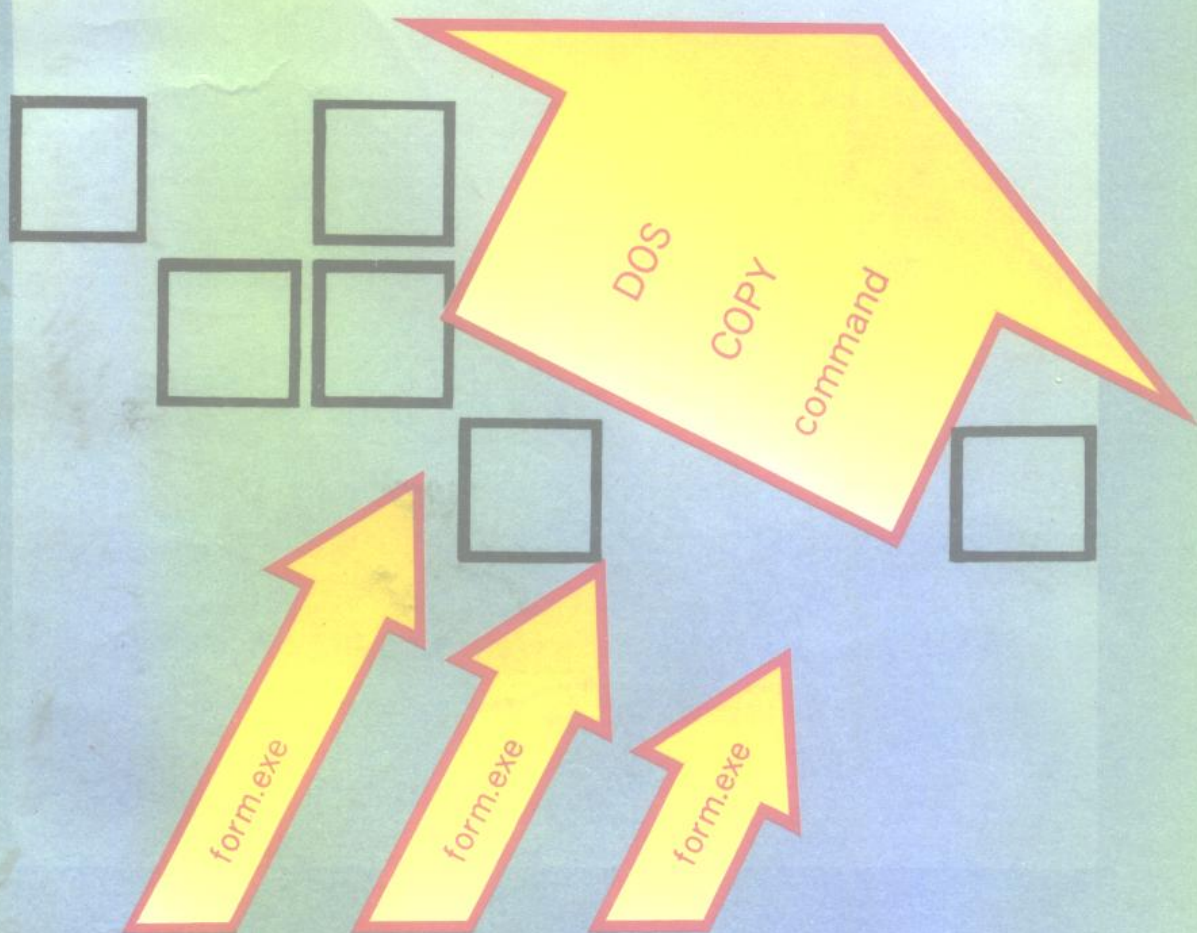


最新 Borland C++ 实用教程 2

Borland C++ 进阶

求实编著



科学出版社

2
最新 Borland C ++ 实用教程 2

Borland C ++ 进阶

求 实 编 著

科 学 出 版 社

1994

(京)新登字 092 号

JS193/18
内 容 简 介

本书论述最新 Borland C++ 高级程序设计技术,包括 C++ 基础知识、高级知识、预处理技术、覆盖技术、嵌入式汇编语言、C++ 流、Container 类库,以及如何和 C++ 建立 Windows 应用程序,并给出了一些 Borland C++ 程序设计实例。对面向对象程序设计技术(Dos 下和 Windows 下)如继承窗口概念、菜单、对话框、弹出窗口、按钮、输入框等程序设计技术,此外,本书还细致地介绍了帮助用户找出影响程序执行效率的剖析器。

