Windows 95 程序设计

——OWL 篇

蔡明志　编著

清华大学出版社

目　录

第 1 章　从一简单的范例谈起 …… 1
1-1　什么是 ObjectWindows? …… 1
1-2　ObjectWindows 的优点 …… 1
1-3　如何学好 ObjectWindows 程序设计 …… 2
1-4　给您一个建议 …… 2
1-5　ObjectWindows 框架图 …… 3
1-6　头文件(headerfiles) …… 5
1-7　ObjectWindows 的资源文件 …… 9
1-8　一个简单的范例 …… 10
程序实习 …… 15

第 2 章　弹出式窗口与子窗口的制作 …… 16
2-1　弹出式窗口的建立 …… 16
2-2　有无父窗口的弹出式窗口 …… 19
2-3　如何建立一子窗口 …… 23
2-4　如何建立多个子窗口 …… 28
2-5　有父窗口的弹出式窗口与子窗口的比较 …… 32
程序实习 …… 38

第 3 章　Windows 消息的处理 …… 39
3-1　WM_PAINT 消息的处理 …… 39
3-2　WM_SIZE 消息的处理 …… 42
3-3　鼠标消息的处理 …… 46
3-4　键盘与字符消息的处理 …… 50
3-5　滚动条相关的消息处理 …… 57
程序实习 …… 62

第 4 章　鼠标与键盘的应用 …… 63
4-1　命中测试 …… 63
4-2　观察系统的特征值 …… 69
程序实习 …… 77

第 5 章　计时器消息及其应用 …… 78
5-1　计时器消息应用之一 …… 78
5-2　计时器消息应用之二：监视内存 …… 82
5-3　计时器消息应用之三：制作一个数字时钟 …… 87
程序实习 …… 94

第 6 章　菜单的制作与应用 …… 95
6-1　如何制作一菜单 …… 95
6-2　浮动式菜单的建立 …… 114
6-3　改变原来的系统菜单 …… 123
6-4　菜单与加速键的结合 …… 130
程序实习 …… 139

第 7 章　其它 Windows 的资源 …… 141
7-1　图标的运用 …… 141
7-2　光标的运用 …… 147
7-3　位图的应用 …… 154
7-4　利用位图制作菜单 …… 160
7-5　制作各式各样的插入符 …… 169
7-6　字符串资源的应用 …… 178
程序实习 …… 183

第 8 章　窗口控制元的制作 …… 184
8-1　编辑框的制作 …… 184
8-2　各种按钮的制作 …… 190
8-3　滚动条控制元的制作 …… 197
8-4　如何制作列表框(一) …… 207
8-5　如何制作列表框(二) …… 213
8-6　组合框控制元的制作 …… 218
程序实习 …… 224

第 9 章　对话框 …… 225
9-1　有模式对话框的制作(一) …… 225
9-2　对话框与主窗口之间数据的传送 …… 230
9-3　对话框数据的核对 …… 238
9-4　有模式对话框的制作(二) …… 264
9-5　无模式对话框的制作 …… 277

程序实习…… 293

第 10 章　共用对话框 …… 294
10-1　输入字符串对话框 …… 294
10-2　打开文件与保存文件共用对话框 …… 299
10-3　字型共用对话框 …… 308
10-4　色彩共用对话框 …… 314
10-5　查询与替换共用对话框 …… 320
程序实习…… 325

第 11 章　工具条 …… 327
11-1　工具条 …… 327
11-2　消息条 …… 336
11-3　状态栏 …… 349
11-4　进一步应用 …… 361
程序实习…… 381

第 12 章　工具箱 …… 382
12-1　如何建立一固定式的工具箱 …… 382
12-2　如何建立一浮动式的工具箱 …… 396
程序实习…… 410

第 13 章　MDI …… 411
13-1　MDI 的第一个范例 …… 411
13-2　MDI 的第二个范例 …… 416
程序实习…… 421

第 14 章　打印机 …… 422
14-1　OWL 下的 TPrinter 与 TPrintout …… 422
14-2　另一种打印方式 …… 436
程序实习…… 448

第 15 章　剪贴板 …… 449
15-1　剪贴板概况 …… 449
15-2　剪贴板与文字数据格式 …… 450
15-3　剪贴板与图形数据格式 …… 460
程序实习…… 472

