序数组	(444)
7.38.3	ADT 栈	(445)
7.38.4	ADT 队列和双端队列	(449)
7.38.5	ADT 包和集合	(452)
7.38.6	异质 ADT 包容	(456)

第二部分 开发 Windows 和 DOS 应用程序

第 8 章	用于 Windows 程序设计的类	(459)
8.1	运行检测程序	(459)
8.2	最终为 PC 机提供标准 GUI	(460)
8.3	围绕对象设计的 Windows	(460)
8.4	C++ 中对 Windows 资源的管理	(461)
8.5	设备描述表	(461)
8.6	绘图闭包	(471)
8.7	对话框	(476)
8.8	位图	(485)
8.9	裁剪板	(488)
8.10	真正的出发点	(494)

第 9 章	完整的窗口程序	(495)
9.1	文件对话框	(495)
9.1.1	文件打开用的对话框	(501)
9.1.2	文件存储对话框	(504)
9.2	使用打印机	(510)

9.2.1 使用 DLL	(510)
9.2.2 Printer Driver DLL	(512)
9.2.3 Printer 类	(514)
9.3 文本编辑器	(520)
9.3.1 主菜单	(520)
9.3.2 实现	(521)
9.3.3 About 对话框	(526)
第 10 章 对象窗口类库	(528)
10.1 忘记 ANSI C 风格的 Windows 程序	(528)
10.2 Borland“所见即所得”	(528)
10.3 OWL 应用程序的解析	(529)
10.3.1 定制主窗口	(533)
10.3.2 对话框处理	(535)
10.3.3 增加菜单	(537)
10.4 无模式对话框	(538)
10.4.1 带色斑的对话框	(541)
10.4.2 全部数字式	(543)
10.4.3 对话框数据的读写	(544)
10.4.4 子控制框的数据合法性检查	(550)
10.5 自画式控制	(558)
10.5.1 BWCC 的自画式控制	(558)
10.5.2 更多的数字	(558)
10.5.3 非 BWCC 的自画式控制	(561)
10.5.4 自画式圆按钮	(567)
10.6 另外一个文件打开对话框	(573)
10.7 持续的 OWL 对象	(582)
第 11 章 OWL 应用程序	(589)
11.1 OWL 应用程序的概貌	(589)
11.2 状态行	(589)
11.2.1 子窗口方式	(590)
11.2.2 GDI 方式	(597)
11.3 弹出式菜单	(606)
11.3.1 菜单的处理	(607)
11.3.2 一个完整的应用程序:WPOPUP	(608)
11.4 浮动式工具调色板	(614)
11.4.1 一般的画图工具	(615)
11.4.2 专用的画图工具	(616)