本书可供软件专业的程序设计、开发人员和高等院校计算机专业师生参考。

最新 Borland C++ 实用教程 2

Borland C++ 进阶

求实 编著

责任编辑 杨家福

科学出版社出版

北京东黄城根北街 16 号

邮政编码:100717

北京外国语学院印刷厂印刷

新华书店北京发行所发行 各地新华书店经售

1994 年 5 月第 一 版 开本:787×1092 1/16

1994 年 5 月第一次印刷 印张:27 1/4

印数:1—5 000 字数:622 400

ISBN 7-03-004043-0/TP·341

定价:26.00 元

前 言

本书是“Borland C++ 实用教程”之二，主要讨论用 Borland C++ 设计 Windows 应用程序的方法和技巧。我们知道，作为一种方便、实用的 C/C++ 语言编译程序，Borland C++ 可以构造运行于传统 DOS 环境下的 C/C++ 语言程序。但是，随着 MS Windows 窗口图形环境的推出，Borland C++ 的主要目的已转向开发基于 Windows 的应用程序。原因是，Borland C++ 以对象作为程序单元，而 Windows 应用程序也采用面向对象的风格。例如，Windows 中的窗口就是一个对象。其次，用 Borland C++ 集成环境开发 Windows 应用程序比用 MS C/C++ 7.0 更方便、更有效。

本书按开发 Windows 应用程序的一般步骤进行组织，而且假定读者已经熟悉 Borland C++ DOS 程序设计的基本方法（这也是“Borland C++ 实用教程”之一的主要内容）。对于初学 Windows 程序设计的人，可能会因一开始就要接触到许多全新的概念而不知所措。为此，本书的前两章用一些简单的 Windows 程序实例引入这些概念，并在此基础上讲述 Windows 应用程序的组成、开发步骤和构造方法。第三章到第五章则讨论设计 Windows 程序时所要考虑的界面对象，如窗口、菜单和对话框等。有了这些知识，读者就可以尝试着用 Borland C++ 开发简单的 Windows 应用程序了。但是，这几章的内容还没有充分利用 Borland C++ 对 Windows 的支持，即 Borland C++ 中固有的专用于 Windows 的类体系。这正是第七章到第十五章的主题。这几章着重讨论 ObjectWindows 程序设计，内容涉及界面、窗口、模块、对话框、控制、多文档界面 (MDI) 及流等预定义的对象。应尽量利用这些对象进行编程，而不是自己重复开发代码来实现现有的功能，因为它们可以极大地减少开发时间和编码的复杂性。在已编制出程序代码后，可利用第十六章的内容在集成环境或命令行上编译和连接各程序模块，并利用第十七章的内容加入资源，以生成可执行的 Windows 应用程序。

“动态链接库 (DLL)”是迄今为止尚未涉及到的一个陌生概念，因为它是 Windows 特有的。作为相对独立的一部分，第十八章给出了对 DLL 的完整解释。接下来的两章是一些程序实例；读者可以模仿这些实例的程序结构和设计风格。最后四章讨论如何调试和剖析用 Borland C++ 开发出的 Windows 程序。

值得指出的是，Windows 是一个复杂的窗口图形环境，限于篇幅，本书无法涉及它的全部内容，如 Windows 内存管理、图形设备接口 (GDI)、动态数据交换 (DDE) 等。因为这些内容是特定于 Windows 的，与本书的主题非直接相关，有关这些内容，请读者参考其它相关书籍。

