Borland C++ 5.0

OWL 5.0 编程技术与实例

雷宇 江明霞 代国定 高宏福

西安电子科技大学出版社

Borland C++ 5.0

OWL 5.0 编程技术与实例

雷 宇 江明霞 代国定 高宏福

西安电子科技大学出版社

1997

(陕)新登字 010 号

内 容 简 介

本书介绍了 Borland C++ 5.0 中的面向对象类库 OWL 5.0 及其编程技术。OWL 是支持高效编写 Windows 95 与 Window NT 的 32 位应用程序的有力工具。本书中包括了 Windows 95 程序设计基本概念、OWL 基础、各种控件类、对话框类、单文档和多文档界面及 OWL 的类库说明。针对 OWL 中的主要控件，书中给出了大量的程序实例代码。另外，本书通过一个较大、较系统的实例由浅入深地介绍了 OWL 应用程序的开发过程及开发技巧。这些内容对于学习面向对象的程序设计技术及学习 Windows 95 程序开发技术都会有较大的帮助。

本书可作为初学面向对象编程者的自学教材，更可作为想深入了解和掌握 OWL 编程技术的各类读者的参考书。

Borland C++ 5.0 OWL 5.0 编程技术与实例

雷 宇 江明霞 代国定 高宏福 编著

责任编辑 陈宇光

西安电子科技大学出版社出版发行

地址：西安市太白南路 2 号 邮编：710071

西安市长青印刷厂印刷

各地新华书店经销

开本 787×1092 1/16 印张 27 字数 645 千字

1997 年 6 月第 1 版 1997 年 6 月第 1 次印刷 印数 1—6 000

ISBN 7-5606-0523-0/TP·0255 定价：35.50 元

序言　用OWL 5.0编程

1995年到1996年期间，信息技术的发展真是令人兴奋。几乎所有的计算机杂志和报纸都在讨论未来的桌面计算机的发展趋势，讨论Internet的发展，讨论Intranet的结构，两家大公司为了争夺个人电脑32位操作系统的统制权在进行着兵团作战，各种各样的应用软件也在真刀真枪的拼杀；多少公司被收购，多少公司脱颖而出。

时至今日，有一点已经非常明朗：对于商业软件、家用软件和数据库前端软件的开发商和广大用户来说，Windows 95或Windows NT是最合适的32位桌面操作系统。尤其是Windows 95，它采用了新的用户界面，更加易学、易用，系统更加稳定、高效。

都知道Windows 95好用，但Windows 95程序难编。Windows 95是一个复杂的系统，它对应用程序有严格的要求，使得通常的程序设计方法难度大而效率低。为了摆脱这个局面，许多软件公司推出了基于Windows 95的程序开发工具，其中有Microsoft公司的Visual Basic 4、Visual C++ 4，Powersoft公司的Powerbuilder 5，Borland公司的Delphi 2、BC++ 5.0、Intrabuilder等等，它们都能不同程度的满足用户的需要，而且各自都有许多过人之处，用户可以针对不同的任务选择不同的开发工具。

那么在什么情况下选择BC++ 5.0作为Windows 95的开发工具呢？

1. 如果希望尽量发挥Windows 95系统的功能并且展示你自己的风格，还想要有一个高效率，希望利用BC++ 5.0所提供的工具尽量减少手工编制代码的繁琐过程；

2. 如果需要做大量的科学计算(包括图形、图像处理)，并且充分利用以前在BC上所做的程序代码；

3. 如果希望同时开发基于Windows不同版本的应用程序(比如基于Windows 3.x的16位应用程序和基于Windows 95或Windows NT的32位应用程序)，BC++ 5.0是能够满足这一要求的为数不多的开发工具，BC++ 5.0甚至依然能够开发高效的DOS应用程序，这是Borland公司为方便用户而努力的结果；

4. 如果希望快速开发“专业”的Windows 95应用程序，充分发挥面向对象和可视化编程的特点，那么BC++ 5.0是合适的选择，因为它包含了灵活而高效的OWL 5.0。

OWL不是猫头鹰，OWL 5.0(Object Windows Library)是BC++ 5.0中的精华，它采用面向对象技术封装了Windows的应用程序接口，使C++程序员能够高效地编写Windows程序。它提供了Windows中的界面元素(比如，窗口、按钮等)的“类”形式，使程序员更容易控制程序的用户界面；它提供了更为简洁的事件驱动定义方式，摆脱了用C语言时的繁琐模式。

