ObjectWindows

程序设计

Borland C++ 3.1 for Windows

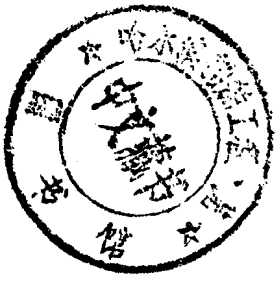
乔斯　韦诚　编著

北京大学出版社

ObjectWindows 程序设计

Borland C++ 3.1 for Windows

乔 斯 韦 诚 编著



北京大学出版社

新登字(京)159号

图书在版编目(CIP)数据

ObjectWindows 程序设计:Borland C++ 3.1 for Windows/乔斯,韦诚编著.—北京:北京大学出版社,1994,6
ISBN 7-301-02509-2

Ⅰ O…
Ⅱ ① 乔… ② 韦…
Ⅲ C 语言-程序设计
Ⅳ TP312

出版者地址: 北京大学校内
邮 政 编 码: 100871
排 印 者: 蓝地公司激光照排
北京飞达印刷厂印刷
发 行 者: 北京大学出版社
经 销 者: 新华书店
787×1092 毫米 16 开本 11.75 印张 286 千字
1994 年 6 月第 1 版 1994 年 6 月第 1 次印刷
印 数: 0001—3500 册
定 价: 22.00 元

前　言

微机的操作系统正处于变化之中，自从 Microsoft 推出 Windows 3.0 版后，很多微机用户都从 DOS 操作平台转移到 Windows 平台上来，Windows 正逐步成为微机操作系统的主流。与此同时，程序设计语言也经历了一场革命，在传统的结构化程序设计理论的基础上发展起来的面向对象程序设计(OOP)理论正日益深入到各种语言和软件中。Borland公司在C++ 上实现的 ObjectWindows 类库(OWL)正是利用 OOP 的强大功能为 Windows 程序员设计的，它能极大地减轻程序员的工作，并使程序便于理解和维护。考虑到近年来 Windows 程序员的需要会大大增加，作者写下此书以供需要学习 Windows 编程的读者使用。

本书的重点在于介绍 Borland 公司的 ObjectWindows 类库，在此之前简要介绍了 Windows 操作系统中的一些基本概念。本书包括十章：

第一章介绍 Windows 操作系统，主要引入一些概念。

第二章介绍利用 C 语言编写 Windows 应用程序，主要是为了引入 Windows 程序设计的主要思路。

第三章介绍 OOP 的概念，并简介了 ObjectWindows 类库。

第四章介绍 Windows 资源设计的工具 Resource Workshop。

从第五章开始介绍 ObjectWindows 类库中的具体内容。

第五章介绍 OWL 程序的主体——应用程序类。

第六章介绍 OWL 中各种界面的基类——TWindowsObject。

第七章介绍窗口类——TWindow。

第八章介绍几种特殊的窗口。

第九章介绍对话的使用。

第十章介绍各种控制。

类库是程序简化的根本原因，读者应着重于正确理解上述类库中各种类所包含的内容及其使用方法。这样就能很快地掌握 OWL 程序设计的方法。

作　者

1993 年 12 月

目　录

第一章　Windows 操作环境概述

Windows 操作环境是指 Microsoft 公司的微机操作系统 Microsoft Windows 3.0 或其更高版本。自从 Windows 出版以来，深受广大用户的青睐。Windows 丰富的界面使用户的操作变得特别简单。连同 Windows 一起销售的部分应用软件更是用户日常工作的帮手。读者如是 Windows 的老用户，相信对这一点会深有体会。读者也许曾经尝试过编制 Windows 应用程序，由于 Windows 的复杂性，使得 Windows 编程比较麻烦。Windows 的编程思想不同于一般的直线式编程思想，要使程序员和 Windows 操作系统协同工作，才能得到所希望的结果。此外，Windows 的六百多个应用程序接口（Application Programming Interface，API）函数也使很多读者望而生畏。Borland 公司推出的在 Borland C++中使用的 ObjectWindows 可以使程序员的工作减轻很多。本书的重点是讲述怎样利用 ObjectWindows 进行编程，在此之前，为使后面讲述方便，首先概要介绍一下 Windows 操作环境，请读者注意其中的概念，有很多是要在稍后的讲述中用到的。