第 1 章　从一简单的范例谈起

在未进入本章的主题之前，先来介绍本书所探讨的 ObjectWindows，看看它是什么东西？有何优点？然后再以一简单的范例说明以 ObjectWindows Library（简称 OWL）所写出来的程序结构。

1-1　什么是 ObjectWindows？

ObjectWindows 是 Borland 公司将 Windows API（应用程序接口）加以重新包装集成的一套软件应用框架（Application Framework），此应用框架在包装上皆是经过专家们的精心雕琢而成的。Windows 程序设计者皆晓得 Windows 应用程序好比 COBOL 一般，会让人觉得它是那么的冗长而乏味。就以设计一重叠式窗口（OVERLAPPED WINDOW）来说，您会觉得以 ObjectWindows 所写出的程序码简短且执行速度快。

ObjectWindows 在 1.0 版时，笔者即已在钻研了，当时发现此应用框架不是很好，且必须设定一堆的路径（path），导致使用者不知所措，因此在当时并未引起太多人的实际应用（但欲尝试的人很多）。之后，Borland 继 Borland C++ 4.0 版所附的 ObjectWindows 2.0 就像样多了，截至目前 Borland C++ 4.5 的 ObjectWindows 2.5 让您挑剔的地方就很少了。题外话—— 有一次笔者替 Borland 台湾分公司所办的 Borland C++ 4.0 发表会上，意外发现许多软件公司或信息部门采用 Borland's ObjectWindows 来开发软件，竟比笔者估计的数目来得多。

1-2　ObjectWindows 的优点

ObjectWindows 是属于应用框架，而应用框架（即 ObjectWindows）的优点如下：

1. 程序码少

由于 ObjectWindows 各个类（class）经过专家将它们加以组合包装，而且具有 OOP（面向对象程序设计）的性质，更能应用继承的特性，使之程序码减少。

2. 执行速度快

程序码减少，相对地执行速度也加快了。

3. 有标准的应用框架

在 ObjectWindows 中，各种类的分类及从属的关系，不是随意拼凑而成的，而是有一定的标准加以包装，因此学会了一种应用框架，如 ObjectWindows 的 OWL，您便可以很容易地踏入另一应用框架如 Microsoft 的 MFC 等等。

4. 提供更丰富的及更有效率的功能

此功能包括工具条(Control Bar)、消息栏(Message Bar)、状态栏(Status Bar)、工具箱(Tool Box)及更有效率的 MDI…等等。

5. 具有 C++的 Windows API 的特性

C++是面向对象程序设计所使用的程序语言，众所周知，面向对象程序设计的优点——大大地降低维护成本，此一优点不但给软件公司带来福音，甚至对整个信息软件工业也是一大冲击，因为"软件的 IC"为期不远了。

1-3 如何学好 ObjectWindows 程序设计

如同 Windows 程序设计一样，因为 ObjectWindows 也是属于 Windows 下的程序设计，读者需掌握程序中每一类之间的关系，如 A 类继承了 B 类后，A 类此时有哪些数据成员或成员函数可利用，随时注意每一类的构造函数(construction)、析构函数(destructor)、成员函数的原型(prototype)是什么，才有办法和它沟通。请随时查阅本章所附的 ObjectWindows 框架图。

还有，必须随时注意 Windows 的消息(Message)是如何产生的，这一点非常的重要，而设计者如何响应 Windows 所送过来的 Windows 信息，这些信息大致上类似 Windows 基本程序设计的原理，只不过 ObjectWindows 利用了某些技巧或间接的方式处理之，此点往后笔者会加注释的。

最后注意 ObjectWindows 一些新增的函数或处理方式，这也是 ObjectWindows 在设计世界级软件时的利器，留待往后的章节再一并讨论之。

1-4 给您一个建议

学习 ObjectWindows 程序设计，笔者认为读者若具备了下列两个条件后，在学习的效果上，您会有事半功倍的效果：

1. 具备有面向对象程序设计(Object-Oriented Programming，OOP)的基础，如 C++，因为在 ObjectWindows 中所有的类(class)皆已为我们分门别类，并且每一类与另一类的关系，如继承，专家们也为我们设计好了，所以读者可以轻松地看懂每一类之