前两章由李波、朱志勇、朱尽染执笔，第三到第六章由谢宇、阎继忠、何晓珊执笔，接下来的九章由何大庆、覃社庭、魏泱、何国辉、王勇、韩亮执笔，第十六到第十八章由魏宁、李梅执笔，第十九章到第二十章由刘秀琳、徐伟、张荣军执笔。最后四章由张勇、欧阳琼执笔。诗颖、白琴为本书的编排付出了辛苦的劳动。在此谨对以上同志表示感谢。

目 录

第一部分 ObjectWindows 初步

第一章 继承窗口	2
1.1 何谓 Windows 应用程序	2
1.1.1 Windows 的优点	2
1.1.2 需求	3
1.2 面向对象的窗口编程	3
1.2.1 一个较好的 Windows 接口	3
1.3 Windows 程序的结构	7
1.3.1 Windows 的结构	7
1.3.2 与 Windows 和 DOS 的交互作用	8
1.4 “Hello, Windows”	8
1.4.1 应用程序的启动步骤	8
1.4.2 主窗口的职责	9
1.5 应用程序开发步骤	9
第二章 Windows 基础	10
2.1 编制一个 ObjectWindows 应用程序: 预备知识	10
2.1.1 容器类库	10
2.1.2 目录	10
2.1.3 指定正确的库	11
2.1.4 生成资源文件	12
2.2 建立 ObjectWindows 应用程序	13
2.2.1 使用 IDE 建立 ObjectWindows 应用程序	13
2.2.2 使用 Borland C++ 命令行工具建立 ObjectWindows 应用程序	13
2.3 STEP1.CPP: 一个简单的 Windows 应用程序	14
2.3.1 应用程序要求	14
2.4 STEP2.CPP: 主窗口类	16
2.4.1 何谓窗口对象	16
2.4.2 生成主窗口对象	17
2.4.3 响应消息	18
2.4.4 终止应用程序	19
第三章 填充窗口	21
3.1 显示上下文	21
3.2 STEP3.CPP: 在窗口中绘制文本	22
3.2.1 消息结构	22
3.2.2 清屏	23
3.3 STEP4.CPP: 在窗口中画线	23

3.3.1	拖拽模型	24
3.3.2	响应拖拽消息	24
3.4	STEP5.CPP: 改变线宽	26
3.4.1	选择新的画笔	26
3.4.2	运行输入对话框	28
3.5	STEP6.CPP: 绘制图形	29
3.5.1	绘图模式	29
3.5.2	把图形当作对象来存储	29
3.5.3	重绘存储的图形	31
第四章	增加菜单	33
4.1	菜单资源	33
4.2	STEP7.CPP: 主窗口菜单	34
4.3	解释菜单消息	35
4.4	响应菜单消息	36
第五章	增加对话框	37
5.1	STEP8.CPP: 增加一个弹出式窗口	37
5.1.1	生成弹出式窗口	38
5.1.2	MakeWindow 函数	38
5.2	增加一个对话框	38
5.2.1	增加一个数据成员	39
5.2.2	运行对话框	39
5.3	STEP9.CPP: 存储图形到文件中	40
5.3.1	状态管理	40
5.3.2	打开和存储文件	42
第六章	弹出窗口	44
6.1	STEP10.CPP: 弹出一个帮助窗口	44
6.2	使用 ObjectWindows 中的模块	44
6.2.1	修改主程序	44
6.2.2	建立 HELPWIND 模块	44
6.3	增加控制到窗口中	46
6.3.1	何谓控制	46
6.3.2	建立窗口控制	46
6.3.3	作为数据成员和控制对象	47
6.3.4	管理控制	47
6.3.5	对控制事件的响应	49