1.1　用户界面

对于操作系统，用户最先接触到的就是用户界面。传统的 DOS 操作系统使用的是文本界面，在屏幕上只能显示固定大小的字符。而 Windows 使用的是图形界面，也就是说它使用画图的方式进行输出。这样设计的好处是不言而喻的。举个例子，字处理软件中所追求的“所见即所得”（What You See Is What You Get）只有在图形界面下才能实现。此外，Windows 是一个多任务操作系统，因此，Windows 应该保证在其中运行的不止一个应用程序的输出，即大多数应用程序应该能够同时在屏幕上显示自己的运行结果。Windows 为每个应用程序分配一部分屏幕，这样，在任何时刻，应用程序都可以在屏幕上显示自己。Windows 的输出函数使用一种称为设备环境（DC，Device Context）的句柄（Handle）来获得输出所需要的参数，这个句柄资源是有限的，所以，在使用 DC 后迅速释放这个资源就是用户应用程序的责任之一，否则其它应用程序就可能因此而不能输出。应用程序在进行输出前要建立一个窗口，然后向这个窗口输出应用程序结果。一旦建立了这样的窗口后，Windows 将提供大量与此相关的信息，同时将自动完成许多用户要求的工作，如移动窗口，改变窗口大小等。

图 1.1 显示了一个典型的 Windows 界面。其中包括了 Windows 中的大多数元素，如菜单、对话框，控制等。请读者熟记这些名词，因为在后面需要用到这些概念。

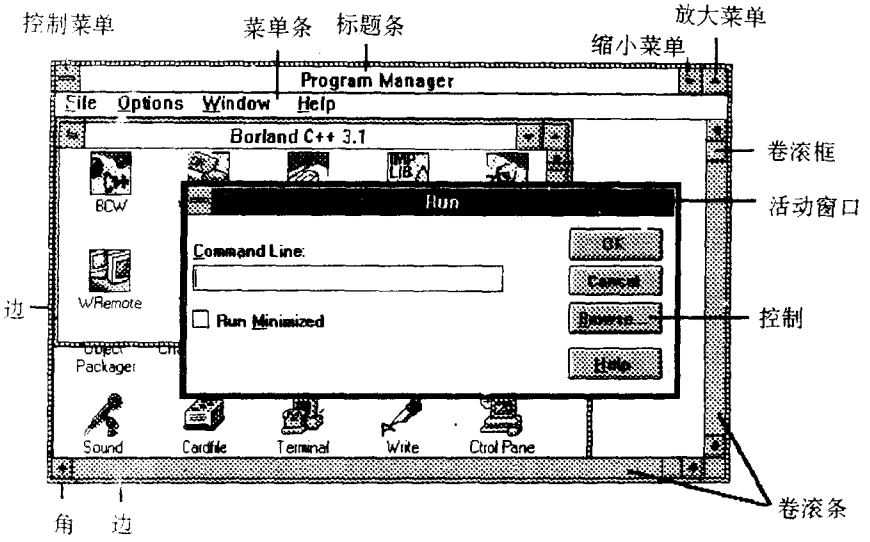


图 1.1 典型的 Windows 应用程序界面

1.2 队列输入

Windows 操作系统的另一个特点是应用程序不需要自己查询是否有用户输入。读者也许有过 DOS 编程经验，在 DOS 环境中，程序员需要使用类似于 C 语言中的 getchar()这样的函数来查询或获得用户输入。而在 Windows 环境下，Windows 负责接收所有的输入，包括键盘输入、鼠标器输入和时钟输入等。同时，Windows 也负责管理这些输入，把他们分送到相应的应用程序的"消息队列"中。这个过程被称为发送 Windows 消息(Message)给应用程序，应用程序根据 Windows 消息进行反应。

请读者注意 Windows 环境中的这种输入方式。正是由于这种输入方式，使 Windows 应用程序成为"消息驱动"的应用程序，这也是 Windows 程序设计区别于传统的程序设计方法的一个方面。在传统的程序设计思想中，当需要输入信息时，应用程序查询并等待用户输入，在得到输入后，经过判断，决定程序的运行方向。而在 Windows 应用程序中，Windows 接收输入并把它送给适当的程序段处理。这样，应用程序的设计成为设置很多程序段(函数)的工作，用这些函数响应用户输入或其它输入。同时，如果应用程序不想对输入作出反应，也可以使用 Windows 提供的缺省操作。