间的关系是什么，从而应用更多的基础类的资源。请参阅有关C++的书籍及第1-5节有关ObjectWindows的框架图。

2. 最好具备有Windows程序设计的基础，因为ObjectWindows乃是应用了大部分Windows API再加上一些函数而成的程序库，通称为ObjectWindows Library(简称OWL)。读者若有此项基础后，则往后便可省下大部分的时间，去注意ObjectWindows在程序处理的技巧，而不必花太多的时间去注意Windows API的用法。

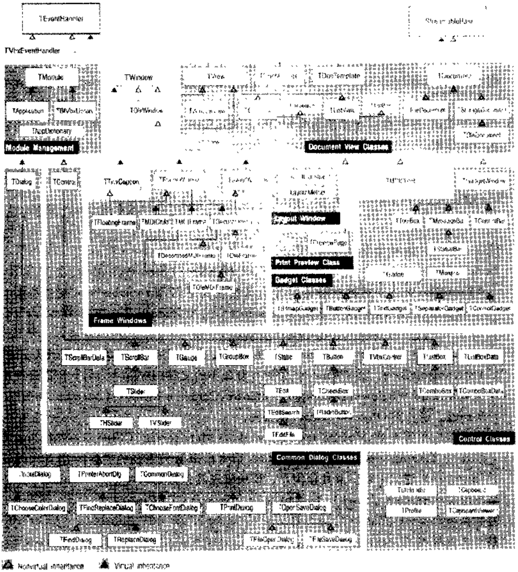
当然，也许其他先进人士及专家不以为然，认为不必有上述的基础一样可以走入ObjectWindows的殿堂，没错，也可以，据笔者的教学经验，有上述的两项基础比没有基础的同学更易掌握重点，并可撰写出更佳结构的ObjectWindows程序，这是笔者的浅见，尚请各位同仁及专家指教。

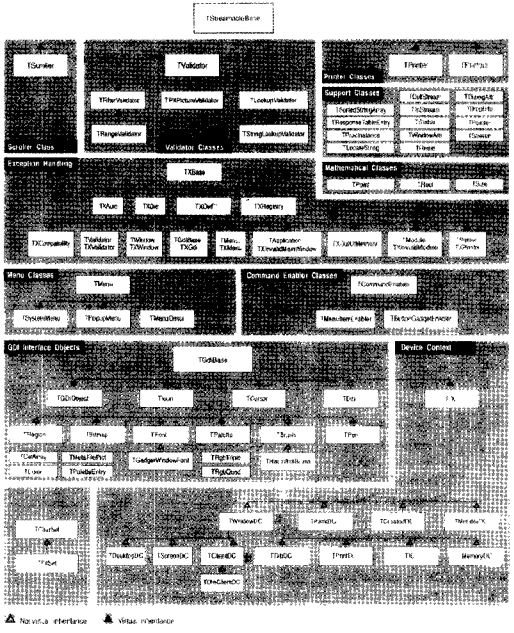
1-5 ObjectWindows框架图

ObjectWindows经由专家的精心设计，大致分为下列几类：

- ■基类(Base Class)：此类分别为TEventHandler、TStreamableBase及TGdiBase。TEventHandler传送消息给适当的消息处理程序；TStreamableBase支持有关C++的数据流(Stream)及持续性(persistence)；而TGdiBase为一支持Window's GDI程序库。请参阅ObjectWindows框架图。
- ■窗口管理类(Window management classes)：此类大致皆继承TEventHandler及TStreamableBase。而TWindow为所有窗口类的父类(parent class)。窗口管理类包含有Frame Windows、Decorated Windows、Common dialogs、Controls、Gadgets及Menus。请参阅ObjectWindows框架图。
- ■模块与应用管理类(Module and Applicatioon Management Classes)：TModule类衍生出一TApplication。从TApplication衍生出来的类主要的功能为建立类的实例(instance)，建立主窗口及处理消息。模块管理类包括Doc/View和Printer。
- ■图形类(Graphics classes)：此类包括DC Class及GDI Classes。
- ■数据核对类(Validators)。
- ■异常处理类(Exception handling classes)。
- ■支持类(Support class)：包含的项目甚多，请参阅ObjectWindows框架图。
- ■数学类(Mathematical classes)，主要包含有TPoint、TRect及TSize。