OWL从BC++ 3.1中的OWL 1.0发展到BC++ 5中的OWL 5.0，其间变化剧烈，把面向对象的Windows编程技术一步步提高到新的水平，因而受到软件开发商和越来越多的C++程序员的青睐。

凡是用过OWL的C++用户都会有同样的感受：用OWL开发Windows程序要比传统的开发方式优越得多，它省去了不少编程时的麻烦，把程序员从繁重的近似于“体力劳动”

的工作中解脱出来，而能把主要的精力放在软件的功能上。BC++ 5.0 面市以来，OWL 的优势又一次将 Windows 编程方式提高到新的水平。OWL 5.0 几乎覆盖所有 16 位和 32 位的 Windows API 函数，它的封装自然而富有逻辑，总让人觉得 Windows 的面向对象机制本来就是这样的。OWL 5.0 中包括的一系列高层次的类，比如，工具条、状态条、浮动窗口、打印预览以及菜单类等等，都可以帮助你建立起复杂的应用程序。

对于想同时进行 16 位和 32 位 Windows 应用程序开发的人员来说，OWL 5.0 是再合适不过的了。只要利用多目标项目管理器、TargetExpert 和 Source Pools 来管理目标程序，就可以轻而易举地将相同的程序原代码在 16 位和 32 位 Windows 目标代码之间切换。

如果想学习 OWL，而你又没有多少 C++的经验，不要紧，学习 OWL 本身就可以是学习 C++的一个精彩的开头。OWL 中的类结构的规范性和灵活性是你在任何 C++教程中所不常见到的；如果想学习 Windows 编程，OWL 中的类层次结构和体系可以帮助你正确的理解 Windows 编程中的难懂的概念。它就是你编写 Windows 程序的开始。OWL 就是 Borland 公司用 C++语言强化了 Windows 中的面向对象机制的结果，所以在你学习 OWL 的过程中，就可以深入地理解 C++的用法和 Windows 的编程方法。

在这本书中包括了 OWL 5.0 中常用的和有用的类的详细解释，尤其对编程实践中经常遇到的类给出了大量的程序实例，书中还包括了 OWL 的一些使用技巧，可以使读者更加深入地理解 OWL 的用法。

衷心感谢杨银堂老师、周端老师对这本书的大力支持！衷心感谢陈宇光老师在本书出版过程中的大力帮助！

作者水平有限，书中难免存在错误和不足之处，敬请批评指正。

衷心希望本书会对您的事业有些帮助。

编　者

此为试读，需要完整PDF请访问：www.ertongbook.com

目　录

第 1 章

Windows 95 程序设计基本概念

1.1　Windows 95 的优越性

一般认为，Microsoft Windows 是一个优于 DOS 的图形用户界面的操作系统，它为应用程序提供了一致的窗口用户界面和多任务环境。而 32 位的 Windows 95 是比 16 位的 Windows 3. x 更具有优势的操作系统，它的新型用户界面更加容易学习和使用，32 位应用程序的执行速度更快，更能够充分发挥计算机硬件的功能，系统也更加稳定(相对于 Windows 3. x)，因此，目前 Windows 95 成为主流的桌面操作系统是必然的。

Windows 95 的界面对用户是友好的，Windows 95 的编程环境对程序员也是友好的。如果你有编写 DOS 应用程序的经验，你会记得给应用程序增加友好的界面是一项复杂的工作，所有的图形界面元素都需要从头编起。而在 Windows 95 环境中，一致的界面带来了一致的界面设计过程，通用的界面元素都已经建立(包括丰富的 OCX 和 VBX 控件)，你只需要学会使用它们，就可以在不同的程序中利用这些界面元素生成应用程序界面。你的主要精力可集中在编写应用程序的功能上。

因为是图形用户界面，所以在 Windows 95 环境中开发图形软件就比较方便；Windows 95 环境直接支持鼠标，所以在程序中使用鼠标功能是非常方便的；Windows 95 环境对内存的管理能力强，因而应用程序可以使用更多的内存资源；Windows 95 提供了“与设备无关”的程序运行环境，因此程序会不依赖于某一种显式硬件或打印机；Windows 95 中内含丰富的多媒体功能，使开发多媒体应用程序更加方便。