图 1.2 给出了 Windows 和应用程序对用户输入作出反应的例子。当用户按下某一键后，Windows 接收到一键盘输入消息，并将此消息从系统队列拷贝到应用程序队列。应用程序得到此消息后把它翻译成 ANSI 字符消息 WM_CHAR，然后经由 Windows 把此消息连同原键盘消息一起发送给相应的窗口函数，窗口函数利用 Windows 将其输出。这里的部分概念将在以后详细叙述。

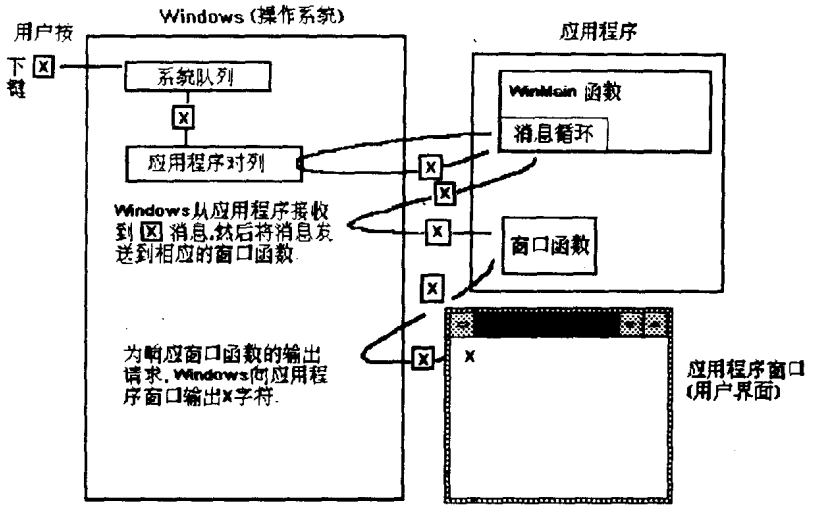


图 1.2　对输入的响应过程

1.3　与设备无关的图形输出

前面讲过,DOS 是文本界面,虽然也可以让 DOS 在图形界面下工作,但通常这项工作是由应用程序来负责的。由于计算机硬件的差别很大,因此应用程序要负责驱动所有它想要输出的图形设备,但仅打印机的型号就有上百种,如果程序员要为每种打印机编制一套驱动程序,将是一种费时而又低效的重复劳动。

Windows 使用"设备驱动程序"与相应的硬件打交道。程序员只要使用 Windows 所提供的丰富的图形操作函数,就可以很方便地产生较复杂的图形。由于把设备驱动的任务交给了操作系统,用户就可以使用同一应用程序向不同的设备输出图形。这里所说的图形是广义的图形,包括文字。

1.4　多任务操作系统

Windows 是一个多任务操作系统,在同一时间段内可以执行几个应用程序,并可以运行一份程序的几个不同拷贝。读者对多任务这个概念应该有所了解,多任务操作系统中的各个应用程序并不是同时运行的,而是等待一个应用程序放弃对 CPU 的控制,其它应用程序才有机会运行。因此程序员在编程时应该注意,不要控制 CPU 太久。关于怎样设置 Windows,使它更有效地工作,有兴趣的读者可以翻阅相应的操作手册。

同样,在 Windows 下运行多个任务时,由于内存也是共享资源,因此 Windows 负责管理内存,协调各应用程序。为了更好利用内存,不造成浪费,Windows 经常要移动内存数据,有时甚至要抛弃一些内存块中存储的数据,把这部分内存分配给其它应用程序使用。因此用户不应认为他曾经赋值的某个内存块会永远保持,要注意保持数据的正确性。另外要提到的是,Windows 运行同一程序的多个拷贝时,仅拷贝应用程序的数据段。这样做可以节省内存。因此用户

编制 Windows 程序时应注意不要在代码段内放置数据。

1.5 动态链接库

所谓“动态链接”就是只有当应用程序需要时，才把适当的函数调入内存链接运行，而在可执行文件中仅含有一张记载有关这个函数的资料和它在动态链接库中的代码或名字。动态链接库即存有这些 Windows 函数执行码的文件。

