Windows 程序员使用指南(五)

——ObjectWindows库

[美] Namir Clement Shammas 著
王 敏 张津泉 译
黄 宇 审校

Windows Programmer's Guide to ObjectWindows Library

清华大学出版社

383077

Windows
程序员使用指南(五)
——ObjectWindows 库

[美]Namir Clement Shammas 著
王 敏 张津泉 译
黄 宇 审校

清华大学出版社

(京)新登字 158 号

Windows 程序员使用指南(五)——ObjectWindows 库
Windows Programmer's Guide to ObjectWindows Library
Namir Clement Shammas

Authorized translation from the English language edition published by Sams, a Division of Prentice Hall Computer Publishing Inc.

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 © 1995 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。
中文版版权(1995 年)属于清华大学出版社。未经出版者书面允许,不得以任何方式复制或抄袭本书的内容。

本书封面贴有清华大学出版社激光防伪标签,无标签者不得销售。

图书在版编目(CIP)数据

Windows 程序员使用指南(五):ObjectWindows 库
/(美)沙姆斯(Shammas, N. C.)著;王敏,张津泉译. —北京:清华大学出版社,1995.4
ISBN 7-302-01763-8

Ⅰ.W… Ⅱ.① 沙… ②王… ③张… Ⅲ. 操作系统(软件)-手册
Ⅳ.TP316

中国版本图书馆 CIP 数据核字(95)第 01521 号

出版者:清华大学出版社(北京清华大学校内,邮编 100084)
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:29.75 字数:703 千字
版 次:1995 年 5 月第 1 版 1995 年 5 月第 1 次印刷
书 号:ISBN 7-302-01763-8/TP・773
印 数:0001—5000
定 价:51.50 元

引　言

编写 Windows 应用程序无疑要比编写 DOS 应用程序复杂。由于过去 DOS 占主宰，所以大多数程序员都不需担心早期 Windows 程序员的可怕经历。最常见的就是仅仅为了要说"Hello World"Windows 程序就需要很多行。Windows 3.0 的普遍使用给广大程序员带来了一种与创建 Windows 应用程序相关的新型编程方法。Microsoft 的 SDK for the C Compiler 并未解决多少问题。现在你不必受累了：面向对象 C++ 语言的出现解决了一些问题。Windows 界面类层次结构，如 Borland ObjectWindows 库(OWL)，大大加强了 C++的使用。

本书读者

本书讲授如何使用 ObjectWindows 库编写 Windows 应用程序。为了更好地使用本书，我建议你：

- 熟悉 C++。
- 熟悉 Windows。
- 初学编写 Windows 应用程序。

本书旨在使用循序渐进的方法向大家介绍 ObjectWindows 的不同组成部分。但是，ObjectWindows 类不包括 Windows 编程的所有方面，如打印和图形。因此，我使用和讨论了一些 Windows API 函数以补充说明 ObjectWindows 应用程序。这些 Windows API 函数是有选择性的。如果我包括了大部分甚至全部 API 函数，那么本书就成了一本只有几章介绍 ObjectWindows 的笼统介绍 Windows 的书！

本书内容

本书包括 11 章，介绍了使用 ObjectWindows 库编程的不同方面。

第一章介绍了 ObjectWindows 类，教你如何使用 Windows API 函数和处理消息。本章为使用 ObjectWindows 库创建 Windows 应用程序提供了背景知识。

第二章给出了一些简单的 OWL 程序，说明了它们的基本组成部分。本章还讨论了使用菜单和菜单资源，创建 OWL 应用程序的多重实例和安全关闭一个窗口。

第三章更详细介绍了对应用程序窗口的操作。本章给出了许多有关创建非缺省窗口和 Windows 类登记的信息，还讨论了将文本写到并保持在窗口中和在窗口中滚动文本。

第四章是第三章的后记。它介绍了实现文本编辑器和文本文件编辑器的类。本章还讨论了如何使用 Windows API 函数给系统打印机发送文本。

第五章是三章介绍各种窗口控制中的第一章。本章讨论了静态文本、编辑框和按钮控制。第六章介绍了组框、复选框和单选按钮控制。这些控制可以使你细微地调整 Windows 应用程序的动作。第七章介绍了滚动条、列表框和组合框控制。本章还介绍了如何操作多