Windows 95 的优势一言难尽，相信它的用户会有体会。

1.2　窗口的组成

窗口是 Windows 95 程序进行屏幕输出的方式，它包括了一系列部件(见图1 - 1)，其中有：

窗口边框：窗口边框包围着窗口，它可以使窗口的大小改变；

客户区域：窗口中除了菜单条、滚动条、边框等窗口的外围部分以外的窗口的实际工作区域，窗口的客户区域由应用程序控制；

菜单条：菜单条是应用程序提供给用户的命令选择工具，菜单条由程序管理；

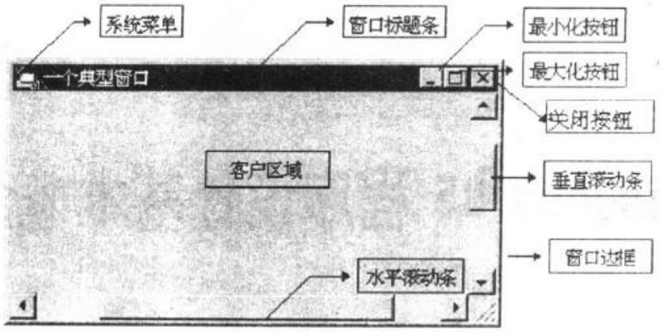

图 1-1　典型的 Windows 95 窗口

滚动条：包括水平滚动条和垂直滚动条，它可以帮助窗口显示比窗口实际尺寸大的文档或图形；

最大化按钮：用户点击最大化按钮可以使窗口最大化，充满整个屏幕；

最小化按钮：用户点击最小化按钮可以使窗口最小化，缩小为一个图标；

系统菜单：系统菜单在应用程序窗口的左上角，用户可以通过此菜单对系统进行操作，双击此处则可以关闭应用程序；

标题条：标题条显示出窗口的名称或应用程序的名称；

关闭程序按钮：用户点击此按钮可以关闭应用程序。

1.3　32 位 Windows 应用程序接口与 OWL 5.0

Windows 95 应用程序接口(Win32 API)函数是 Windows 95 应用程序的核心。通过 Win32 API 函数可以利用 Windows 95 的功能和"与设备无关性"，并可编制出运行于 Windows 95 与 WindowsNT 的 32 位应用程序。不过，Win32 API 中的函数达数千个之多，这足以使程序员们感到头昏眼花。OWL 5.0 正是为这些痛苦不堪的程序员们提供的妙方，它采用清晰的逻辑、自然的结构，用面向对象的方法封装了几乎所有的函数，使用亦非常方便。

BC++ 5.0 中包含了一个 Microsoft 公司编写的专门用于 Windows 环境的游戏开发应用程序接口(WinG)，它使游戏程序可以像在 DOS 环境中一样快速显示图形，这对那些热心 Windows 95 环境下开发游戏程序的用户来讲真是一个福音。

另外，WindowsNT 中包含了从 SGI 工作站移植过来的 3D 图形库 OpenGL，它使高性能的 WindowsNT 的工作站可以像图形工作站那样进行复杂的图形处理。BC++ 5.0 中提供了 OpenGL 的应用程序接口，这对于 CAD 或图形处理的用户又是一个福音。

1.4　消息循环与事件驱动

DOS 程序一般采用顺序的、过程驱动的结构，而 Windows 95 程序是采用事件驱动的结构方式(见图 1-2)。事件驱动程序不是按某种固定顺序来执行的，而是由事件的发生来控制执行的。通常，事件以消息的方式进入程序，比如，当鼠标的一个按钮按下时，应用程序将接到一个鼠标按钮按下的消息，程序执行这个消息的响应程序。

事件驱动的程序在启动初始化后就进入了消息循环中。Windows 95 应用程序接收的消息种类很多，包括：输入设备的消息、键盘的状态、设备的状态和时钟消息等。Windows 95 负责把所有输入设备的消息放入系统消息队列中，然后将消息分配到各个应用程序的消息队列中，由应用程序处理。