下图是ObjectWindows框架图的全貌，读者在学习OWL的过程中，需常常翻阅，以了解类之间的继承关系为何，其中图中的△表示非虚拟继承(nonVirtual inheritance)，而▲表示虚拟继承(Virtual inheritance)。


TEventHandler
TVbxEventHandler
TModule
TWindow
TApplication
TAppDictionary
Module Management
TView
TDocTemplate
TListView
Document View Classes
TDialog
TControl
TFloatingFrame
TDecoratedMDIFrame
TOleMDIFrame
Layout Window
TPreviewPage
Print Preview Class
TStatusBar
TGadget
Gadget Classes
TButtonGadget
TSeparatorGadget
TControlGadget
Frame Windows
TScrollBarData
TScrollBar
TGauge
TGroupBox
TStatic
TButton
TVbxControl
TListBox
TListBoxData
TSlider
TEdit
TCheckBox
TComboBox
THSlider
TVSlider
TEditSearch
TRadioButton
TEditFile
Control Classes
Common Dialog Classes
TPrinterAbortDlg
TCommonDialog
TChooseColorDialog
TFindReplaceDialog
TChooseFontDialog
TPrintDialog
TOpenSaveDialog
TFindDialog
TReplaceDialog
TFileSaveDialog
TProfile
TClipboardViewer
Virtual inheritance



1-6 头文件(headerfiles)

ObjectWindows 提供了许多头文件,这些头文件皆放在\INCLUDE\OWL 目录下。头

文件里包含了类的成员函数的语法、数据类型的定义及符号常数。下面列出，当程序中使用了哪一个类时，需将其对应的头文件装入到程序中。

头文件	类	使用时机
appdict. h	TAppDictionary	应用程序与过程 ID 之间的消息。
Applicat. h	TApplication	控制所有 ObjectWindows 应用程序的基本运行。
bitmapga. h	TBitmapGadget	不超过 256 位的字节。
bitset. h	TBitSet	设定或解除一个或多个位。
	TCharSet	一组字符
button. h	TButton	建立不同类型的按钮控制元。
buttonga. h	TButtonGadget	建立可开/关的按钮装饰图样。
celarray. h	TCelArray	建立一个图样的数组。
checkbox. h	TCheckBox	表示检查框控制元。
chooseco. h	TChooseColor	表示一选择色彩的排父对话框。
choosefo. h	TChooseFont	表示一选择字型的排父对话框。
clipboar. h	TClipboard	剪贴板数据控制的函数。
clipview. h	TClipboardViewer	登记一 TClipboardViews 当作剪贴板的浏览。
color. h	TColor	简单的标准的 Windows 颜色运行函数。
combobox. h	TCombobox	建立一组合框或组合框控制元及组合框之间数据交换的类 TComboBoxData。
commdial. h	TCommonDialog	共用对话框的对象。
compat. h		定义 ObjectWindows 内部使用的函数和常数。
control. h	TControl	在衍生类建立一些控制(Control)对象。
controlb. h	TControlBar	建立一些控制条元件。
controlg. h	TControlGadget	在装置窗口中安置一些控制元件。
dc. h	TBandInfo, TClientDC, TCreatedDC, TDC, TDesktopDC, TDibDC TIC, TMemoryDC, TMetaFileDC, TPaintDC TPrintDC, TScreenDC, TWindowDC	建立 GDI 的 DC 对象。
decframe. h	TDecoratedFrame	建立一装置的 Client Window。
decmdifr. h	TDecoratedMDIFrame	建立一 MDI 性质的装置子窗口。
dialog. h	TDialog	建立排父和共存对话框的界面元素。
	TDialogAttr	对话框元素的属性。
dispatch. h		定义发送消息函数。
docmanag. h	TDocManager	建立一管理文件和样板(templates)的文件管理对象。
doctpl. h	TDocTemplate	建立样板。
	TDocTemplateT	登记文档(document)和视图(View)的关系。
docview. h	TDocument, TView, TWindowView, TStream,	有关文档视图的建立、消除及传送的信息。

