目　录

第一章 概述

Windows 操作系统已经广泛地被人们所接受，人们越来越多地体会到图形界面(GUI)带来的巨大好处。随着 Windows 98 的正式推出，微软公司随即又推出了基于 Windows 98 操作平台的强大编程工具软件： Visual C++ 6.0。

Visual C++ 6.0 与 Windows 操作系统有着紧密的联系，因此在学习 Visual C++ 之前，掌握一定的 Windows 编程基础是很有帮助的。本章将系统地总结 Windows 的编程模式，再对 Visual C++ 6.0 进行简单的介绍，让读者对 Visual C++的编程平台和工作方式有一个初步的认识。然后列出 Microsoft 基本类库（MFC）中的常用类的类名及其功能，以便读者对 MFC 有一个全面的了解，并为编程时的查询提供方便。

1.1 Windows 编程模式

Windows 操作系统是基于 PC 机的图形操作系统。Windows 操作系统主要有 Windows 95、Windows 98、Windows NT...。由于微机功能的不断增强，应用越来越广泛，越来越多的系统由微机构成，微机上占主导地位的操作系统就是 Windows 操作系统。因此，编写 Windows 程序是跟上世界软件时代潮流的关键。

不管人们使用什么样的开发工具，Windows 程序设计同以往已经过时的批命令或面向事务的程序设计有了根本的不同。首先， 读者必须了解一些有关 Windows 的基础知识，下面将参照大家熟知的 MS-DOS 编程模式来进行介绍。

1.1.1 消息处理

当用户编写 MS-DOS 应用程序时，最起码要用到 main 函数。当用户运行该应用程序时，操作系统会自动调用 main，然后就可以使用任何所需要的程序结构。如果程序需要得到用户键盘输入，或者需要使用操作系统所提供的功能时，main 函数就可以调用适当的函数，如 getchar 等等，或者也可以使用基于字符的窗口库。

当 Windows 操作系统运行程序时，它首先调用程序中的 WinMain 函数。因此，在 Windows 应用程序中一定要有 WinMain 函数，该函数一般用来完成某些特殊的任务，其中最重要的任务之一就是要创建应用的“主窗口”。“主窗口”中必须包含用来处理 Windows 所发送的消息的代码。基于 Windows 的程序和基于 MS-DOS 的程序之间最基本的差别就在于 MS-DOS 通过操作系统来获得用户输入，而 Windows 程序则是通过操作系统发送的消息来处理用户输入。

Microsoft Visual C++和许多的 Windows 开发环境一样，已经通过隐藏 WinMain 函数及构造消息——控制机制来使编程得到简化。虽然使用 MFC 库编程不再需要 WinMain 函数，但是弄清楚操作系统和程序之间的这种联系是非常有用的。

许多 Windows 消息都经过了严格的定义，并且会发送给所有的应用。例如，当窗口被创建时就会自动发送 WM_CREATE 消息，当用户按下鼠标的左键时就会自动发送 WM_LBUTONDOWN 消息，当用户敲了一个字符键时就会自动发送 WM_CHAR 消息，而当用户关闭窗口时系统又会自动发送 WM_CLOSE 消息。当用户进行菜单选择时，系统会相应地发送一些其它消息，而这些消息则完全依赖于应用的菜单设计。当然，程序员还可以定义一些其它消息，这些消息被称作“用户消息”。关于消息处理机制的用法和控制，将在本书第八章中具体介绍。

考虑消息在 Windows 中所起的作用时，请注意以下几点：消息系统允许 Windows 实现其多任务能力；消息系统使 Windows 能够在各应用程序间分配处理器；Windows 每次向应用程序发送消息的同时，也向应用程序提供使用处理器时间的许可。事实上，应用程序访问微处理器的唯一办法是接收到一条消息。其次，消息使应用程序能够响应环境中的事件。这些事件可由应用程序本身、其它同时运行的程序、用户或 Windows 产生。每次事件发生，Windows 便作一记录，并把相应的消息送至有关应用程序。

