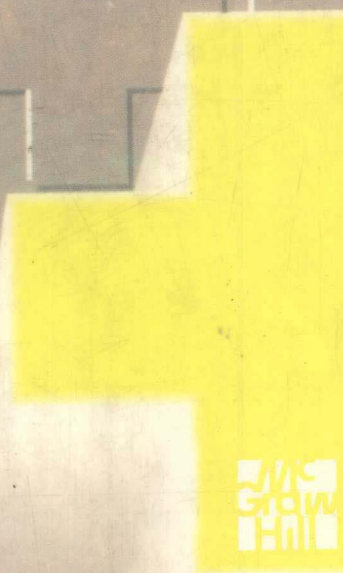
计算机语言技术系列丛书(二)

学苑出版社

WINDOWS™ PROGRAMMING WITH BORLAND C++

程序设计

- Covers Windows™ 3.1, Windows NT™, & Win32s™
- Disk includes a professional-quality Windows class library



THE Len Dorfman PRACTICAL PROGRAMMING SERIES

Jeff Mackay

希望

Windows Programming With Borland C++

BORLAND C++ Windows 程序设计

〔美〕Jeff Mackay 著
张 如 罗继波 译
燕卫华 审校

学苑出版社

(京)新登字 151 号

内 容 提 要

设计和构造具有 Windows 界面优点的功能强大的应用程序，使用本书将使它变得容易。本书作者给出了使用 Vista 类库开发面向对象应用的技术。此外，还提供了大量 C++源代码(包含在磁盘中)，它告诉用户如何做下面事情：

- 使用 Borland C++和 Turbo C++ for Windows 工具开发 Vista 应用程序
- 使用窗口和图形对象
- 用选单和加速键获取输入
- 创建对话框、消息框和选单
- 开发子窗口控制对象和客户对话框
- 管理多文档接口(MDI)窗口对象
- 使用 Windows 内存管理和动态连接库
- 使用剪贴板，动态数据交换和对象连接和嵌入

使用本书，编写 Windows 应用将更快！

欲购本书者，请与北京市海淀 8721 信箱书刊部联系，邮政编码 100080，**电话** 2562329。

版 权 声 明

计算机语言技术系列丛书(二)

BORLAND C++ Windows 程序设计

著　　者：〔美〕Jeff Mackay
译　　者：张　如　罗继波
审　　校：燕卫华
责任编辑：甄国宪
出版发行：学苑出版社　　**邮政编码**：100036
社　　址：北京海淀区万寿路西街 11 号
印　　刷：双青印刷厂
开　　本：787×1092　1/16
印　　张：38.375　　**字 数**：897 千字
印　　数：1～5000 册
版　　次：1994 年 9 月北京第 1 版第 1 次
ISBN7－5077－0905－1/TP・29
本册定价：59.00 元

关于作者

Laura Acklen 居住在 Texas 州的 Austin，是一个独立的作者和讲师。自从 1986 年以来，她一直从事 DOS 和 Windows 产品的培训和技术支持工作。Laura 已经编写了超过 15 本的学生手册和国家培训公司 Productivity Point International 的教师用书。她是 QUE.《WordPerfect 6.0 SureStep》的作者，也是《Oops! WordPerfect... What To Do When Things Go Wrong》和《Windows QuickStart，3.11 Edition》的作者之一。她还是《Using WordPerfect Version 6 for Windows，Special Edition》的作者。

致　谢

首先，我要感谢 Robert Mullen 的帮助，他编写了第七章和第九章，并最后阅读了第六、八、十和十一章。Robert 是一个作者和自由“骑士”，他至少参与编写了 15 本关于初、中级 PC 用户的书。Robert 最近出版了为家用计算机编写的名为《Choosing and Using Your First CD-ROM Drive》的书，是 Ron Person 编写的 Que 出版的《Special Edition Using Windows 95》的主要合伙人。

此外，还要感谢责任编辑 Fred Slone，正是他协调了此书的合作伙伴，并感谢生产部门的专家 Bryan Gambrel，正是他使本书印制精美，可读性好。最后，感谢 Susan Dunn 生产编辑，以及其他生产部门的人员，他(她)们的努力工作，使得本书成为一本优秀的计算机图书。

在个人方面，我要感谢我的丈夫 Jeff，正是他在整个过程中给与了支持。他是我编辑、吵架和埋怨的主要对象。虽然听起来有点愚，但我还是要说没有他的帮助我真的不能完成本书。