第二部分 使用 ObjectWindows

第七章	概述	52
7.1	ObjectWindows 的约定	52
7.2	ObjectWindows 的类体系	53
7.2.1	Object	53

7.2.2	TModule	53
7.2.3	TApplication	53
7.2.4	界面对象	54
7.2.5	窗口对象	54
7.2.6	对话框对象	54
7.2.7	控制对象	54
7.2.8	MDI 对象	55
7.2.9	滚动条对象	55
7.3	Windows API 函数	56
7.3.1	ObjectWindows 调用 Windows 函数	56
7.3.2	对 Windows 函数的访问	56
7.3.3	组合类型常量	57
7.3.4	Windows 函数类型	57
7.3.5	回调函数	57
7.4	Windows 消息	59
7.4.1	Windows 消息参数	60
7.4.2	Windows 的消息类型	60
7.5	缺省的消息处理	61
7.6	发送消息	61
7.7	消息值域	62
7.8	用户定义的消息	62
第八章	模块和应用程序对象	64
8.1	应用程序流程	64
8.2	初始化应用程序	65
8.2.1	初始化主窗口	65
8.2.2	初始化各个应用程序事例	66
8.2.3	初始化第一应用程序事例	67
8.3	运行应用程序	68
8.4	关闭应用程序	68
第九章	界面对象	70
9.1	TWindowsObject	70
9.2	为何要用界面对象	70
9.3	窗口父/子关系	71
9.3.1	子窗口表	71
9.3.2	子窗口遍历函数	72
9.4	消息处理	72
9.4.1	对消息的响应	73
9.4.2	命令和子窗口消息	74
9.4.3	缺省消息处理	75
第十章	窗口对象	76
10.1	TWindow 类	76
10.2	初始化和生成窗口对象	76

10.2.1	初始化窗口对象	76
10.2.2	生成窗口元素	78
10.2.3	初始化和窗口生成概述	79
10.3	窗口类登记	79
10.3.1	登记属性	80
10.4	翻滚窗口	82
10.4.1	滚动条属性	82
10.4.2	为窗口设定滚动条	83
10.4.3	滚动实例	83
10.4.4	自动翻滚和跟踪	84
10.4.5	修改翻滚单位和区间	85
10.4.6	修改翻滚位置	85
10.4.7	设置页的尺度	86
10.4.8	为翻滚优化绘图成员函数	86
10.5	编辑窗口和文件窗口	87
10.5.1	编辑窗口	87
10.5.2	文件窗口	89
10.6	实例	90
第十一章	对话框对象	106
11.1	使用对话框资源	106
11.2	使用子对话框对象	106
11.2.1	构造和初始化子对话框对象	106
11.2.2	生成和执行对话框	107
11.2.3	关闭子对话框	107
11.2.4	把对话框用作为主窗口	108
11.2.5	控制管理和消息处理	108
11.2.6	使用对话框的复杂范例	111
11.3	输入对话框	112
11.4	文件对话框	113
11.5	实例	114
第十二章	控制对象	145
12.1	控制对象的使用	145
12.1.1	控制的构造与创建	145
12.1.2	控制的撤销	146
12.1.3	控制和消息处理	146
12.2	控制焦点与键盘	148
12.3	列表框控	148
12.3.1	构造和创建列表框	148
12.3.2	修改列表框	149
12.3.3	查询列表框	150
12.3.4	从列表框取得所选内容	150
12.3.5	实例	151
12.4	组合框	151

12.4.1	三种组合框	151
12.4.2	创建组合框	152
12.4.3	修改组合框	152
12.4.4	应用实例: CBoxTest	153
12.4.5	实例	153
12.5	静态控制	153
12.5.1	创建静态控制	153
12.5.2	查询静态控制	154
12.5.3	修改静态控制	154
12.5.4	静态控制的例子: StatTest 程序	154
12.6	编辑控制	154
12.6.1	创建编辑控制	155
12.6.2	剪接板和编辑操作	156
12.6.3	查询编辑控制	156
12.6.4	修改编辑控制	158
12.6.5	例子程序: EditTest	158
12.6.6	实例	159
12.7	按钮控制按钮	159
12.7.1	对按钮消息的响应	159
12.7.2	实例	159
12.8	复选框和单选按钮	160
12.8.1	创建复选框和单选按钮	160
12.8.2	查询选择框的状态	160
12.8.3	修改选择框的状态	161
12.8.4	响应复选框和单选按钮消息	161
12.9	成组框	161
12.9.1	创建成组框	162
12.9.2	响应成组框消息	162
12.9.3	程序范例: BtnTest	162
12.10	滚动条	162
12.10.1	创建滚动条对象	163
12.10.2	查询滚动条	163
12.10.3	修改滚动条	164
12.10.4	响应滚动条事件	164
12.10.5	范例程序: SBarTest	166
12.11	传递控制数据	166
12.11.1	定义传输缓冲器	166
12.11.2	创建控制和使能缓冲器	168
12.11.3	传输数据	168
12.11.4	为控制自定义传输	169
12.11.5	数据传输范例	169
12.12	实例	169
第十三章	建立自己的控制对象	194