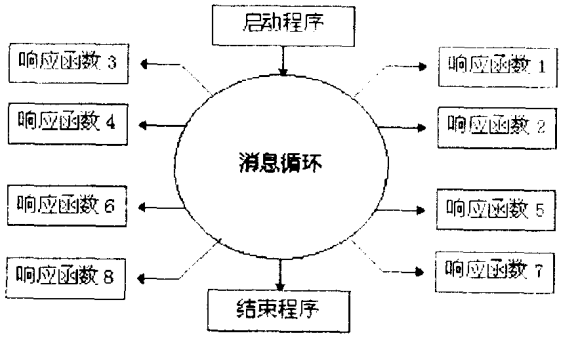

图 1－2　Windows 95 程序的事件驱动方式

1.5　窗 口 过 程

窗口过程是接收和处理消息的函数，它负责执行不同的代码来响应不同的消息。在 C 语言中，窗口过程利用 switch 语句进行消息响应，因而比较繁琐。使用 OWL 5.0 可以大大简化这个编程过程。

1.6　C 或 CPP 文件

以 C 或 CPP 为扩展名的文件包含了 Windows 95 应用程序的源代码，在项目文件中可以有多个 C 或 CPP 文件。

1.7　资源与资源文件

一般的 Windows 95 应用程序都需要使用图标、菜单、光标、位图、对话框之类的资源，资源文件(*.rc)就是用来存放这些资源的。项目文件中可以有多个资源文件。在应用程序的开发中，可以使用 Resource Workshop(资源编辑器)来开发应用程序的资源，资源编辑器将资源文件与编译好的可执行模块 EXE 或 DLL 连接成为完整的 Windows 95 应用程序。

1.8　模块定义文件

定义文件(DEF)中提供了应用程序的特别重要的信息，包括：应用程序性质、代码和数据段大小和性质、局部堆的大小等，如果在项目文件中不包括这个文件，Borland C++会使用一个缺省的定义文件。

此为试读，需要完整PDF请访问：www.ertongbook.com

1.9 最基本的 Windows 应用程序

下面是一个用 C 语言编写的最基本的 Windows 应用程序，它可以改变窗口的大小、能够移动、关闭、最小化、最大化窗口。

```
/*********************************************
        最基本的 Windows 应用程序
*********************************************/
#include <Windows.h>
#pragma hdrstop

LRESULT FAR PASCAL export WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)

//Windows 应用程序的主窗口
int PASCAL WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
          LPSTR lpszCmd, int nCmdShow)
{
   HWND        hwnd ;
   MSG         msg ;
   WNDCLASS    wndclass ;
   //设置窗口类，登记一个窗口
   if (! hPrevInstance)
   {
      wndclass.style         = CS_HREDRAW | CS_VREDRAW ;
      wndclass.lpfnWndProc   = (WNDPROC)WndProc ;
      wndclass.cbClsExtra    = 0 ;
      wndclass.cbWndExtra    = 0 ;
      wndclass.hInstance     = hInstance ;
      wndclass.hIcon         = LoadIcon( NULL, IDI_APPLICATION );
      wndclass.hCursor       = LoadCursor (NULL, IDC_ARROW) ;
      wndclass.hbrBackground = (HBRUSH)(COLOR_WINDOW + 1);
      wndclass.lpszMenuName  = MAKEINTRESOURCE( CHELPAPMENU );
      wndclass.lpszClassName = szAppName ;

      RegisterClass (&wndclass) ;
   }

   //创建一个窗口
   hwnd = CreateWindow (szAppName,
     "Windows 95 Help Example",
     WS_OVERLAPPEDWINDOW,
```

```
        CW_USEDEFAULT,
        CW_USEDEFAULT,
        CW_USEDEFAULT,
        CW_USEDEFAULT,
        NULL,
        NULL,
        hInstance,
        NULL) ;

    //显示窗口
    ShowWindow (hwnd, nCmdShow) ;
    UpdateWindow (hwnd) ;

    //消息循环
    while (GetMessage (&msg, NULL, 0, 0))
    {
     if ( ! TranslateAccelerator( hwnd, hAccel, &msg ))
     {
        TranslateMessage (&msg) ;
        DispatchMessage (&msg) ;
     }
    }
    return msg. wParam ;
}

//窗口过程
LRESULT FAR PASCAL export WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM
lParam)
{
   switch (message)
   {
   case WM_DESTROY:
         PostQuitMessage (0) ;
         return 0 ;
   }
   return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

第 2 章

OWL 5.0 基础

2.1 窗口和 TWindow 类

Windows 的核心是窗口。前面已经介绍了窗口的基本概念，在 OWL 5.0 中，TWindow 是所有窗口类的基类，它封装了 Windows 中窗口的基本特性和基本功能。

TWindow 类所封装的窗口特性和与窗口有关的一系列成员函数，提供了窗口的初始化、创建、操作及关闭窗口的功能。可以直接用 TWindow 来构造一个窗口对象，但是通常不这样做，往往只把它作为一个更有个性的窗口类的基类。

TWindow 派生出一系列窗口类的子类，包括 TFrameWindow，TControl，TDialog 和 TMDIChild 类等。从 ObjectWindows 的类结构层次图可以看到 TWindow 类与它的一部分子类的关系(见图 2-1)。

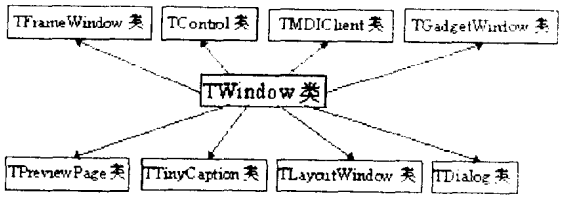

图 2-1 TWindow 类与它的派生类

如何用一个窗口类实现一个典型的窗口？我们可以用下面的两个步骤完成：

1. 用 TWindow 类或者它的派生类构造一个窗口对象；

2. 调用 TWindow 类中的成员函数 Create 或 Execute 来创建窗口界面元素(窗口的实际显示部分)。

例如：