目 录

第一章 面向对象的 Windows 程序 …… (1)
1.1 面向对象系统 …… (1)
1.2 Windows 对象模型 …… (1)
1.2.1 图形对象 …… (2)
1.2.2 窗口对象 …… (2)
1.2.3 事件与消息 …… (2)
1.2.4 Windows 是面向对象的吗? …… (3)
1.2.5 面向对象的 WindowsC++编程 …… (3)
1.3 为什么使用 C++ …… (3)
1.3.1. 一种更好的 C …… (4)
1.3.2 面向对象的扩展 …… (4)
1.3.3 类库 …… (5)
1.3.4 应用程序框架 …… (5)
1.3.5 Windows 程序转换到 C++ …… (5)
1.4 Vista 应用程序框架 …… (6)
1.4.1 应用程序支持类 …… (6)
1.4.2 窗口类 …… (6)
1.4.3 图形类 …… (8)
1.4.4 通信类 …… (8)
1.5 下一步做什么 …… (9)

第二章 Windows 应用程序 …… (10)
2.1 传统的 Windows 程序 …… (10)
2.1.1 程序 …… (10)
2.1.2 主驱动程序:WinMain …… (14)
2.1.3 程序初始化 …… (15)
2.1.4 创建窗口类 …… (15)
2.1.5 创建或显示一个窗口 …… (16)
2.1.6 消息和消息循环 …… (16)
2.1.7 窗口过程 …… (17)
2.1.8 清除(Cleaning up) …… (17)
2.1.9 Windows 命名约定 …… (18)
2.1.10 严格编制程序规范 …… (18)
2.1.11 新数据类型 …… (18)
2.2 一个 Vista Windows 应用程序 …… (19)

2.2.1 程序 …… (22)
2.2.2 创建一个应用程序对象 …… (22)
2.2.3 创建窗口类 …… (23)
2.2.4 创建动作方法 …… (23)
2.2.5 使用图形画对象 …… (24)
2.2.6 Vista 命名转换 …… (24)
2.3 下一步做什么 …… (26)

第三章 建立程序 …… (27)
3.1 Windows 开发周期 …… (27)
3.1.1 创建 C++源文件 …… (27)
3.1.2 创建资源文件 …… (28)
3.1.3 创建模块定义文件 …… (30)
3.1.4 创建帮助文件 …… (30)
3.1.5 创建项目 …… (31)
3.2 建立 WinVista 和 WinApp …… (31)
3.3 调试程序 …… (31)
3.3.1 使用 Turbo Debugger …… (32)
3.3.2 使用 WInspector …… (32)
3.3.3 强有力的测试 …… (32)
3.4 下一步做什么 …… (33)

第四章 应用程序和事件 …… (34)
4.1 Vista 应用程序与 VAppl 类 …… (34)
4.1.1 接口、实现和访问控制 …… (34)
4.1.2 继承和初始化 …… (39)
4.1.3 程序清除 …… (39)
4.1.4 运行程序 …… (40)
4.1.5 重载运算符 …… (40)
4.2 Sizer:使用应用程序类 …… (40)
4.2.1 获取窗口状态信息 …… (40)
4.2.2 维护配置数据 …… (41)
4.3 用 Windows 通信:VEvent 类 …… (42)
4.3.1 Windows 消息 …… (42)
4.3.2 VEvernt 类 …… (47)
4.3.3 事件移植性 …… (47)
4.4 MSGLIST:响应事件 …… (49)
4.4.1 初始化事件调度程序 …… (54)
4.4.2 初始化窗口 …… (54)

4.4.3 画窗口 …… (55)
4.4.4 响应鼠标事件 …… (56)
4.5 下一步做什么 …… (58)

第五章 Vista 窗口对象 …… (59)
5.1 VWindow：基本的窗口类 …… (59)
5.1.1 窗口过程 …… (60)
5.1.2 调度消息 …… (61)
5.1.3 两步初始化 …… (63)
5.1.4 VWindows 类层次 …… (63)
5.2 FileView：一个文件浏览器 …… (63)
5.2.1 设计 FileView …… (63)
5.2.2 FileView 类 …… (64)
5.2.3 Viewer 类 …… (68)
5.2.4 FileBuffer 和 TextLine 类 …… (72)
5.3 滚动窗口 …… (75)
5.4 把 C 程序转移到 Vista …… (81)
5.5 下一步做什么 …… (82)