13.1	修改预定义控制	194
13.1.1	修改生成方式	194
13.1.2	修改预定义消息响应	195
13.2	使用用户控制	196
13.2.1	设计一个用户控制	196
第十四章	MDI 对象	198
14.1	MDI 应用程序的组成	198
14.1.1	每个 MDI 窗口都是一个对象	198
14.2	构造 MDI 窗口	198
14.2.1	构造 MDI 主窗口	199
14.2.2	构造 MDI 子窗口	199
14.3	MDI 应用程序中的消息处理	200
14.4	管理 MDI 子窗口	200
14.4.1	子窗口的活动性	200
14.4.2	子窗口菜单	200
14.5	MDI 应用程序范例	201
14.6	多文档界面 (MDI) 实例	201
14.6.1	MDI 应用程序	201
14.6.2	MDI 消息循环	202
14.6.3	MDI 消息	203
14.6.4	主窗口 (Frame Window) 和子窗口函数	203
14.6.5	MDI 实例	203
第十五章	流对象	211
15.1	I/O 流 (iostream) 库	211
15.2	重载的 << 和 >> 操作符	212
15.3	流类的 TStreamable	213
15.4	流管理器	213
15.5	流类的构造函数	215
15.6	流类名称	216
15.7	使用流管理器	216
15.7.1	同流管理器码连接	216
15.7.2	创建流对象	217
15.7.3	使用流对象	217
15.8	流集合	217
15.8.1	使 Array 变为流类	218
15.8.2	流构造函数	218
15.8.3	StreamableName 成员函数	219
15.8.4	流的读函数	220
第十六章	构造 Windows 应用程序	221
16.1	在集成环境中编译和连接	222
16.1.1	理解资源文件	222
16.1.2	理解模块定义文件	222

16.1.3 编译和连接 WHELLO	222
16.2 WinMain	224
16.3 从命令行编译和连接	224
16.3.1 从命令行进行编译	224
16.3.2 从命令行进行连接	225
16.3.3 使用制作文件	226
16.4 入口和出口	227
16.4.1 Windows All Functions Exportable (-W)	227
16.4.2 Windows Explicit Functions Exported (-WE)	227
16.4.3 Windows Smart Callback (-WS)	228
16.4.4 Windows DLL ALL Functions Exportable (-WD)	228
16.4.5 Windows DLL Explicit Functions Exported (-WDE)	228
16.4.6 _export 关键字	228
16.4.7 入口、出口与输出: 小结	228
16.5 存储模式	229
16.6 模块定义文件	230
16.6.1 快速示例	230
16.7 Windows 程序的连接	231
16.7.1 在集成环境中连接	231
16.7.2 用 TLINK 来连接	231
16.8 动态链接库	233
16.8.1 在集成环境中编译和连接动态链接库	234
16.8.2 从命令行编译连接一个动态链接库	234
16.8.3 创建动态链接库	235
第十七章 RC: Windows 资源编译器	239
17.1 创建资源	239
17.2 添加资源到可执行文件中	239
17.2.1 从 IDE 中编译资源	240
17.2.2 从命令行编译资源	240
17.2.3 从制作文件中编译资源	240
17.3 资源编译器句法	240
第十八章 动态链接库 (DLL)	242
18.1 静态链接与动态链接	242
18.2 输入库	242
18.3 DLL 代码结构	243
18.4 创建一个 DLL	245
18.5 在 Windows 应用程序中使用 DLL	246
第十九章 用 ObjectWindows 进行 Windows 编程	249
19.1 使用 ObjectWindows 的 Windows 应用程序结构	249
19.1.1 WinMain 函数	249
19.1.2 消息循环	250
19.1.3 窗口过程	250