Windows 本身包含三个动态链接库：负责管理内存、加载可执行文件和进行程序安排的 KERNEL；负责用户界面和有关 Windows 的各项事宜的 USER；以及负责图形输出的 GDI (Graphics Device Interface)。这三个库完成大多数 Windows 中的工作。

由于动态链接库的原因，当函数被加载到内存后，未必与程序代码段处于同一段内，因此 Windows 中的函数全部采用远程调用(Far Call)。不过读者完全可以放心，C++的函数原型(在 WINDOWS.H 中定义了 Windows 中的函数原型)功能使 C++编译器能够正确使用程序代码。

1.6 软件开发工具

Windows 软件开发工具包括以下三部分：

① 资源编辑器，负责对资源进行编辑，编译，以及和可执行程序链接。关于“资源”的概念将在第四章中详细讲解。

② 语言编译器及编辑器。程序员以 C 语言或其它语言源程序格式书写 Windows 应用程序，而后经过编译，链接成可执行文件。最终与资源相连成可在 Windows 中执行的应用程序。

③ 帮助系统编译、编辑器。可以用于程序员产生应用程序的 Windows 帮助系统。

Microsoft 公司在推出 Windows 后，提供了一个软件开发工具 SDK (Software Development Kit)其中包括：

Microsoft C 优化编译器：CL.EXE；

Microsoft Segmented-Executable Linker：LINK.EXE；

Microsoft Windows 资源编译器：RC.EXE；

Microsoft Windows SDKPaint：SDKPAINT.EXE；

Microsoft Windows 对话编辑器：DIALOG.EXE。

此外，如果要利用 Code View 查错等工作，还需要另外一些软件。用户可以利用上述软件进行 Windows 应用程序开发。

作者在使用过程中感觉到，使用 Borland C++ & Application Frameworks 3.1 很方便。它可在 Windows 环境下的集成编译环境(IDE)省去用户在 DOS 和 Windows 间来回切换的麻烦。ObjectWindows 类库使用了 C++的最大优点，使用户编制 Windows 应用程序成为一件轻松自如的事。Resource Workshop 提供给用户进行资源编辑、编译过程极大的方便。另外，Turbo Debugger for Windows，Winsight，ImportLib，WRemote，WRSetup，Turbo Profiler for Windows，FConvert 等实用软件也都可以给用户的工作节省很多时间。

第二章　Windows 程序设计基础

2.1　Windows 程序设计中用到的概念

在讲述 Windows 程序设计之前，我们必须先弄懂一些基本概念，这样在此之后的叙述才有基础。

首先，什么是 Windows 应用程序？顾名思义，Windows 应用程序是指为在 Windows 环境下运行而编写的应用程序。Windows 应用程序使用 Windows API 来执行任务。为什么要起名为 Windows 应用程序呢？这是为了和 DOS 应用程序有所区别。因为 DOS 是传统的微机操作系统，因此最初的微机应用程序都是指 DOS 应用程序。在 Windows 得以发展之后，才出现了大量的 Windows 应用程序。

下面我们来讲讲 Windows 应用程序设计中要用到的其它概念。

窗口函数：在第一章内我们讲到，应用程序和 Windows 操作系统要协同工作，才能得到正确的结果，这种工作主要表现在窗口函数上。窗口函数是在应用程序中定义的、由 Windows 调用的函数。窗口函数完成与该窗口相联系的操作并返回信息。请读者注意，窗口函数并不是由应用程序直接调用的，而是由 Windows 调用的。在多数情况下，Windows 接收某种信息，并通知应用程序，由应用程序对其进行翻译并通知 Windows 调用相应的窗口函数。但有时，如用户选择关闭一个窗口（通过两次点按窗口的系统菜单），Windows 将直接发送 WM_DESTROY 消息给相应的窗口函数。如果应用程序的主窗口被关闭，窗口函数将把消息 WM_QUIT 拷贝到应用程序队列，主函数接收到该消息后将退出应用程序。窗口函数有时也称为“回调”(Callback)函数，将在应用程序的模块定义文件的 EXPORTS 段定义。除窗口函数外，回调函数还包括所有的对话函数以及一些特殊回调函数，例如为某些 Windows API 函数准备的枚举函数。统言之，任何在应用程序外被调用的函数都称为回调函数。