```
TWindow * window = new TWindow(this);
window->Create();
```

如果使用 TWindow 类的派生类(比如 TNewWindow)创建窗口也是同样的。

例如：

```
TNewWindow * window = new TNewWindow(this);
window->Create();
```

TWindow::Create()函数为窗口的创建做了一些必要的准备工作，包括：

1. 创建窗口界面元素；

2. 设置窗口句柄 HWindow ；

3. 设置窗口属性参数(包括窗口大小、显示方式等)；

4. 调用 TWindow::SetupWindow()。

在调用 Create 函数时你当然可以不必关心这个函数内部的事情，但知道这个“秘密”是有好处的！你可以在调用 Create 函数创建窗口前设置窗口的属性 TWindow::Attr(窗口类的数据成员)，以得到想要的窗口风格；你还可以在 TWindow 的派生类中重载 SetupWindow()函数(这是窗口创建前调用的最后一个函数)，并用它做一些窗口的初始化工作。之后，这个窗口就可以交付使用了。

当程序关闭这个窗口时，根据具体情况不同，用下面的两个方法之一来删除窗口界面元素(与 Create 函数对应)：

1. 调用成员函数 Destroy 来删除窗口界面元素；

2. 调用成员函数 CloseWindow，它又会调用成员函数 CanClose 弹出一个对话框让用户确认这一操作，如果用户确认就删除窗口界面元素。

当窗口对象不再使用时，有两个方法来删除它(与通常的对象删除方法完全一致)：

1. 如果窗口对象是 new 创建的，使用 delete 删除；

2. 如果窗口对象不是 new 创建的，它会被自动删除。

一个典型的 Windows 应用程序可以具有许多不同风格的窗口对象：有的是覆盖窗口、有的是弹出窗口，有的是有边界窗口，有的是有滚动条窗口，还有的是有标题窗口。不同的窗口风格是通过设置窗口属性参数 TWindowAttr Attr 确定的。窗口风格和窗口标题都是在窗口初始化时设置的，创建窗口元素时将使用这些参数。

一个窗口对象的属性参数包括窗口风格和窗口标题等一系列内容，表 2-1 是窗口属性类型的数据成员。

表　2-1

成员	类型	描　　述
Style	uint32	风格内容
ExStyle	uint32	扩展风格内容
X	int	窗口左上角在屏幕的横坐标
Y	int	窗口左上角在屏幕的纵坐标
W	int	窗口的初始化宽度
H	int	窗口的初始化高度
Menu	TResId	窗口菜单的 ID 值
Id	int	子窗口的 ID 值
Param	char far *	被 TMDIClient 用作保存多文档界面信息
AccelTable	TResId	窗口的加速键表 ID 值

下面的程序代码用 TWindow 类构造了一个窗口对象(见图 2-2)，并创建了与这个窗口对象相对应的窗口界面元素。然后设置了窗口的属性，显示这个窗口，最后删除界面元素和窗口对象。这就是窗口短暂的一生。

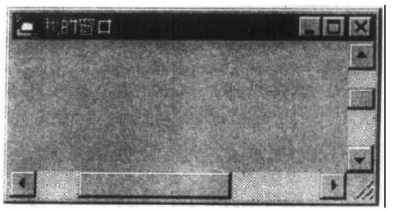

图 2-2 用 TWindow 创建的窗口对象

如果你想要窗口在消失之前做点什么，你就需要在窗口的界面元素被删除之前增加一些程序代码。