19.2	工程文件	251
19.2.1	模块定义文件	251
19.2.2	资源文件	252
19.2.3	C++ 源文件	252
19.2.4	库文件、DLLs 和输入库	252
19.3	为 ObjectWindows 应用程序使用 IDE	252
19.4	ObjectWindows 程序设计实例	253
19.4.1	基本的窗口例子	253
19.4.2	画线的例子	257
19.4.3	画弧的例子	258
19.4.4	填充图形例子	260
19.4.5	文本输出例子	261
19.4.6	制表文本输出例子	263
19.4.7	滚卷例子	265
19.4.8	加速键例子	266
19.4.9	位图例子	270
19.4.10	光标例子	272
19.4.11	对话例子	273
19.4.12	图标例子	277
19.4.13	菜单例子	280
19.4.14	MDI 例子	283

第三部分 程序实例

第二十章	Windows 程序实例	286
20.1	编译 Windows 程序	286
20.2	Windows 程序结构	290
20.3	鼠标应用范例 (一)	301
20.4	鼠标应用范例 (二)	307
20.5	剪贴程序	312
20.6	计时器的使用方式	318
20.7	监视内存	328
20.8	综合时钟应用程序	333
20.9	图标运用范例	343
20.10	光标运用范例	350
20.11	滚动条实例程序	357
第二十一章	Windows 下的 Turbo Debugger (TDW)	378
21.1	运行 TDW 的要求	378
21.2	安装 TDW	378
21.3	配置 TDW	379
21.3.1	使用 TDW 命令行选项	379
21.3.2	和 TDW 一起使用 TDINST	380
21.4	使用 TDW	380

21.4.1	记录窗口信息	381
21.4.2	获得内存和模块列表	385
21.5	调试动态链接库 (DLL)	387
21.5.1	使用 Load Modules or DLL 对话框	388
21.5.2	向 DLLs & Programs 列表添加一 DLL	389
21.5.3	在 DLL 中设置调试选项	389
21.5.4	控制 TDW 装入 DLL 符号表	390
21.5.5	调试 DLL 启动代码	390
21.6	把内存句柄变换为地址	391
21.6.1	调试要点	391
21.7	TDW 错误信息	392
第二十二章	调试一标准 Windows 应用程序	393
22.1	演示程序	393
22.2	编译及连接演示程序	393
22.3	调试 BCWDEMOA	394
22.3.1	决定做什么	394
22.3.2	终止 BCWDEMOA	394
22.3.3	记载信息	395
22.3.4	分析信息记录	395
22.3.5	发现错误	395
22.3.6	结束 BCWDEMOA	398
22.4	调试 BCWDEMOB	399
22.4.1	切换到程序外部	399
22.4.2	测试程序	399
22.4.3	决定做什么	400
22.4.4	比较全局存储列表	400
22.4.5	发现错误——一种功能方法	401
第二十三章	调试 ObjectWindows 应用程序	403
23.1	程序介绍	403
23.1.1	Colar Scribble 窗口类定义	404
23.1.2	创建应用程序	405
23.2	调试程序	405
23.2.1	发现第一个错误	405
23.2.2	发现画笔颜色错误	407
23.2.3	发现离开屏幕进行绘图的错误	410
23.2.4	发现删除屏幕错误	412
第二十四章	用于 Windows 的 Turbo Profiler	414
24.1	安装 TPROFW	414
24.1.1	安装 TDDEBUG.386	415
24.2	配置 TPROFW	415
24.2.1	使用 TPROFW 的命令行选项	415
24.2.2	使用 TFINST 配置 TPROFW	416

24.3 使用 TPROFW	416
24.3.1 剖析 Windows 过程	417
24.3.2 剖析动态链接库 (DLL)	418
24.4 TPROFW 出错信息	419

第一部分 ObjectWindows 初步

第一章 继承窗口

本章重点放在面向对象编程上。读者通过学习本章可以了解到 Windows 应用程序的构成及运行机制，同时还能看到如何利用 ObjectWindows 编写多任务协同运行的面向对象程序。