重选择列表框，使列表框的滚动同步及使组合框象历史列表控制一样工作。

第八章讨论了模式和非模式对话框，介绍了各种对话框的创建和执行，将对话框作为应用程序窗口的用法及数据的传输。本章提供了一些在应用程序和对话框或窗口之间传输数据的例子。

第九章介绍了多重文档界面(MDI)窗口。本章介绍了服从于MDI的应用程序的不同组成部分和两个应用程序的例子。这些例子说明了典型的和增强的MDI窗口。

第十章讨论了使用流存储和回调OWL及非OWL对象。本章介绍了使类流化所需的步骤。本章还说明了在应用程序中存储控制实例时流的使用以及它们在其他应用程序中创建应用程序窗口控制时的使用。

第十一章介绍了创建控制的资源文本语法。为了创建OWL对话框，你需要了解本章中介绍的非OWL知识。

> **警告**：使用ObjectWindows库编程显然比使用Microsoft SDK软件包容易。但是，创建复杂的OWL应用程序仍需一定的努力。虽然OWL类和Windows API函数使你能够从多种途径实现你所期望的程序功能，但是在使用一些技术时很可能会走进死胡同——并不是每个有意义的编程步骤都会起作用的！

本书附带一张盘①，内含两个自己提取的档案文件：CPPOWL30.EXE和CPPOWL31.EXE。如果使用Borland C++ 3.0编译器，就从CPPOWL30.EXE中提取文件。如果使用Borland C++ 3.1编译器就从CPPOWL31.EXE中提取文件。两个档案文件都包含.H,.RC,.RES,.PRJ和.CPP文件，这些文件均与本书中的程序有关。

安装这些文件时，首先创建\BORLANDC\OWL的子目录CPPOWL。使CPPOWL成为当前目录，然后执行CPPOWL30.EXE或CPPOWL31.EXE。这些简单步骤就解开了本书\BORLANDC\OWL\CPPOWL目录中的文件。

本书的其他特征

正如你在内容简介中所见，我会不时地强调一些重要信息。请看下列说明：

注意：这些注意字样提请你注意重要的信息。

提示：这些提示可以为你节省时间并为如何改进你的代码提供重要的线索。

警告：注意这些警告内容可以使你免去一些挫折。

除了这些注意、提示、警告外你还会发现一些重要代码的程序样板。这些样板被框在框里，如下所示：

```
void _TWinApp_::InitInstance()
{
    TApplication::InitInstance();
    // put additional statements here
}
```

① 需要该盘者请与清华大学出版社软件部联系，联系电话：2594831，邮政编码：100084。

试读结束，需要全本PDF请购买 www.ertongbook.com

目　录

试读结束，需要全本PDF请购买 www.ertongbook.com

第一章

ObjectWindows 基础

ObjectWindows 库给你提供了一个有力的工具，帮助你开发 Windows 应用程序。如果没有这个库，编制 Windows 应用程序将会变得更加困难、恼人，而且代码极长。ObjectWindows 库成功地将面向对象和事件驱动两种编程概念结合在一起，证实了这两种编程方法可以很好地在一起工作。本章向你介绍 ObjectWindows 类库及一些有关 Windows 的基本知识，你将学到下概念：

- 常用 Windows 数据类型
- ObjectWindows 数据类型定义约定
- ObjectWindows 层次结构
- Windows API 函数
- 如何响应 Windows 消息
- 如何发送消息
- 用户定义消息

本章中的信息为使用 ObjectWindows 库创建 Windows 应用程序作了准备。虽然 ObjectWindows 库提供了一个开发 Windows 应用程序的优秀工具，但是它并未包括所有的 WindowsAPI 函数。所以，你仍需要了解一些 API 函数。