读者完全不用担心如何用代码来管理发送这些消息，因为这些都是应用框架的事情，不过需要注意的是，Windows 的这种消息处理机制同时也强加给了用户程序许多固定的结构。因此，在此已经无法在 Windows 程序中找到以前老式的 MS-DOS 程序的影子。只要仔细研究本书中的例子，读者就不难发现这些程序的确与以往编写的程序有所不同。

1.1.2 多任务环境

Windows 多任务环境允许用户在同一时刻运行多个应用程序或运行同一应用程序的多个例程。每个应用程序占据了屏幕上的一个矩形窗口。在任意给定时刻，用户可以在屏幕上移动窗口，在不同应用程序间切换，改变窗口大小，在窗口之间交换信息。

但是，在任意时刻，只有一个应用程序可以使用处理器。区别正在处理的任务与仅是正在运行的任务的是很重要的。活动的应用程序正在接收用户的注意信号。由于在任意给定时刻只能有一个应用程序正在处理，所以，在某一时刻也只能有一个活动应用程序。但是，可以有任意数量的任务同时在运行。分配微处理器的处理时间(也叫时间片)，是 Windows 的责任。Windows 通过使用队列输入或消息来控制微处理器的共享。

在 Windows 实现多任务操作之前，应用程序假定自身控制着所有的计算机资源，包括输入和输出设备、内存、视频显示，甚至控制 CPU 本身。但在 Windows 下，所有这些资源必须被共享。例如，内存管理是由 Windows 控制的，而不是由应用程序控制。

1.1.3 硬件无关性

Windows 也提供了独立于硬件或设备的特性。Windows 使程序员在开发程序时不必考虑显示器、打印机、计算机输入设备等各种条件。许多 MS-DOS 程序都直接往视频存储区或打印机口输送数据，这种做法的不利之处在于需要对每种显示卡或打印机类型提供相应的驱动程序。而 Windows 则提供了一个抽象的接口，称作图形设备接口(GDI)。这样人们就可以不必关心与系统相连的显示卡及打印机类型。人们可以通过调用 GDI 函数自动参考被称为设备环境的数据结构。Windows 会自动将设备环境结构映射到相应的物理设备，并且会提供正确的输入/输出指令。GDI 在处理速度上几乎和直接进行视频访问一样快，并且允许 Windows 的不同应用共同享用显示器。

1.1.4 基于资源的程序设计

在 MS-DOS 下，为了实现数据驱动的程序设计，必须将数据定义为初始化常量，或者提供单独的数据文件供程序读取。而在进行 Windows 程序设计时，可以用一些特定的格式将这些数据存储在资源文件中，Windows 通过“联编”过程将资源文件嵌入到连接程序中。资源文件可以包含位图、表象、菜单定义、对话框设计和字符串，甚至可以包含用户自定义格式。

用户通常通过文本编辑器来编辑程序，而对各种资源则通过“所见即所得”风格的工具来加以编辑。例如，用户在设计对话框时，首先从控制模板(也可以叫作工具箱)标象组中选取适当的元素，然后再对按钮、列表框等通过鼠标来进行定位和设置尺寸。利用 Visual C++ 的 App Studio 资源编辑器，用户可以对大部分资源进行有效的编辑。

1.1.5 动态链接库(DLLs)

在 MS-DOS 环境下， 所用程序的目标组件在创建过程中都被静态的连接起来，而 Windows 则允许动态的连接，及一些特定结构的库可以在运行过程中被装入和连接了，并且多个应用可以共享同一个动态链接库，这可大大节省内存和磁盘空间，同时动态连接还可以大大增强对程序调试的灵活性，因为用户可将动态链接库单独编译和调试。