	TInStream,TOutStream	定义文件的数据流。
edit.h	TEdit	建立一个编辑控制界面的元素。
editfile.h	TEditFile	建立一个文件编辑窗口。
editsear.h	TEditSearch	建立一个可以查询和替换命令的编辑控制元。
editview.h	TEditView	提供 TEdit 类视图。
eventhan.h	TEventHandler	用于掌管信息的衍生类。
except.h	TXBase	ObjectWindows 与 ObjectComponents 类的异常处理基类。
	TXOwl	ObjectWindows 与 ObjectComponents 类的异常处理基类。
	TXCompatibility	ObjectWindows 类的异常处理基类。
	TXOutOfMemory	ObjectWindows 类的异常处理基类。
	TStatus	包含后续的相容性,描述内存不足和状态的异常。
filedoc.h	TFileDocument	打开和关闭文件的视图(Views)。
findrepl.h	TFindDialog	定义一些有关查询和替换命令的排父对话框属性。
	TFindReplaceDialog;;	定义一些有关查询和替换命令的排父对话框属性。
floatfra.h	TFloatingFrame	浮动式的窗口。
framewin.h	TFrameWindow	控制特定窗口的行为,如键盘与命令处理。
gadget.h	TGadget	建立一装置窗口的对象。
gadgetwi.h	TGadgetWindow	维护一串装置式的窗口。
	TGadgetWindowFont	定义使用于装置窗口的字型。
	TSeparatorGadget	有关装置之间的距离。
gauge.h	TGauge	计量的控制。
gdibase.h	TGdiBase	所有 GDI 类的基类。
gdiobjec.h	TGdiObject	GDI 的基类。
	TPen,TBrush,TFont	建立 GDI 对象的类。
	TPalette,TBitmap,	
	TIcon,TCursor,TDib,	
	TRegion	
geometry.h	TPoint,TSize,TRect	数学类。
	TDropInfo	支持拖放动作。
	TProcInstance	Win16 支持类。
	TPointer	提供指针的操作。
	TResId	建立一资源 ID。
groupbox.h	TGroupBox	组合框的类。
inputdia.h	TInputDialog	接受一字串的输入对话框。
layoutco.h	TLayoutConstraint	建立布局的限制式。
Layoutwi.h	TLayoutMetrics	布局的矩阵。
	TLayoutWindow	布局的窗口。
listbox.h	TListBox	建立一列表框类。
	TListBoxData	转换列表框的内容。
listview.h	TListView	提供列表框的视图功能。
locale.h	TLocaleString	局部化的字符串替换。
mdi.h	TMDIClient	管理 MDI 的子窗口。

	TMDIFrame	MDI 的主窗口。
mdichild. h	TMDIChild	定义 MDI 子窗口的行为。
menu. h	TMenu，TPopupMenu，	建立菜单对象、菜单基类及菜单的说明。
	TSystemMenu	系统菜单。
	TMenuDescr	菜单的描述。
messageb. h	TMessageBar	消息条。
metafile. h	TMetaFilePict	TMetaFileDC 类。
module. h	TModule	定义 ObjectWindows 函数库与应用程序的一些基本操作行为。
oledoc. h	TOleDocument	Ole 文件类。
olefacto. h	TAutoFactory< >	样板类。
olefacto. h	TOleFactory< >	样板类。
oleframe. h	TOleFrame	支持 OLE 的 SDI 元窗口。
olemdifr. h	TOleMDIFrame	支持 OLE 的 MDI 元窗口。
oleview. h	TOleView	支持 OLE 视图。
olewindo. h	TOleClientDC	两个不同坐标系统的转换。
opensave. h	TOpenSave	文件的打开与保存类。
owlall. h		所有 ObjectWindows 类的装入文件。
owlcore. h		ObjectWindows 核心类的装入文件。
owldefs. h		ObjectWindows 宏指令的定义。
owlpch. h		包含使用于 ObjectWindows 的宏定义、数据和函数。
preview. h	TPreviewPage	预览功能。
	TPrintPreviewDC	映射打印机装置坐标到屏幕坐标。
printdia. h	TPrintDialog	打印设定对话框。
printer. h	TPrinter	表示打印机装置。
	TPrintout	表示打印的文件。
	TPrinterAbortDlg	表示中止打印的对话框。
radiobut. h	TRadioButton	建立一个单选按钮。
scrollba. h	TScrollBar	表示垂直或水平滚动条。
	TScrollBarData	有关滚动条滚动方块的数值。
scroller. h	TScroller	滚动条控制元。
signatur. h		使用于 ObjectWindows 事件处理函数的信息。
slider. h	TSlider	定义滚动条的基本动作。
	THSlider	水平滚动条。
	TVSlider	垂直滚动条。
static. h	TStatic	静态控制元。
statusba. h	TStatusBar	状态条。
stgdoc. h	TStorageDocument	支持复合文件机制。
textgadg. h	TTextGadget	建立文字装饰对象。
tinycapt. h	TTinyCaption	小小的标题栏。
toolbox. h	TToolBox	工具箱。
uihandle. h	TUIHandle	定义 UI 代码。