窗口：窗口是应用程序唯一能够使用的屏幕输出手段，同时也是应用程序的主要输入手段。窗口包含有标题栏，滚动杆，系统菜单，边框，在显示器上占一块矩形区域等特征。理论上讲，应用程序创建窗口并拥有其独占权；但实际上，窗口管理由应用程序和 Windows 共同负责。Windows 维护窗口的位置和外观，管理边框、滚动杆、标题这些标准窗口特征，并执行很多由用户发出的、能够直接影响窗口的任务。除上述标准操作以外的工作由应用程序负责，窗口中用户区的显示也完全由应用程序控制。图 2.1 显示的是一个标准窗口。

由于协调工作的需要，Windows 向每个可能要变化的窗口发送消息(Message)，每个窗口则以它的窗口函数接收此消息，并根据其内容作出反应。

消息(Message)：消息是指由 Windows 发送或应用程序自己产生的一组内容。用户按动

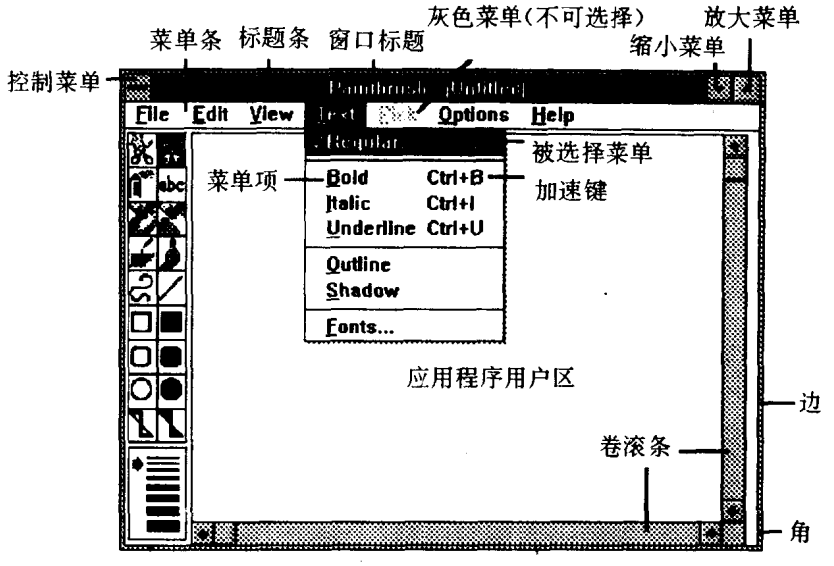


图 2.1 标准窗口

键盘,移动鼠标器,选择一个菜单等操作都会引起 Windows 向应用程序发送消息。定时器和端口等也会由 Windows 向应用程序发送消息。消息的格式有两种:

① 由 Windows 置于应用程序队列的消息,这种消息为 MSG 结构。

② 由 Windows 直接传递给窗口函数的消息。

MSG 消息结构如下:

① 当前光标位置;

② 当前系统时间;

③ 有关窗口句柄(hWnd);

④ 消息标识符(message);

⑤ WORD 类型参数(wParam);

⑥ LONG 类型参数(lParam)。

直接传送给窗口的消息不包括前两个参数。对于应用程序而言,最后两个参数比较有用。关于具体消息中的具体内容,读者可以在程序设计时使用联机 Help 查看,或者查看有关资料。

菜单:读者在图 2.1 中可以看到一个菜单。事实上,菜单是一个命令表,用户可以察看和选择其中的选项(命令)。菜单是由 Windows 来显示和管理的,而对菜单命令的响应则需要由应用程序来负责。在应用程序的运行中,应用程序可以根据情况来增加或删除一些菜单,或使某些菜单变灰(不能选择该菜单)。

对话框:对话框是一个临时窗口,它为用户显示有关命令的更多信息,并获得用户输入。图 2.2 显示了一个文件对话框,以便让用户选择或输入一个文件名,使应用程序可以利用该文件名完成相应的工作,如读取该文件的内容等。