连接器并不把库函数拷贝到可执行文件中，而是当程序运行时，调用库中的函数。自然，这可以节省内存。无论有多少应用程序正在运行，在某一给定时刻，只有一份库在内存中，这个库可以共享。

经过不懈的努力，Microsoft 基本类库的开发者终于将应用框架的所有类组合进了动态链接库，并且人们还可以创建自己的基于 Microsoft 基本类库的 DLL。

1.2 Visual C++ 6.0 简介

Visual C++包含了两套完整的 Windows 应用开发系统。如果人们愿意，仍然可以使用 Windows SDK 所提供的 API 来开发用 C 语言编写的 Windows 应用。Windows SDK 程序设计技术已经为人们所熟知，并且已经有许多书籍对此专门作了介绍。然而人们利用 Visual C++ 所提供的一些新的工具可以使编写 Windows 应用程序变得更加方便。

在本书并不介绍如何用 Windows SDK 来进行 Windows 程序设计，而是介绍如何用 Visual C++所提供的 MFC 库应用程序框架来进行 C++程序设计。这正是当今编程方法的潮流和趋势。不过，使用 Microsoft 基本类库程序设计接口并不意味着不能使用 Windows SDK 函数，实际上人们几乎总是需要在 MFC 库程序设计中直接调用 Windows SDK 函数。

浏览一下 Visual C++的各组成元素将有助于人们了解应用框架。

1.2.1 Visual 工作平台和 Visual C++工程

“工程”（Project）是一些相互关联的源文件的集合，这些源文件经过编译、链接，然后被组合在一起，形成可执行的 Windows 应用程序。工程源文件一般被存储在一个单独的子目录下，工程也需要依赖该子目录之外的许多文件，如包含文件和库文件等。

熟练的程序员对 make 文件不会感到陌生，它表述了所有源文件之间的关系(源代码文

件需要的特定包含文件，可执行文件要求包含的目标文件模块及库等等)。创建程序首先读取 make 文件，然后激活编译器、汇编器、链接器和资源编译器来产生最后的输出，最后输出并生成的通常是可执行文件。在命令行工作方式下，人们必须手工编写 make 文件，而 Visual 工作平台则能够自动生成 make 文件，不过此时称为“工程文件”（Project File）。图 1－1 描述了 Visual C++工程创建的大致过程。