第六章 用菜单和加速键获得输入 …… (83)
6.1 Windows 菜单 …… (83)
6.1.1 下拉式菜单 …… (83)
6.1.2 层叠式菜单 …… (85)
6.1.3 弹出式菜单 …… (86)
6.2 使用 Vista 菜单对象 …… (87)
6.2.1 为窗口对象增加菜单 …… (90)
6.2.2 响应用户命令 …… (91)
6.3 用 VMenu 类创建动态菜单 …… (92)
6.4 创建浮动菜单 …… (99)
6.5 使用加速键 …… (104)
6.6 C 程序移植到 Vista …… (109)
6.7 下一步做什么 …… (109)

第七章 对话和消息框 …… (110)
7.1 Vista 对话框 …… (110)
7.2 消息框 …… (110)
7.3 Windows 通用对话框 …… (112)
7.3.1 用 VFileOpenDialog 类打开文件 …… (121)
7.3.2 管理打印机 …… (122)

7.3.3 用 VFontDialog 类选择字体 …… (125)
7.3.4 用 VColorDialog 类设置颜色 …… (126)
7.3.5 用 VFindDialog 类查找文字 …… (127)
7.4 下一步做什么 …… (129)

第八章 子窗口控制对象 …… (130)
8.1 子窗口控制 …… (130)
8.2 把数据对象放入对话框 …… (130)
8.3 用控制的数据验证 …… (146)
8.3.1 字符级验证 …… (146)
8.3.2 字段级验证 …… (147)
8.4 定做控制 …… (147)
8.5 使用全局窗口消息 …… (148)
8.6 移植 C 程序到 Vista …… (149)
8.7 下一步做什么 …… (149)

第九章 MDI 窗口对象 …… (150)
9.1 多文档界面 …… (150)
9.2 Vista MDI 类 …… (150)
9.2.1 管理子窗口 …… (151)
9.2.2 交换菜单 …… (158)
9.2.3 对象依赖关系 …… (165)
9.3 非常规的 MDI 窗口 …… (166)
9.3.1 拖放操作 …… (173)
9.3.2 模拟程序管理器图标 …… (174)
9.4 一个 MDI FileView(文件浏览器) …… (177)
9.5 把 MDI 程序从 C 移植到 Vista …… (194)
9.6 下一步做什么 …… (195)

第十章 图形对象 …… (196)
10.1 Windows 图形模式 …… (196)
10.1.1 设备描述表 …… (197)
10.1.2 绘图工具 …… (197)
10.1.3 图形原语(graphics Primitives) …… (197)
10.1.4 Win32 的改进 …… (197)
10.2 Vista 图形对象 …… (198)
10.2.1 设备对象 …… (198)
10.2.2 图形工具对象 …… (200)
10.2.3 高级图形对象 …… (201)

10.2.4 打印对象 …… (206)
10.3 下一步做什么 …… (207)

第十一章 内存管理和动态连接库 …… (208)
11.1 Windows 3.X 内存结构 …… (208)
11.2 内存和 Windows 操作模式 …… (208)
11.2.1 内存段属性 …… (208)
11.2.2 标准模式 …… (209)
11.2.3 386 增强模式 …… (211)
11.2.4 程序结构 …… (211)
11.2.5 Windows 3.X 内存管理 API …… (212)
11.3 Win32 内存结构 …… (213)
11.4 对象内存管理 …… (213)
11.4.1 缺省 new 和 delete 运算符 …… (213)
11.4.2 VAllocator 类 …… (215)
11.4.3 VHandle 类 …… (217)
11.4.4 模板类 …… (218)
11.4.5 重载指针递引用 …… (221)
11.4.6 扩充 VHandle 类 …… (222)
11.5 使用动态连接库 …… (225)
11.5.1 LibMain 和 WEP …… (225)
11.5.2 Windows NT 动态连接库 …… (226)
11.5.3 引入或引出函数 …… (226)
11.5.4 对象的特殊需求 …… (227)
11.6 下一步做什么 …… (228)

