微软程序设计核心技术丛书

Visual C++ 6.0

Visual C++6.0实用教程

杨 敏 编著

电子科技大学出版社

Microsoft

Visual C++ 6.0实用教程

杨 敏 编著

电子科技大学出版社

内容简介

本书以实用、易于学习为基本准则，系统地讲解了应用 Visual C++ 6.0 进行程序设计的编程基础及开发技巧。本书共分十二章:第一章概述 Windows 程序设计的相关内容；第二章介绍 Visual C++ 6.0 的开发环境、工程及编程步骤；第三章介绍 MFC 的相关内容；第四章介绍如何使用菜单和快捷键；第五章介绍了工具条和状态条的设计和控制；第六章介绍了如何使用键盘和鼠标；第七章介绍对话框的设计和控件的使用；第八章介绍如何创建文档、视图结构的应用程序；第九章介绍多媒体应用程序设计的相关内容及实例；第十章介绍数据库的应用；第十一章介绍 Internet 的应用；第十二章介绍 ActiveX 的相关内容。

本书可作为初学者系统学习 Visual C++ 6.0 的入门工具，也可作为中等水平人员精通编程的途径，尤其适合大中专院校作为程序设计教材。

图书在版编目（CIP）数据

Visual C++ 6.0 实用教程/杨敏编著. —成都：
电子科技大学出版社，2001.8
ISBN 7-81065-734-8

Ⅰ.V… Ⅱ.杨… Ⅲ.C 语言-程序设计-高等
学校-教材 Ⅳ.TP312

中国版本图书馆 CIP 数据核字（2001）第 062937 号

Visual C++ 6.0 实用教程

杨 敏 编著

出　　版：电子科技大学出版社（成都建设北路二段四号，邮编 610054）

责任编辑：陈建军

发　　行：电子科技大学出版社

印　　刷：四川五洲彩印有限责任公司

开　　本：787mm×1092 mm　1/16　印张 19.5　字数 479 千字

版　　本：2001 年 9 月第一版

印　　次：2001 年 9 月第一次

书　　号：ISBN 7-81065-734-8/TP・492

印　　数：0001－3000 册

定　　价：25.00 元

前　言

面向对象程序设计初出市面，便受到了热烈的欢迎，为程序设计的发展树立了另一座里程碑。它以其快捷、易懂、功能强大、界面友好等一系列优点而深得程序员们和程序设计爱好者的衷心喜爱。你应该注意到：面向对象程序设计的概念已经不再是天方夜谭，它正以光一般的速度发展，它已经触及到人们生活的各个方面，商店、银行、学校、政府机关、商业公司、家庭、个人，几乎无所不在。因此，计算机的实用程序设计者和程序设计的爱好者如果不能掌握一门面向对象的程序设计工具，那么他是落伍的，更是不能适应信息时代的前进速度的。Visual C++6.0 是美国 Microsoft 公司于 1998 年推出的基于 Windows 98 操作平台、功能强大的编程工具，它具有友好的图形化用户界面、面向对象的程序开发、完善而又丰富的库函数以及 Internet 网络应用等诸多功能，使广大程序开发人员能够利用它开发出各种有用的应用程序。

针对当前程序设计发展的形势和计算机书籍市场良莠不齐的现象，我们出版了这套程序设计实用教程，希望以我们的实战经验和心得体会，用浅显易懂的语言、循序渐进的方法向大家展示程序设计的基础知识和编程技巧。

本书主要讲解如何利用 Visual C++6.0 编程工具开发 Windows 程序的方法、技巧，内容丰富,涉及键盘和鼠标输入、菜单、各种对话框、图形和鼠标的程序、文件、多文档和多视图以及对象链接和嵌入等。并且着重介绍了 Windows 98 的常用控制、数据库的管理和 VisualC++6.0 在 Internet 网络中的应用。本书在编写时本着内容易学易懂、由浅入深、实用性强的原则，力求达到既能帮助初学者入门，也能使 VisualC++的老用户进一步提高编程水平的目的。尤其适合大中专院校作为程序设计教材。