在对话框中可以有很多控制。每个控制都是一个小窗口,可以实现简单的输入、输出功能。

在图 2.2 中有编辑控制，列表控制，按钮控制等。关于具体的控制将在后续章节中详细讲述。

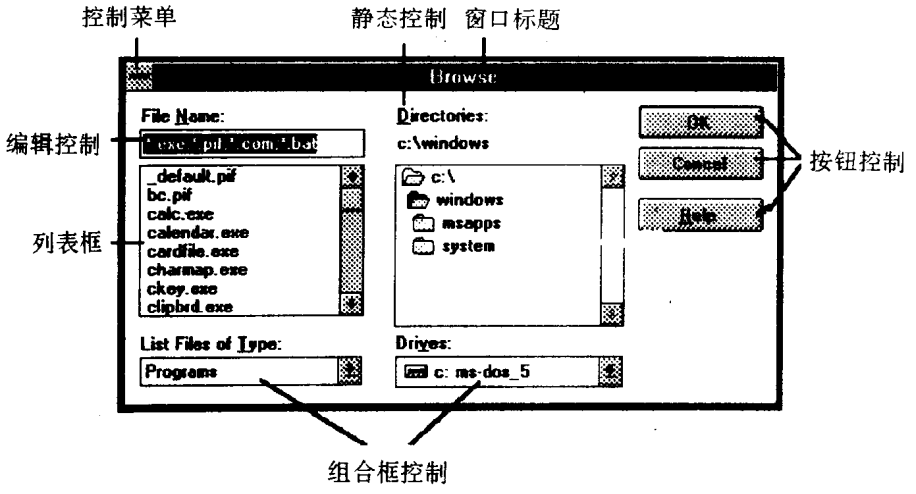


图 2.2　一个文件对话框

句柄(Handle)：句柄是一个特殊的整型数。它被 Windows 使用来标识诸如应用程序事例、窗口、菜单、控制、分配的内存、输出设备、GDI 笔和刷等对象。大多数句柄为一个对内部表格的索引值。应用程序通常只接触句柄，而不直接存取表中数据，这样做可以保护表中数据不受 Windows 因多任务的关系而作出调整的影响。

Windows 中的数据结构：读者可能对 C 语言很熟悉，在 C 语言中只有几种数据类型：整数型，浮点数型，字符型、指针型数据。但读者也许已经遇到过，或许在将来会遇到很多并没有在程序语言中定义的基础数据，如 WORD，LONG，HANDLE，HWND，LPSTR，FARPROC 等。这些都是从基础数据用 typedef 定义出来的，事实上它们仍没有超出这些基础数据类型。它们的定义都在 WINDOWS.H 中。另外，在这个文件中还定义了另外一些结构，如 MSG、WNDCLASS、PAINTSTRUCT、RECT 等。关于这些数据或结构的具体含义，将在后续章节解释。

在 Windows 应用程序设计过程中，还会遇到很多新的概念，如图标，加速键等，这些将在程序设计中讲述。

2.2　Windows 软件开发周期

所谓软件开发周期就是进行一个应用程序的开发过程。主要包括以下几个步骤：

① 编写源程序。在适当的软件支持下，可以利用任何语言编写程序。由于本书是讲利用 C/C++语言进行 Windows 程序设计，因此只讲述利用 C 语言编程。在这一步中，程序员将编写包括窗口函数，WinMain 函数在内的 C 语言源程序，C 语言头文件(Header Files)，汇编语言源程序。

读者也许注意到了前一段中提到的 WinMain 函数，这是 Windows 应用程序的入口。启动应用程序后，将首先以 WinMain 开始执行，这和 C 语言 DOS 应用程序不同(以 main 函数作为

程序主入口)。

② 建立程序资源。程序资源指程序中要用到的菜单、光标、位图、对话框等。关于资源的更详细的解释请看第四章。

有经验的程序员可以直接使用文本编辑器编写资源文件,初学者可以使用 Borland C++ 提供的 Resource Workshop 来直观地编辑各种资源,并把它加到应用程序资源描述文件(.RC)中。关于 Resource Workshop 强大功能的具体情况,请读者看第四章。

③ 建立应用程序模块定义文件(.DEF)。这是一个定义应用程序中如段属性、栈的尺寸、堆的大小之类属性的文件,这个文件在 Microsoft C 中是必需的。但如果读者使用 ObjectWindows 将不需要此文件。