第十二章 剪贴板、DDE 和 OLE 对象 …… (229)
12.1 Windows 内部通信 …… (229)
12.2 剪贴板 …… (229)
12.2.1 VClipboard 类 …… (230)
12.2.2 VClipItem 类 …… (230)
12.2.3 一个应用程序例子:CLIP …… (231)
12.3 动态数据交换(Dynamic date exhange) …… (240)
12.3.1 DDEML 基础 …… (240)
12.3.2 Vista DDE 类 …… (243)
12.3.3 VDdeObject 类 …… (243)
12.3.4 VDdeServer 类 …… (244)
12.3.5 数据管理回调函数 …… (246)
12.3.6 VDdeClient 和交谈类 …… (250)

12.4 对象连接和嵌入(OLE) …… (256)
12.4.1 为什么使用 OLE …… (257)
12.4.2 什么是 OLE …… (258)
12.4.3 OLE 服务器应用 …… (259)
12.4.4 OLE 客户应用 …… (260)
12.4.5 Vista OLE 类 …… (260)
12.4.6 事件调度 …… (261)
17.4.7 构造一个 OLE 服务器 …… (261)
12.4.8 构造一个 OLE 客户 …… (273)
12.4.9 调试 OLE 应用 …… (284)
12.5 小结 …… (284)

附录 A …… (287)
A.1 Vista 库 …… (287)

第一章　面向对象的 Windows 程序

图形用户接口(GUI)与面向对象的程序设计之间有着密切的关系，在 Xerox′s Palo Alto 研究中心，第一个 GUI 应用于 Smalltalk 开发系统的界面，初期的 GUI 开发者认识到该图形环境为面向对象的程序设计提供了一个理想工具。

当包括 Microsoft 在内的其他销售者开始开发他们自己的图形环境时，他们确定了他们使用的程序语言，象 C 和 Pascal。对成功的商业系统而言，面向对象的语言当时尚不够成熟或跟随不够，因此，大多数 Windows 程序用 C 语言写成。

本章讨论面向对象的原理，该原理为 Windows 程序设计使用 C++语言概念，首先，我们将讨论一个面向对象的系统到底是什么，其次，我们将考察 Windows 环境本身使用的对象模型；最后，我们将考察一些有用的 C++特性，我们将用此来增强 Windows 对象模型。

1.1　面向对象系统

回顾过去的几年，术语“面向对象”几乎已变为陈词滥调，软件发行者都宣传他们的产品是面向对象的，而且媒介极力称赞面向对象技术的好处，但很多人对术语“面向对象”的真正含义意见不一。有鉴于此，在深入讨论之前，我们将定义一些术语。

一个面向对象的系统或语言以对象作为一个基本部分，Grady Booch 在他的《面向对象的应用程序设计》一书中，简单地定义一个对象为“你能对其做事情的某物”。更进一步，“一个对象有状态、过程和标识”。换句话说，一个对象有操作，作用其上的数据全部统一到一个包装内，一个没有操作的对象仅仅是数据，而没有数据的对象仅仅是过程的集合。

按照 Booth 的说法，要把一个系统或语言划为面向对象的，它必须显示三个特性：

- 它必须对对象和实例提供直接支持，实例就是对象的同义词。对象的数据对一般的使用应当是看不见的，而且它的行为由专门的操作设置控制。
- 对象以类分类。一个简单类对象拥有数据和过程，面向对象的类的概念类似于过程环境下的数据类型。
- 对象类通过继承相互关联，或在类的层次结构中共享过程或数据的能力。

C++语言的最早设计者 Bjarne Stroustrup 对面向对象的程序设计语言增加了第四个要求，即我们能将其扩展到包括象 Windows 这样的环境中去。一种面向对象的语言或环境必须使其适合所有对象。如果它花太多的工作开发对象，那么该环境便不是面向对象的。例如传统的程序设计语言象 C、Pascal 或 COBOL 允许你创建对象，但它们对继承不提供直接支持，因此，虽然你可以用这些语言强迫程序使用对象，但它们不是面向对象的。

1.2　Windows 对象模型

因为您现在已经清楚了对象的含义，而且知道术语“面向 对象”意味着什么，所以，我们将考察 Windows 环境的一些主要部分，看看它们怎样实现自己的对象。

1.2.1 图形对象

Windows 在它的图形库中使用有限制的面向 对象的概念，即图形设备接口(GDI)，所有图形操作作用于设备描述表，它描述象显示器或打印机这样的物理设备，设备描述表实际上是由 GDI 支持的数据结构，包含了状态信息，同样方式，C++类包含数据成员。用户通过过程集检索或修改信息，就象您用成员函数修改对象成员数据。