validate.h	TValidator,	核对基类。
	TPXPictureValidator	文字核对功能。
	TFilterValidator	过滤核对功能。
	TRangeValidator	区间核对功能。
	TLookupValidator	查询核对功能。
	TStringLookupValidator	文字查询核对功能。
vbxctl.h	TVbxControl	VBX 控制元。
	TVbxEventHandler	VBX 事件代码。
	TBIVbxLibrary	装入及启动 BIVBXxx.DLL。
version.h		ObjectWindows Library 的版本。
window.h	TWindow	封装 Windows API 函数。
windowev.h		定义 Windows 信息事件代码及响应表格宏。

1-7 ObjectWindows 的资源文件

ObjectWindows 资源文件定义了资源(resource)及选项指令的 ID,而这些资源文件所在的目录为\INCLUDE\OWL。

资源文件	用　　途
docview.rh	用于 docview.h 与 docview.rc
edit.rh	用于 edit.h
editfile.rh	用于 editfile.rc 与 editfile.h
editsear.rh	用于 editsear.rc 与 editsear.h
editview.rh	用于 TEditView
except.rh	用于 except.h 与 except.rc
inputdia.rh	用于 inputdia.rc 与 inputdia.h
listview.rh	用于 listview.h
locale.rh	用于定义区域性的资源 ID
mdi.rh	用于 mdi.h
oleview.rh	用于 OLE 有效时
printer.rh	用于 printer.rc 与 printer.h
slider.rh	用于 slider.h
statusba.rh	用于 statusba.h
validate.rh	用于 TValidator 与其衍生类
window.rh	用于 window.h

1-8 一个简单的范例

接下来看一个简单的范例，此范例试图建立一标准的窗口，即重叠式窗口，此窗口包括系统菜单框(System Box)、标题栏(Caption Bar)例中为 Skeleton of Window、最大化按钮(Maximum Box)、最小化按钮(Minimum Box)及一工作区域(Client Area)。程序清单如下：

EX1-1.CPP

```
//////////////////////////////////////////////////////////////
// EX1-1.CPP: Demonstrates the process of compiling a
//            ObjectWindows program using Borland C++ 4.5
//////////////////////////////////////////////////////////////
#include <owl\applicat.h>
#include <owl\framewin.h>

// My application class.
class MyApp : public TApplication
{
protected:
   TFrameWindow *mainwndw;
public:
   MyApp(): TApplication() {}
   void InitMainWindow();
};

// My main window class.
class MyMainWndw : public TFrameWindow
{
public:
   MyMainWndw(TWindow *parent, const char far *title):
              TFrameWindow(parent, title) {}
};

//////////////////////////////////////////////////////////////
// TMyApp::InitMainWindow()
// This function creates my application's main window.
//////////////////////////////////////////////////////////////
void MyApp::InitMainWindow()
{
   mainwndw = new MyMainWndw(0, "Skeleton of Window");
   SetMainWindow(mainwndw);
}
```

```
//////////////////////////////////////////////////////////////
// OwlMain()
// This function run my application.
//////////////////////////////////////////////////////////////
int OwlMain(int, char*[])
{
   return MyApp().Run();
}
```

程序解说

在此范例程序中包含两个类，一为 MyApp；一为 MyMainWndw。此两个类分别继承了 TApplication 和 TFrameWindow 类，而有关 TApplication 和 TFrameWindow 类的位置请参阅本章 1-5 节的 ObjectWindows 框架图。一般而言，我们皆利用 TFrameWindow 类当作主窗口的基类(base class)，而 TApplication 做为应用程序的基类，通常这两个基类是撰写 OWL 程序不可缺少的。