④ 编译并链接源程序成为可执行文件。

⑤ 编译资源文件,并将结果和第 4 步完成的可执行文件相链接,成为最终的可执行文件。

读者可能认为,到此可以休息一下了。但事实上,上述工作完成后,应用软件开发的工作才完成了一半。要开发出真正完美的软件,还应该进行下一步工作。

⑥ 检测应用程序,查找程序错误,并改正之。程序员也许要在这一步上耗费大量时间。

图 2.3 是一个典型的开发过程示意图,它包括了前 5 个步骤。

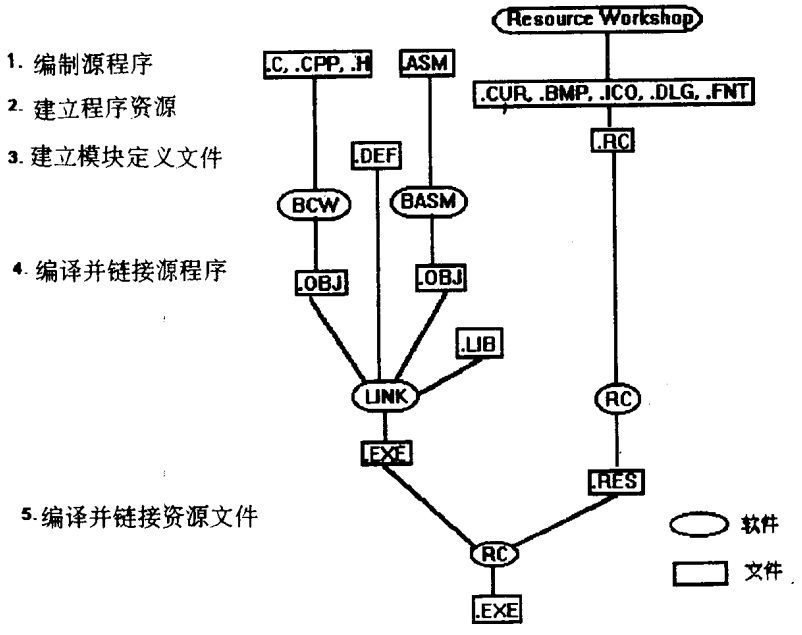


图 2.3 开发 Windows 软件过程示意图

读者也许习惯于使用命令行编译程序,但作者强烈建议使用 Borland 产品中的集成编译环境(IDE)。在 IDE 中你既可进行源程序编辑,又可以对之编译、链接,还可以编译资源描述文件,并把它与可执行程序相链接。Borland C++ for Windows 事实上就是一个集成编译环境。如果在 Windows 中还使用命令行的话,可以相信,不到十分钟,用户就会不再想用了。事实上,使用 Borland C++ for Windows 进行 Windows 软件设计是最方便的。

2.3 使用 Borland C++ for Windows

虽然在这一章中，我们要讲述的是使用 C 语言编写一个简单的 Windows 应用程序，但我们仍要使用 Borland C++环境，因为 C 毕竟是 C++的一个子集。另外，读者如果从现在就接触 C++环境，在将来使用 C++语言时就不会因为需要重新熟悉软件环境而浪费时间。

2.3.1 Borland C++软件包

Borland C++ 3.1 包括一组软件。图 2.4 是 Borland C++的软件包。包括十个实用软件和八个帮助信息，下面简略介绍各软件的用途。

(1) BCW

BCW 即 Borland C++ for Windows。这是一个集编辑、编译、链接、运行、调试于一体的集成环境，在其中可以编辑源程序；编译 C++语言和汇编语言编写的源程序；链接成可执行程序；调用 RC 编译并链接资源文件；调用 TDW 进行调试和查错。

(2) Workshop

Workshop 即 Resource Workshop，这是 Borland 公司的资源编辑/编译软件。在其中可以以直观的方式和以文本的方式编辑应用程序资源，供应用程序使用。如不想利用源程序来编译/链接应用程序或在得不到源程序的情况下，也可以直接读取应用程序的可执行文件，或经过编译后得到的 RES 文件，对其中的资源进行修改并存入可执行文件中。