```
//构造一个 Twindow 对象：
TWindow * AWindow;
AWindow = new TWindow(0,"我的窗口");

//设置窗口的属性：
//窗口风格设置：
AWindow->Attr.Style |= WS_BORDER | WS_CAPTION | WS_CHILD | WS_HSCROLL | WS_MAXIMIZEBOX | WS_MINIMIZEBOX | WS_SYSMENU | WS_VISIBLE | WS_VSCROLL;

//窗口扩展风格设置：
AWindow->Attr.ExStyle |= WS_EX_TOPMOST;

//窗口的位置和大小：
AWindow->Attr.X =10;
AWindow->Attr.Y =20;
AWindow->Attr.W = 80;
AWindow->Attr.H = 20;

//设置窗口的背景颜色：
AWindow->SetBkgndColor(RGB(240,252,128));

//创建与窗口对象相关的窗口界面元素：
AWindow->Create();

//显示窗口
AWindow->Show(SW_SHOW);

//在这里加入你自己的程序代码
⋮
//删除窗口的界面元素：
AWindow->Destroy();

//删除这个窗口对象：
delete AWindow;
```

这段代码并不是完整的程序，但它们却完成了窗口对象从创建到删除的整过程。

另外，如果一个窗口是应用程序的主窗口，那么应用程序运行开始时将自动创建并显示它；应用程序结束时当然也会自动删除它。

2.2　从最简单的程序开始

也许你总是抱怨编写 Windows 程序是多么罗嗦，即使是最简单的 Windows 程序也需要写一大堆代码，你要定义一个回调函数、在 WinMain 函数中定义窗口、设置窗口的风格、创建窗口、显示窗口以及进行消息循环。是啊，真罗嗦！

现在，我们要用 OWL 5.0 来实现同样的功能(其实什么功能都没有)。让我们欢迎第一个 OWL 程序出场，它是最简单但同时也是一个完整的 Windows 应用程序。而且这个程序确实已经运行通过了。

```
#include <OWL\pch.h>
int OwlMain(int , char * [])
{
        return TApplication().Run();
}
```

在这个程序中，我们用 TApplication 类创建一个应用程序，这个应用程序具有一个缺省的 Windows 窗口，可以最大化、最小化和移动。你不需要为 ObjectWindows 应用程序显式地给出 WinMain 函数，而使用替代函数 OwlMain，OwlMain 可以使用在传统 C 和 C++程序的 main 函数中用到的参数 int argc 和 char * * argv。

当然，我们学习 OWL 并不是想要编写这样的简单程序，我们需要窗口能够完成一些事情(比如绘图或写字)，还需要我们的应用程序和其他流行的 Windows95 应用程序一样具有菜单条(比较复杂的那种)、工具条、对话框、包含子窗口、可以响应许多事件，最后，还要一个漂亮的图标。这些都是 Windows 95 应用程序的基本要求，而通过 OWL 都可以得到圆满的解决。

2.3　应用程序和 TApplication 类

ObjectWindows 用 TApplication 和 TModule 类封装了 Windows 的应用程序和 DLL 模块。TModule 类封装了 Windows DLL 的初始化和关闭功能，也包含了 hInstance 和 lpCmdLine 两个参数，这两个参数与非 OWL 应用程序的 WinMain 中同名的两个参数有着相同的意义。除非你要用 DLL，否则你是不需要自己建立 TModule 类的。以下，我们主要讨论 TApplication 类。

一个应用程序能够运行，它所需要的条件是：

1. 在包含文件中加入正确的 ObjectWindows 头文件。TApplication 的定义在头文件 OWL\applicat.h 中，使用 TApplication 必须包含这个头文件。当然，就正如你在前面的“最简单的程序”中看到的那样，如果用 OWL 5.0 创建应用程序，那么，CPP 文件中只要包含 OWL\pch.h 头文件就可以省去这些麻烦。

此为试读，需要完整PDF请访问：www.ertongbook.com