1.1 常用 Windows 数据类型

Windows 程序包括一大批数据类型。本节中，我将重点放在最相关的数据类型上。熟悉这些数据类型使你能更好地理解 ObjectWindows 类的声明。Windows 数据类型包括一些简单数据类型和一组结构。简单数据类型包括许多用 typedefs 声明的别名，这些别名意义更加明确。它们区分开了函数和变量的声明。表 1.1 给出了最常用的 Windows 数据类型。

表 1.1 最常用 Windows 数据类型

数据类型	含　义
char	有符号 8 位字符
int	有符号 16 位整数
long	有符号 32 位整数
short	有符号 16 位整数
void	无类型值
BOOL	16 位整数，代表布尔值

续表

数据类型	含　义
BYTE	无符号 8 位整数
DWORD	无符号 32 位整数或段/偏移地址
UNIT	无符号 32 位整数，为将来兼容用
FAR	数据类型属性，用于创建 far 指针
FARPROC	指向函数的长指针
HANDLE	通用句柄
HINSTANCE	替换 Borland C++3.1 中的 HANDLE
HDC	显示上下文的句柄
HWIND	窗口句柄
LONG	同 long 类型
LPSTR	指向字符串的长指针
LPVOID	指向 void 类的长指针
PWORD	指向 WORD 类的指针
WORD	无符号 16 位整数

除了表 1.1 中列出的简单数据类型外，Windows 程序还包括许多结构。其中有两个简单但很重要的结构是：POINT 和 RECT. POINT 结构存储了一个点的 X 和 Y 坐标，声明如下：

```
struct POINT{
        int x;
        int y;
};
```

RECT 结构定义了矩形的左上和右下角坐标，声明如下：

```
struct RECT{
        int left;
        int top;
        int right;
        int bottom;
}
```

1.2 ObjectWindows 数据类型定义约定

ObjectWindows 使用了一种数据类型定义约定，使得更易区别数据类型。表 1.2 列出了数据类型定义约定的通用语法。表 1.3 描述了 ObjectWindows 类常用的某些数据类型，它们中间隐含了数据类型定义约定。

表 1.2 ObjectWindows 数据类型定义约定

约定	含　义
Pclass	指向 class 类的指针类型
Rclass	class 类的引用类型
RPclass	class 类指针的引用
PCclass	指向 const class 类的指针类型
RCclass	const class 类的引用类型

表 1.3 隐含数据类型定义约定的 ObjectWindows 类常用数据类型

数据类型	含　义
PCchar	指向 char 常量的指针
Pchar	指向 char 类型的指针
PCvoid	指向 void 常量的指针
Pint	指向 int 类型的指针
Pvoid	指向 void 类型的指针
RCObject	常 Object 的引用
Rint	int 类型的引用
Ripstream	ispstream 类型的引用
Ropstream	opstream 类型的引用
RPvoid	void 类型指针的引用

1.3 ObjectWindows 层次结构

ObjectWindows 库是类的层次结构，它使应用程序用户界面的创建成为一个整体。图 1.1 给出了 ObjectWindows 类的层次结构。Borland C++ 3.1 将类 TBWindow，TBButton，TBCheckBox 和其他一些未列入图 1.1 的类作为 ObjectWindows 层次结构的一部分。在 Borland C++ 3.0 中，它们也是可获得的，但是被作为 bonus 类。这个ObjectWindows 层次结构有两个基类：

类 Object，基本基类

类 TStreamable，第二基类

只有 TModule 类和它的子类 TApplication 是由 Object 类唯一派生的。其他ObjectWindows 类均继承自两个基类。有了这种多重继承，类就可以被包括在输入/输出流中以永久支持对象。

有了 ObjectWindows 层次结构，你就可以在创建 Windows 应用程序的用户界面时大大减少代码长度。本节给出各个 ObjectWindows 成员的类声明，并简要讨论每个类支持的功能。