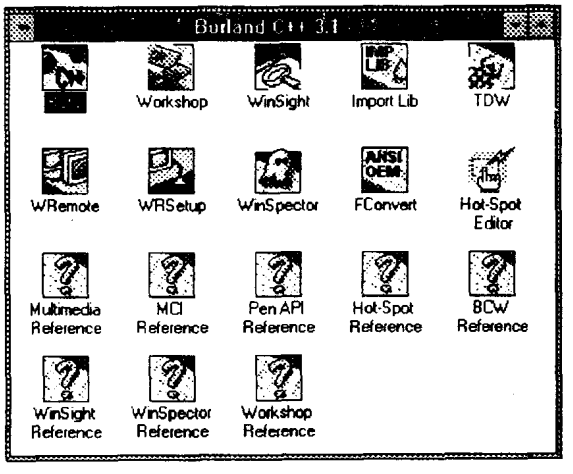


图 2.4 Borland C++软件包

(3) WinSight

WinSight 是一个调试工具，用户可以利用它来查询关于窗口，窗口类以及消息方面的信息，用户可以利用它研究自己或别人的 Windows 应用程序，也可以利用它看到窗口和窗口类是怎样产生的，怎样使用的，以及窗口接收到了什么消息。读者在编制 Windows 应用程序时，

对 WinSight 的用途将会有所了解。如果读者的应用程序没有像读者所想象的那样运行，则很可能是没有接收到正确的消息，用 WinSight 可以很方便地查出窗口所有接收到的消息，从而编制正确的消息响应函数。

(4) ImportLib

ImportLib 是为配合动态链接库(DLL)使用的，在第一章中讲过，Windows 使用 DLL 来提高系统利用率，例如，使用 DLL 可以节省内存、共享应用程序代码、建立设备驱动程序等。

在 Windows 应用程序中，我们把在某一应用程序模块中定义，而被其它模块调用的函数称为"输出"函数。在这里，模块是指 Windows 中的一个基本结构单元。有两种类型的模块：应用模块和 DLL 模块。每个 Windows 应用程序的.EXE 文件被认为是一个应用模块；DLL 模块为每一个具有.DLL、.DRV、.FON 扩展名的 Windows 系统文件。

在一个应用模块中调用而在另一个模块中定义的函数称为"输入"函数。对于一个用户应用程序而言，可能会使用很多"输入"函数，如各种 Windows API 函数或用户自己定义的被几个程序共用的函数。由于 Windows 使用动态链接方式，所以应用模块中不具有输入函数代码，仅包含与此函数相关的信息。C++语言使用三种方式获得这些信息：①链接 DLL 的输入库。输入库中包含有与输入函数相关的信息。②使用模块定义文件(.DEF 文件)引入相关信息，供链接程序连入。③在应用程序运行时，使用 LoadLibrary 函数和 CreateInfo 函数来动态链接输入函数。

使用 ImportLib 可以为每个 DLL 模块定义一个输入库，供应用程序链接时使用。

(5) TDW

TDW (Turbo Debugger for Windows) 是一个源语言级的调试工具，它能够与 Borland C++共同工作。当然，TDW 也可以进行汇编级调试。利用 TDW，用户可以对程序进行跟踪(Trace)、反向跟踪(Back Trace)、步进(Step)、观测(View)、检查(Inspect)、更改(Change)和监视(Watch)等操作，以便迅速而准确地查出程序的出错地点和出错原因，从而方便程序员的开发工作。当然，TDW 不能重新编译程序，这个工作要由 BCW 来完成；同时，TDW 只能在字符界面下工作，当然，用户屏幕将仍是 Windows 的图形界面。使用 TDW 应注意的一点是：TDW 不能代替程序员思考，真正发现错误并解决问题的将是程序员本人。

(6) WRemote

WRemote 是一个远程调试软件，它使用户可以使用两台机器对应用程序进行调试工作。这样做有以下两个理由：

① 使用两台机器，其中之一运行 Windows 和 WRemote 及应用程序(称为远程系统)，而在另一台机器上使用运行 TDW(称为本地系统)可以省去在文本界面(TDW 使用)和 Windows 的图形界面之间的来回切换，这种切换引起的屏幕的闪烁现象使人觉得厌烦。

② WRemote 使用的内存比 Turbo Debugger 使用的内存少得多，因此 Turbo Debugger 不必在后台运行。

使用 WRemote 必须使用两个系统，两个系统之间用串行口或通过局域网(LAN)相连。

(7) WRSetup

WRemote 的配置软件。在第一次运行 WRemote 之前，应该先运行 WRSetup 程序以建立通讯设置。