GDI 同样提供工具集用于画图形到设备描述表。该设备描述表可跟踪在给定的任何时间里正使用的工具——画笔、笔刷、调色板和字体。每种工具用独立的函数集来控制不同的图形属性。GDI 的设备描述表和绘图工具的使用简化了图形操作。该设备描述表能记住用户事先设定的属性可用于其后的绘图要求，从而代替了绘图时用户时刻得指定每种属性的需要。

GDI 的图形模块中的概念运用了面向对象的基本原理：该设备描述表可视为一个对象和 GDI 函数对该对象操作。不幸的是，在此接口全部是过程。有几百个过程管理设备描述表和绘图工具于用于画图形元素。在第二部分，我们将使用基于 GDI 由 Vista 库提供的 对象来扩充 GDI 的图形模块。

1.2.2 窗口对象

顾名思义，在 Windows 中的中心论题是窗口。每个窗口是一个其所拥有的数据和定义该窗口行为过程的对象，各个窗口按其类型分类，相应 C++中的类。窗口类提供一种窗口的一般属性和行为。例如，所有按钮控制属于一个单个窗口类，同样诸如列表框、滚动条及其它控制，以及对话框和多文档接口(MDI)窗口。

包含在窗口类之中的属性之一是窗口过程的地址，该过程实现类的行为 。当可能影响窗口的事件发生时，窗口过程响应传给它的信息。所有应用程序共享窗口过程，以便任何时候一个窗口接收一个事件，类的实例以同样方式作用于信息，这种安排增加 Windows 环境的一致性。一个应用程序的按钮的作用与其它应用程序的按钮一样。同样，对话框、滚动条、编辑控制和列表框都共享普通的窗口过程，因而不同的程序共享相同的行为。

Windows 窗口过程是等价于 C++成员函数，但有两点不同。一个 C++成员函数的范围是非常有限的，仅处理 对象行为的一小部分。一个窗口过程，从另一个方面来说，担负着实现一个窗口行为的全部。除了处理单个操作，它如同一个交通警察，用于管理窗口行为。

Windows 通过子类和超类支持有限继承形式，通过重写或参量化一个窗口过程，用户可能指定该窗口的一个新行为。例如通子类编辑控制，你可以创建一个仅从用户接受数字数据的文本区域。与面向 对象的原理一致，一个窗口子类重复使用代码实现其父类的行为。

虽然子类看起来可以继承，其实不能。当用户创建一个窗口并再细分类，你改变的仅是一个窗口的行为。超级分类，从另一方面来说，修改一个已存在的窗口类生成一个新的类，这可以被认为是真正的继承。

1.2.3 事件与消息

在 Windows 内部具有面向对象本质的另一个地方是它的通信方案。Windows 使用一种事件驱动的通信方式，非常类似于很多面向 对象语言使用的方式，当一个事件产生时，它传

递消息给窗口，要求窗口注意。当该窗口接收一个消息时，它决定将做什么，假如有的话。

该相同消息传递通信方案也用于一个应用程序的窗口之间及多个应用程序之间，要改变一个窗口控制的显示，传给它一个消息；要获取一个编辑控制的文本，传给它一个消息。Windows 的两个内部过程通信协议：动态数据交换(DDE)、对象连接和嵌套(OLE)，也是由消息实现的。

Windows 使用消息传递通信方案提供一个明显的面向对象特性：多态性，指的是不同类型的对象响应相同操作的能力。所有窗口从系统中接收相同的标准消息，但它们接收这些消息时，不同窗口类的实例呈现不同的行为。

例如，Windows 给所有窗口发送一个 WM_PAINT 消息，通知它们需要显示它们自身，各窗口则调用必要的 GDI 函数画它自身，当 Windows 传递该消息时，并不区分窗口类：它以同样方式对待应用窗口，弹出窗口及窗口控制，窗口接收同一消息但基于它们的内部状态进行不同的动作。

1.2.4　Windows 是面向对象的吗？

在观察 Windows 使用的图形、窗口管理及通信模型之后，您可以对它是否是面向对象的做出自己的决定。初看一眼，它似乎是的。窗口是属于支持继承类的对象，对象通过消息传递和提供多态性的形式支持通信。