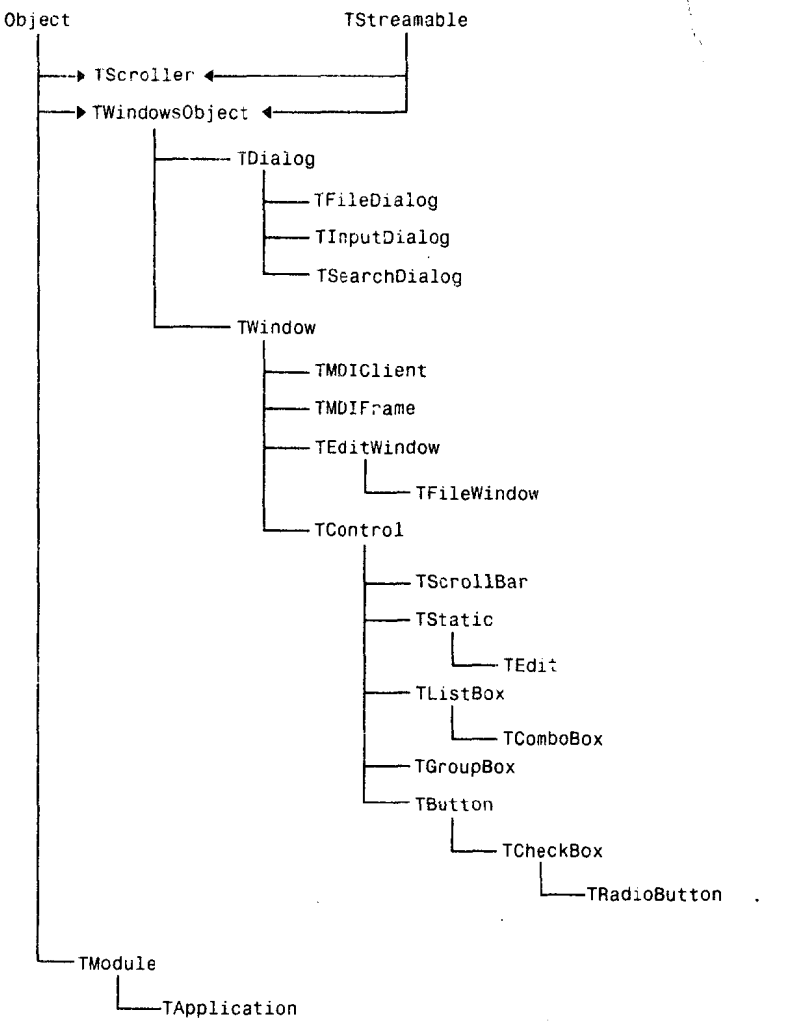

图 1.1 ObjectWindows 类层次结构

1.3.1 Object 类

Object 类是 ObjectWindows 层次结构的根。因此，Object 就是一个抽象类，它定义了通用于 ObjectWindows 类的一般结构和功能。下面给出 Object 类的声明：

```
class _CLASSTYPE Object {
public:
    virtual ~Object() {}
    virtual classType isA() const = 0;
    virtual char _FAR *nameOf() const = 0;
    virtual hashValueType hashValue() const = 0;
    virtual int isEqual(const Object _FAR &) const = 0;
```

```
    virtual int isSortable() const { return 0; }
    virtual int isAssociation() const { return 0; }
    void _FAR *operator new(size_t s);
    virtual void forEach(iterFuncType, void _FAR *);
    virtual Object _FAR & firstThat(condFuncType, void _FAR *) const;
    virtual Object _FAR & lastThat(condFuncType, void _FAR *) const;
    virtual void printOn(ostream _FAR &) const = 0;
    static Object _FAR *ZERO;
    static Object _FAR & ptrToRef(Object _FAR *p)
        { return p == 0 ? *ZERO : *p; }
    friend ostream _FAR& operator << (ostream _FAR&,
                                      const Object _FAR&);
};
```