计算机技术发展飞速，加上作者水平有限，编写当中难免有所疏漏和不足，希望广大读者给与批评和指正。

编　者

目 录

第1章 Windows 编程基础

- Windows 编程概述
- Windows 应用程序的优点
- 对象化的 Windows 编程

本章主要介绍了 Windows 编程的基础知识,包括了 Windows 的面向对象的编程方法,事件驱动的程序运行方式,图形化的界面显示。这些知识是进行 Windows 编程必不可缺的,只有了解了本章的内容才能更好地学习以后的章节。

1.1 Windows 编程介绍

1.1.1 Windows 概 述

微软在 1983 年春季开始研究开发 Windows。它在 1985 年和 1987 年分别推出 Windows 1.03 版和 Windows 2.0 版。但是，由于当时硬件和 DOS 操作系统的限制，这两个版本并没有取得很大的成功。此后，微软公司对 Windows 的内存管理、图形界面做了重大改进，使图形界面更加美观并支持虚拟内存。微软于 1990 年 5 月份推出 Windows 3.0 并一炮打红。这个“千呼万唤始出来”的操作系统一经面世便在商业上取得惊人的成功：不到 6 周，微软公司销出 50 万份 Windows 3.0 拷贝，打破了任何软件产品的 6 周销售记录，从而一举奠定了微软在操作系统上的垄断地位。

这之后推出的 Windows 3.1 对 Windows 3.0 作了一些改进，引入 TrueType 字体技术，这是一种可缩放的字体技术，它改进了性能；还引入了一种新设计的文件管理程序，改进了系统的可靠性。更重要的是增加对象链接和嵌入技术（OLE）和多媒体技术的支持。Windows 3.0 和 Windows 3.1 都必须运行于 MS-DOS 操作系统之上。

微软在 1995 年推出新一代操作系统 Windows 95（又名 Chicago），它可以独立运行而无需 DOS 支持。Windows 95 是操作系统发展史上一个里程碑式的作品，它对 Windows 3.1 版作了许多重大改进，包括：更加优秀的、面向对象的图形用户界面；全 32 位的高性能的抢先式多任务和多线程；内置的对 Internet 的支持；更加高级的多媒体支持（声音、图形、影像等），可以直接写屏并很好的支持游戏；即插即用，简化用户配置硬件操作，并避免了硬件上的冲突；32 位线性寻址的内存管理和良好的向下兼容性等等。这以后微软又推出了功能更加完善的 Windows 98,以后我们提到的 Windows 一般均指 Windows 98。

Windows 之所以取得成功，主要在于它具有以下优点：

1．直观、高效的面向对象的图形用户界面，易学易用

从某种意义上说，Windows 用户界面和开发环境都是面向对象的。用户采用“选择对象-操作对象”这种方式进行工作。比如要打开一个文档，我们首先用鼠标或键盘选择该文档，然后从右键菜单中选择“打开”操作，打开该文档。这种操作方式模拟了现实世界的行为，易于理解、学习和使用。

2．用户界面统一、友好、漂亮

Windows 应用程序大多符合 IBM 公司提出的 CUA（Common User Access）标准，所有的程序拥有相同的或相似的基本外观，包括窗口、菜单、工具条等。用户只要掌握其中一个，就不难学会其他软件，从而降低了用户培训学习的费用。

3．丰富的设备无关的图形操作

Windows 的图形设备接口（GDI）提供了丰富的图形操作函数，可以绘制出诸如线、圆、框等的几何图形，并支持各种输出设备。设备无关意味着在针式打印机上和高分辨率的显示器上都能显示出相同效果的图形。

4．多 任 务

Windows 是一个多任务的操作环境，它允许用户同时运行多个应用程序，或在一个程序中同时做几件事情。每个程序在屏幕上占据一块矩形区域，这个区域称为窗口，窗口是可以重叠的。用户可以移动这些窗口，或在不同的应用程序之间进行切换，并可以在程序之间进行手工和自动的数据交换和通信。

虽然同一时刻计算机可以运行多个应用程序，但仅有一个是处于活动状态的，其标题栏呈现高亮颜色。一个活动的程序是指当前能够接收用户键盘输入的程序。

1.1.2 Windows 应用程序的优点

如前所述，Windows 操作系统具有 MS-DOS 操作系统无可比拟的优点，因而受到了广大软件开发人员的青睐。但是，熟悉 DOS 环境下软件开发的程序员很快就会发现，Windows 编程与 DOS 环境下编程相比有很大的不同。Windows 要求以一种全新的思维方式进行程序设计，主要表现为以下几点：