如果您回想一下以及全面观察下一系统，无论如何，您将发现某些前后不一致，虽然窗口自身是对象；但整个应用程序接口(API)用于创建和管理那些窗口是过程的。没有一个容易的方法创建和管理窗口 对象。GDI 使用巨大的过程库增加了该不一致性。

所以虽然 Windows 不是严格面向 对象的，但明确地倾向使用 对象，在面向对象编程中，最困难的任务之一是在你的程序中找出对象作模型，Windows 通过划分定义明确的窗口类级别使得该任务大大简化。在下一节，我们将讨论你怎样使用 C++建立 Windows 对象模型和使用真正的面向对象技术建立 Windows 程序。

1.2.5　面向对象的 WindowsC++编程

Windows 程序已经用 C 语言传统地开发出来。C 给了程序员处理应 Windows 这样动态的环境所需的灵活性，也提供了在十分需要的图形环境中建立敏感程序所需的效率。尽管如此，一对面向对象的语言，即 Aotor 和 Smalltalk，已经艰难地维持住它们自己的 Windows 开发市场份额。随着 Borland C++这样的 Windows C++编译器的新近引入，该语言正迅速成为 Windows 程序员最普遍的面向对象语言。

1.3　为什么使用 C++

Windows 应用程序能以 DOS 应用程序同样方式从 C++中获益。Bjarne Stroustrup 设计的 C++语言使得编写程序更加容易，程序更可靠和更易扩展。

1.3.1 一种更好的 C

C++"继承"了 C 语言的绝大部分特色。它使用相同的运算、控制语句和输入输出函数。良好的 C 程序——那些没有明显使用指针运算和 C 语言特征的程序,这些使得 C 成为低级语言——同样也是良好的 C++程序。大部分写得好的 Windows 程序勿需大的修改就能在 Borland C++中编译。

除了支持 C 提供的大部分特色,C++增加了在 C 语言上改进的新特点。

严格的类型检查

C++对函数参数的数据类型的测试防止了许多编程错误,通过要求函数原型,编译器知道函数所希望的参数数目和参数类型以及函数的返回值类型。如果你在调用函数时出错,编译器在编译时就能知道。

引用变量

C++对其它变量使用引用变量使用别名。利用引用,你可以用地址传递一个变量而不用递引用一个指针来访问它。与指针不同,引用变量必须初始化指向一个有效的变量。这样它们可以消除许多 C 程序普遍的指针错误。我们将使用引用变量延伸到试图减少你自己程序中的那些问题。

常数

C++提供的真常量减少了对协处理器的需要。C++程序使用允许编译器对常参数进行类型检查的常量,以取代用处理器指令定义常数,Const 关键字也可用于说明不变的变量,即使传给不同的函数。

函数与运算符重载

在 C++中,两个函数可以共用同一名称,只要它们具有不同数目或类型的参数。函数重载允许定义通用函数,能以不同的参数辨别相同操作。C++也允许重载运算符,如同用户定义类型对预定义类型的支持。

1.3.2 面向对象的扩展

C++为 C 语言增加的严格类型检查及其它特点使任何程序受益。不管怎样,C++提供的最重要的扩展是对对象提供支持,当你用 C++面向对象的特点写程序时,你不用费神于实现程序所需的过程,而代之以集中于对象以及它与程序内其它对象的关系上。

建立一个 C++程序,你要决定你将给出哪几个对象、描述不同对象的关系以及确定作用于那些对象的操作。C++提供的数据抽象和继承机制,允许你如同预定义类型一样定义新的数据类型(类)。这些新的数据类型使你写程序看起来更自然且更易于维护,它们也允许你建立对普遍编程难题提供通用解法的组件。

所有这些对一些 Windows 程序员意味着什么呢?它代替你集中精力于创建每个程序时对 Windows 的需求,为用户关照用对象产生的那些的程序的细节。用 C++,放在首位的是,你应致力于你的程序打算提供的操作。创建在多应用之间能共享的一些可重复使用的类,而不是写实现低水平的窗口管理代码。

1.3.3　类库

为了使产生可重复使用的类的工作更容易，贯穿本书的范例使用 Vista APplication Framework 类库的对象来建立，类库如 Vista 备有工具集，提供了一些问题的解决方法，它不同于过程语言的库，象屏幕管理或通信库（或 Windows API）仅提供函数集。你调用库中的函数执行专门操作。类库，从另一方面来说，提供了通用对象集，在你的程序里你可以将其作为更特殊对象的出发点。