值得一提的是在 MyApp 类中，有一指向 TFrameWindow 类的指针变量 mainwndw，主要作为建立一主窗口时使用之。还有一个 public 的成员函数为 InitMainWindow，主要的功能为建立一主窗口，其标题栏文字为 Skeleton of Window。其实 InitMainWindow 为 TApplication 类下的 protected 成员函数，在此我们将它重新定义(Redefine)，利用 SetMainWindow 建立一主窗口，SetMainWindow 函数必须接受一指向 TFrameWindow 的指针变量，它是属于 TApplication 类下的 public 成员函数，其语法为：

```
TFrameWindow * SetMainWindow(TFrameWindow * Window);
```

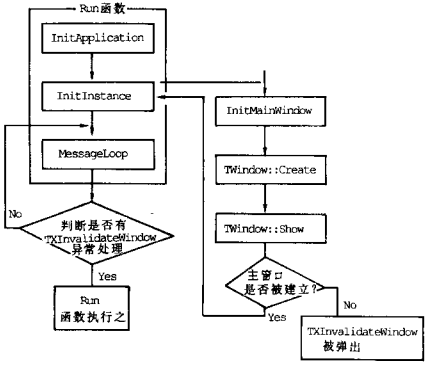
OwlMain 函数为 OWL 的主程序，也是程序的进入点，此函数只执行 TApplication 类下的 Run 函数。Run 函数主要的功能为：

1. 首先调用 TApplication 下的 InitApplication 函数，替第一个实例(Instance)做初始化的工作。
2. 继而调用 InitInstance 函数，执行所有实例的初始化工作。
3. 假使 1，2 项的初始化工作皆成功地完成，Run 函数继而调用消息循环函数(MessageLoop)及执行此应用程序。

其中第 2 项的 InitInstance 函数又会调用 InitMainWindow 函数(此函数旨在建立 TframeWindow 对象)。然后利用 TWindow::Create 及 TWindow::Show 来建立和显示主窗口。假使主窗口无法建立时，则弹出 TXInvalidateWindow 异常处理。当消息循环外有异常处理时，Run 函数则会撷取此异常处理。范例程序中的 InitMainWindow 函数就是由 InitInstance 所调用的。

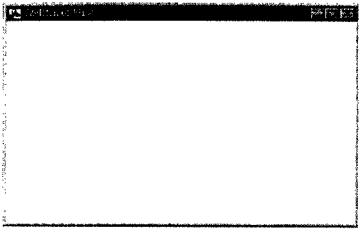
我们将上述所讨论的函数以下图表示之：

范例程序装入了 applicat.h 和 framewin.h 头文件，这是因为此程序中运用了 TApplication 和 TFrameWindow 类，在以后的程序中，读者也需随时注意程序中装入了哪些头文



件，而有关 OWL 所提供的这些头文件的详细说明，请读者利用联机帮助(on-line help)就可明了。

输出结果



EX 1-1　输出结果

接下来再利用第一个范例加以扩充，建立多个标准窗口，也借此范例说明 InitInstance 函数的用法，程序清单如下：

EX1-2.CPP

```
//////////////////////////////////////////////////////////////
// EX1-2.CPP: Create many windows based on the same
//            TFrameWindow class.
//////////////////////////////////////////////////////////////

#include <owl\applicat.h>
#include <owl\framewin.h>

// My application class.
class MyApp : public TApplication
{
protected:
   TFrameWindow *mainwndw;
public:
   MyApp() : TApplication() {}
   void InitMainWindow();
   void InitInstance();
};

// My window class.
class MyWndw : public TFrameWindow
{
public :
   MyWndw(TWindow *parent, const char far *title):
          TFrameWindow(parent, title) {};
};

//////////////////////////////////////////////////////////////
// MyApp::InitMainWindow()
// This function creates my application's main window.
//////////////////////////////////////////////////////////////
void MyApp::InitMainWindow()
{
   mainwndw = new MyWndw(0, "Window Number 0");
   SetMainWindow(mainwndw);
}

//////////////////////////////////////////////////////////////
// MyApp::InitInstance()
// This function creates and show my application's 7 window
// without parent.
//////////////////////////////////////////////////////////////
void MyApp::InitInstance()
```