这一声明表明 Object 类不是一个具有最少成员个数的高度抽象的对象。一个高度抽象的对象应该使用虚方式。Object 不是这样。相反，这个类定义了一些虚成员函数，它们提供了包容类的特性。这些成员函数包括 forEach，firstThat，lastThat，isSortable 和 isAssociation。静态指针类型的数据成员 ZERO 是分配失败时由新操作返回的地址。isA 和 nameOf 成员函数是抽象函数，它们返回类分类和某一类名。这些成员函数在查询类名或实例的分类时有用。

注意：_CLASSTYPE 标识符是一个特殊的宏，因为它根据正在使用的数据模型扩展为 huge，far 或 near。

注意：ObjectWindows 类声明通常包括_CLASSDEF 宏。这个宏声明了与已被声明的类有关的指针和引用 typedefs。

_CLASSDEF 宏使用一系列其他的宏定义了这些 typedefs，结果产生如下声明：

```
typedef classname _FAR * Pclassname
typedef classname _FAR & Rclassname
typedef classname _FAR * _FAR & RPclassname
typedef const classname _FAR * PCclassname
typedef const classname _FAR & RCclassname
```

其中_FAR 宏在适当的时候被自动定义为 far。如果设置了__DL __AKG _CLASSDLL 宏，则将_FAR 宏扩展为 far。

1.3.2 TModule 类

TModule 类是 Object 类的直接后继。TModule 给出了动态连接库(DDLs)的模型，并将其成员用于 ObjectWindows 应用程序(通过它的子类 TApplication)。下面是 TModule 类的声明：

```
class _EXPORT TModule : public Object {
public:
    // Lib and WinMain args
```

```
    HINSTANCE hInstance;
    LPSTR   lpCmdLine;
    int Status;
    LPSTR Name;
    TModule(LPSTR AName, HINSTANCE AnInstance, LPSTR ACmdLine);
    virtual ~TModule();
BOOL LowMemory();
void RestoreMemory();
virtual PTWindowsObject ValidWindow(PTWindowsObject AWindowsObject);
virtual PTWindowsObject MakeWindow(PTWindowsObject AWindowsObject);
virtual int ExecDialog(PTWindowsObject ADialog);
HWND GetClientHandle(HWND AnHWindow);
virtual PTWindowsObject GetParentObject(HWND ParentHandle);
virtual void Error(int ErrorCode);
// define pure virtual functions derived from Object class
virtual classType isA() const

    { return moduleClass; }
virtual Pchar nameOf() const
    { return "TModule"; }
virtual hashValueType hashValue() const
    { return hashValueType(hInstance); }
virtual int isEqual(RCObject module)  const
    { return (hInstance == ((RTModule)module).hInstance); }
virtual void printOn(Rostream outputStream) const
    { outputStream << nameOf() << "{ hInstance = "
        << hInstance << " }\n"; }
};
```

如果创建了DDL(设置了__DLL__)，由将_EXPORT宏扩展为__EXPORT。否则，EXPORT宏将扩展为_CLASSTYPE。

类构造函数创建了Tmodule的一个实例，其参数为应用程序/DDL名、应用程序/DDL句柄和命令行。

成员函数LowMemory和RestoreMemory完成内存管理任务。MakeWindow成员函数创建一个窗口，该过程包括使对象有效的ValidWindow成员函数。在成员函数ValidWindow验证了被激活对象的合法性之后，ExecDialog成员函数执行一个模式对话框。ExecDialog函数与被激活对话框的Execute成员函数一起工作。还要注意isA和nameOf成员函数分别返回moduleClass的枚举值和"TModule"串。

1.3.3 TApplication类

TApplication类是TModule类的子类，也是你所开发的每个ObjectWindows应用程序的父类。TApplication的成员和由TModule继承来的成员一起提供了基本的数据成分以及操作以支持最小的Windows应用程序。这种应用程序给出一个窗口的特征，并具有一个最小的命令集。

下面清单给出TApplication类的声明：

试读结束，需要全本PDF请购买 www.ertongbook.com