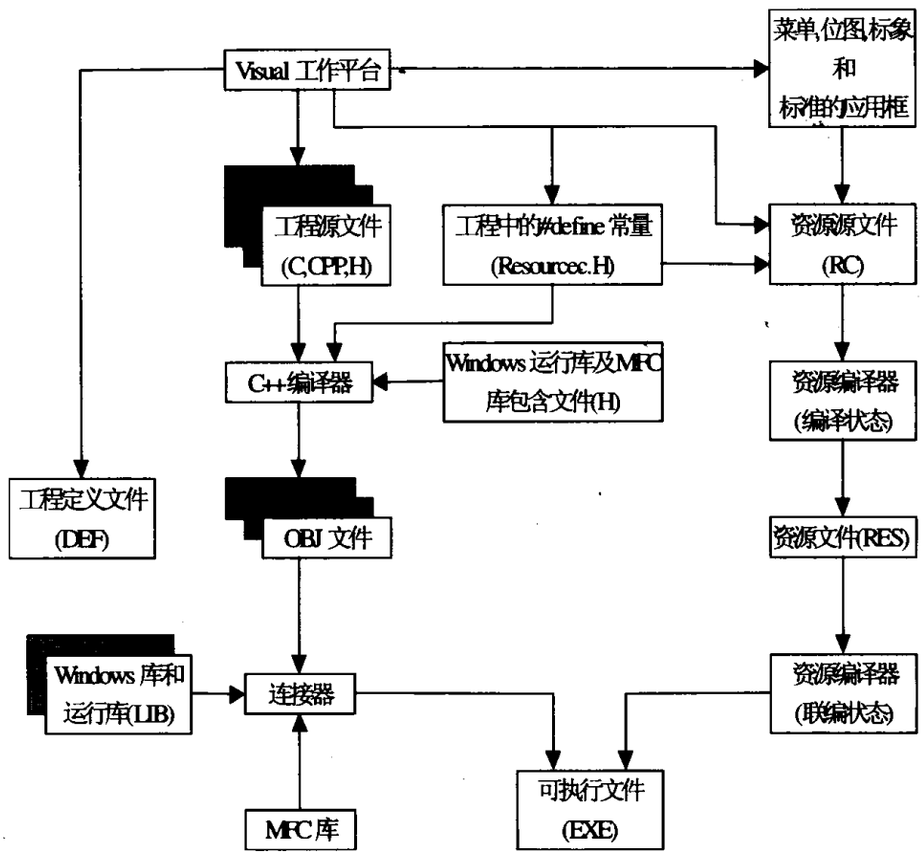

图 1-1　Visual C++工程创建过程

Visual 工作平台是一个运行于 Windows 上的交互式开发环境。这个工作平台不但能够自动生成工程文件，它还能够“记住”当前人们正处理的源代码文件，并且还会保存一份最近使用过的工程列表。作为工程的一部分，人们可以通过一系列对话框来保存所指定的编译器和链接器的开关设置。人们只要从 Visual 工作平台 Project 菜单中选择 Build 命令，系统就会自动生成可执行程序。

Visual 工作平台还包含了一个完全符合 Windows 界面标准的文本编辑器，该编辑器会用颜色来加亮 C++文法，并向人们动态提供类成员函数、函数参数、API 等各种用法的相关提示。

1.2.2　Visual C++ 6.0 工作平台

当用户在 Visual C++工作平台的上部工具栏空白处单击鼠标右键时，屏幕上会显示出如图 1－2 所示的选择框，这个选择框列出了平台中可以加载的各类工具条和工作窗口。用户只要在所选项目（工具条或窗口）上单击鼠标左键就可以将此项目在平台上显示或隐藏。这里的工具条几乎包括了工作平台的所有工具和窗口。同时，使用工具条将比使用菜单更加方

便和快捷，下面将对各类工具条展开论述，介绍常用工具条和各个工作窗口的用法。

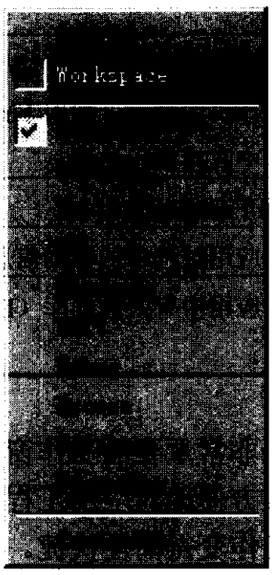

图 1-2 Visual C++工作平台的项目列表

1. Output 信息输出窗口

Output 窗口是一个信息显示窗口，如图 1－3 所示，用来显示编译、调试、查询及 SQL 数据库的查询结果，可以帮助用户修改编译错误，获得系统信息，以便于应用的调试和开发。这个窗口分为六个不同的页。这六个页各有不同的功能，用鼠标可以实现各页面间的切换。单击窗口左下角的一对箭头按钮，可以使页面条左右滚动，以显示出所有的页面选项。如果信息过长，可以拖动窗口右下角的水平滚动条，使当前页面的显示窗口左右移动，如果信息过多，则可以拖动窗口右边的垂直滚动条来查看。具体功能的描述如表 1－1。