典型的类库提供可构造完美应用程序的对象层次。例如，Borland C++提供的包容类库（Contairer Class library）提供类属包容对象（Container objects）如：袋、集合、队和二叉树。你可以用这些对象类为基础建立一个框架构造完美的应用程序。

建立一个良好的类库需要仔细地设计和组织，当你写程序时，你可以对将使用的成分进行假设，而类库的设计者不能做这些假设。因为库中的类意味着要被许多程序使用，它们的灵活性，一致性和可扩展性是非常重要的。

1.3.4　应用程序框架

应用程序框架为用户提供了一个框架用于包裹你的应用程序代码，供给你建立一个应用程序所需的任何东西。不象特殊目的类库中的类，应用程序框架中的类提供非常广的接口-仅用框架提供的成分你就可以建立一个完善的应用程序。

仅使用那些应用程序构架中对象建立应用程序将不能用于任何可用的用途。使程序可用是你的工作。当你使用应用程序框架时，你不需要对你的应用环境强加的特殊需要获得允许，取而代之的是，从框架提供的那些定义中导出新类。你的类仅需实现你的应用程序独特的行为。

在 Windows 环境中，应用程序框架是特别有用的。让应用程序类去操心建立 Windows 程序所需的处理消息、访问高和低字、引出函数以及所有其它细节。你集中精力解决的问题是那些对象的行为。这是你的应用程序最初要解决的。附加受益的是，应用程序框架能使你的程序更易移植到其它环境。

对于 Windows 程序有一些应用程序框架和类库可得到，包括 Borland 的 Object、Windows、Microsoft Foundation Classes、Zinc Interface Library、XVT、CommonView、C++ / Views 以及其它。本书中范例是基于 Vista 库，但很容易换成另一个。

1.3.5　Windows 程序转换到 C++

C++为你的过程 Windows 程序转换到更面向对象的形式提供了理想的途径。用 C++，你有两个选择来转换你的程序。首先是用任何最新的面向对象设计和分析方法来完全重新设计你的应用程序，如果你的应用程序要作大的修改，那么这是最好的解决方案。

第二个解决方法——本书所用——是一次转换一步。因为 C++是 C 的扩展，你可以在同一程序里混合和搭配 C 与 C++代码。这种能力使得 C++成为对很多开发者具有极强吸引力的语言。若要使用 Vista 库转换你的应用程序：

- 给程序增加一个应用程序对象。将程序的初始化和结束动作转换到属于该对象的成员函数中。

- 下一步，转换产生窗口和对话的代码到产生对象的代码。
- 再下一步，将你的消息处理开关语句分裂成动作方法。
- 最后一步，将你的画图操作修改成使用图形对象。

你可依自己的期望将 C 程序转换到 C++，从而避免从零开始，利用 C++新特点又不损失 C 的简单性和速度。贯穿本书的剩余部分我们将介绍转换技术。

1.4 Vista 应用程序框架

在本书中，我们将使用 Vista 应用程序框架建立范例。Vista 库包含的类集比 Windows 提供的抽象水平更高，它包括类的层次，你在屏幕上可以看到，最直接映入对象，框架本身被分成若干域。

1.4.1 应用程序支持类

Vista 提供隐藏了 Windows 编程细节的高级应用类集和一些低级支持类。表 1-1 列出了应用和支持类。

表 1-1 Vistal 应用和支持类

类	说明
VAppl	顶层对象，控制一个 Vista 应用。
VEvent	Vista 窗口类所有通信的资源。VEvent 类等价于 Windows 消息。
VString	字符串类。对支持字符串、字符资源及 DDE 字符串提供操作。
VAllocator	大小固定的内存分配类。
VHandle	抽象句柄类。
VGlobal	全局内存句柄 表示的对象的句柄类。
VLocal	局部内存句柄表示的对象的句柄类。
VArray	数组模板类。
VDynArray	动态数组模板类。
VList	链表模板类。
VDblList	双链表模板类。

1.4.2 窗口类

窗口类封装 Windows 提供的窗口类，它们通过提供有效的实现通用操作的方法增加了 Windows 编程环境的一致性。它们为访问整个 Windows API 提供了便利，而不必复制其操作且不需对 C++进行任何扩展，表 1-2 表出了窗口类。