1.1 何谓 Windows 应用程序

Windows 应用程序是一种特殊的程序类型，它具有下述特点：

- (1) 必须是一种专门的可执行文件格式。
- (2) 仅在 Windows 下运行。
- (3) 一般在屏幕上的一个矩形窗口内运行。
- (4) 应用程序的显示和功能执行遵循标准的用户界面原则。
- (5) 可以同时与其它 Windows 应用程序或非 Windows 应用程序一起运行，当然还包括同一程序的其它实例。
- (6) 能够与其它 Windows 应用程序通信和共享数据。

1.1.1 Windows 的优点

对用户和开发人员来说，Windows 具有许多的优点。

对用户说来，这些优点包括如下几方面：

- (1) 标准且可预断的操作。

如果知道某一应用程序的使用方法，那么便能够使用所有的应用程序。因为 Windows 应用程序的界面是标准化的，它们具有一致性。

- (2) 不须为每一应用程序建立各种设备的驱动程序，因为 Windows 为其所支持的全部外设都提供了驱动程序。

- (3) 互相协作与通信。
- (4) 多任务：能够同时运行多个应用程序。
- (5) 能访问更多的内存。

对于 80286，80386 及 i486 机器，Windows 支持保护模式；在 80386 和 i486 机器上，Windows 还支持虚拟存储。

对开发人员说来，这些优点包括如下几方面：

- (1) 设备无关的图形保证图形应用程序可以运行于多种标准显示适配器。
- (2) 对于各种类型的打印机、显示器和定点设备(如鼠标和跟踪球)的直接支持。
- (3) 丰富的图形例程库。
- (4) 为大型程序提供更多的内存。
- (5) 支持菜单、图标、位图等等。

1.1.2 需求

Windows 提供上述诸多优点必须有一定的硬件条件作基础。一般情况下, Windows 应用程序要求较快的 CPU (相对于 DOS 应用程序而言), 较好的图形硬件 (如 VGA 卡及显示器) 和较多的内存。

对于常规使用, 有一台 80286 机器, 内带至少 2MB 内存, Windows 就能运行得很好了。

1.2 面向对象的窗口编程

面向窗口环境编程要求熟悉许多新的概念。即使开发一个简单的 Windows 程序都可能令人望而却步。ObjectWindows 简化了 Windows 编程的过程, 它使程序员能够把主要精力放在应用程序的功能上, 而不是放在它的格式上。

借助 ObjectWindows, 程序员可以使用对象来表示相当复杂的 Windows 程序单元。

ObjectWindows 的窗口对象封装了所有窗口需要的数据, 能执行公共的窗口操作, 并且能够响应 Windows 的公共消息和事件。ObjectWindows 的窗口类和应用程序类完全管理消息的处理。

1.2.1 一个较好的 Windows 接口

ObjectWindows 使用了 Turbo C++ 面向对象特征, 把 Windows API 的各个部分都封装了起来, 从而使程序员同 Windows 编程的细节隔离开来。其结果是, 只须花少量的时间和精力便能编写 Windows 程序。特别是, ObjectWindows 提供了下述三个有益的特点:

- (1) 窗口信息的封装。
- (2) 诸多 Windows API 函数的抽象化。
- (3) 自动消息响应。

1. 封装窗口信息

ObjectWindows 提供了定义 Windows 窗口、对话框及窗口控制的操作与数据存储的对象。在一个 ObjectWindows 应用程序中, 一个界面对象便充当一个可见的 Windows 界面元素的代表。尽管它们在运行上是紧密配合的, 但是有必要弄清两者之间的区别。

界面对象是与新的界面元素相关联的。直接调用界面对象的 Create 成员函数便能生成新的界面元素。Windows 调用时传递的界面对象数据成员定义了所要生成的窗口单元的物理属性。Windows 返回所生成界面元素的句柄和标积。界面对象用其数据成员 HWindow 来存放这个句柄。

由于对窗口操作的 Windows 函数需要获取被操作窗口的句柄, 故把句柄作为一个数据成员能使其访问更为容易。类似地, 数据成员可以用来存放某个特定窗口的画图工具或状态信息。

2. 抽象化的 Windows 函数