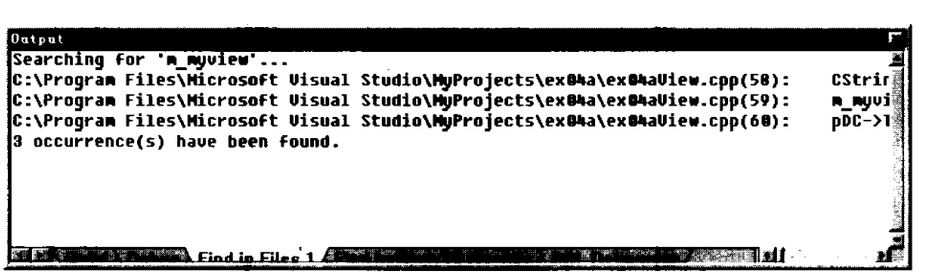

图 1-3 Output 信息输出窗口

表 1－1 信息输出窗口页面描述

页面名称	页面功能
Build	指示编译、连接、产生资源代码的过程以及编译连接的内容，并显示编译连接过程中的错误信息以及警告。错误信息包括错误代码、错误位置，错误描绘和错误总数。当在某条错误信息上双击鼠标时，光标会自动转移到错误代码处，并在错误行前以右箭头指示。
Debug	观察调试过程中变量值、窗口句柄以及代码指针，并可以显示用户在程序中设置的 TRACE、TRY、VERIFY 等调试信息。

续 表

页面名称	页面功能
Find In Files 1	在 Find In Files 对话框中可以设置查找结果输出的窗格，当设置为窗格一时则在该页面显示。如用 Find 组合框时，则此窗口不显示查找信息，而在文本中直接显示结果。
Find In Files 2	同 Find In Files 1。
Result	显示开发应用过程中的数据信息结果，不常用。
SQL Debugging	用于显示应用中有关数据库（DAO 或 ODBC）的 SQL 语言查询调试过程及结果。

2. WorkSpace 工作空间

WorkSpace 是 Visual C++的一个非常重要的工作窗口，对于程序员来说，这个窗口是必不可少的。在这个窗口里用户可以很方便地对应用工程的类、类成员函数、变量、资源以及每一个文件进行操作，从而使编程工作变得简便快捷。

如图 1－4 所示，WorkSpace 是一个多页的窗口，支持水平和垂直的滚动条，更方便的是，当水平方向的显示内容超出了窗口的显示大小时，只要程序员将鼠标移动到所要查看的内容上，这一行的全部内容就会以提示活动小窗口的形式显示出来，因此大大方便了程序员的操作。

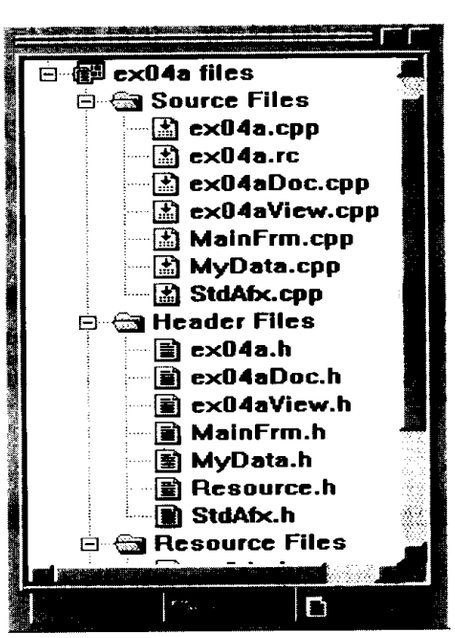

图 1-4 WorkSpace 工作空间

WorkSpace 包括三个页面，它们分别是 ClassView（类管理页）、ResourceView（资源管理页）和 FileView（文件管理页），程序员可以用鼠标单击，在这三个页之间切换。这三个页管理了不同的三个工程内容，这三个内容是相互独立的，又相互联系，同时通过操作还可以激活一系列工作窗口和对话框，从而把编程的全部过程有机地联系起来成为一个整体。

（1）ClassView 页

WorkSpace 的 ClassView 页用来管理工程中的类，它将工程中所有的类名和全局变量（Globals）都显示在此工作窗口中，如果程序员用鼠标单击所选类前面的加号，还可显示