1．事件驱动的程序设计

传统的 MS-DOS 程序主要采用顺序的、关联的、过程驱动的程序设计方法。一个程序是一系列预先定义好的操作序列的组合，它具有一定的开头、中间过程和结束。程序直接控制程序事件和过程的顺序。这样的程序设计方法是面向程序而不是面向用户的，交互性差，用户界面不够友好，因为它强迫用户按照某种不可更改的模式进行工作。它的基本模型如图 1-1 所示。

事件驱动程序设计是一种全新的程序设计方法，它不是由事件的顺序来控制，而是由事件的发生来控制，而这种事件的发生是随机的、不确定的，并没有预定的顺序，这样就允许程序的用户用各种合理的顺序来安排程序的流程。对于需要用户交互的应用程序来说，事件驱动的程序设计有着过程驱动方法无法替代的优点。它是一种面向用户的程序设

计方法，它在程序设计过程中除了完成所需功能之外，更多的考虑了用户可能的各种输入，并针对性的设计相应的处理程序。它是一种“被动”式程序设计方法，程序开始运行时，处于等待用户输入事件状态，然后取得事件并作出相应反应，处理完毕又返回并处于等待事件状态。它的框图如图 1-2 所示。

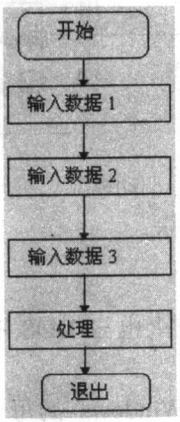

图 1-1　过程驱动模型

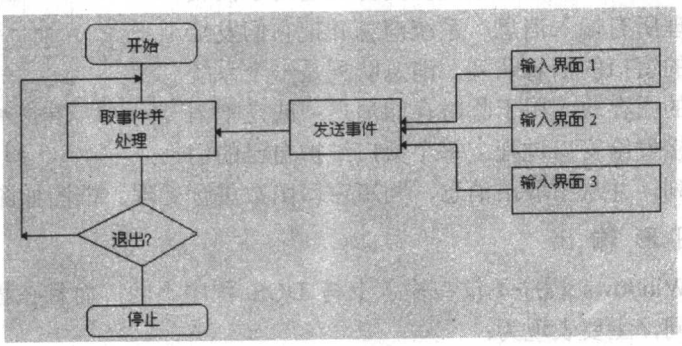

图 1-2　事件驱动程序模型

在图中，输入界面 1~3 并没有固定的顺序，用户可以随机选取，以任何合理的顺序来输入数据。

2. 消息循环与输入

事件驱动围绕着消息的产生与处理展开，一条消息是关于发生的事件的消息。事件驱动是靠消息循环机制来实现的。消息是一种报告有关事件发生的通知。

消息类似于 DOS 下的用户输入，但比 DOS 的输入来源要广，Windows 应用程序的消息来源有以下四种：

- 输入消息：包括键盘和鼠标的输入。这一类消息首先放在系统消息队列中，然后由 Windows 将它们送入应用程序消息队列中，由应用程序来处理消息。
- 控制消息：用来与 Windows 的控制对象，如列表框、按钮、检查框等进行双向通信。当用户在列表框中改动当前选择或改变了检查框的状态时发出此类消息。这类消息一般不经过应用程序消息队列，而是直接发送到控制对象上去。
- 系统消息：对程序化的事件或系统时钟中断作出反应。一些系统消息，像 DDE 消息（动态数据交换消息）要通过 Windows 的系统消息队列，而有的则不通过系统消息队列而直接送入应用程序的消息队列，如创建窗口消息。
- 用户消息：这是程序员自己定义并在应用程序中主动发出的，一般由应用程序的某一部分内部处理。

在 DOS 应用程序下，可以通过 getchar（）、getch（）等函数直接等待键盘输入，并直接向屏幕输出。而在 Windows 下，由于允许多个任务同时运行，应用程序的输入输出是由 Windows 来统一管理的。

Windows 操作系统包括三个内核基本元件：GDI、KERNEL、USER。其中 GDI（图

形设备接口）负责在屏幕上绘制像素、打印硬拷贝输出，绘制用户界面包括窗口、菜单、对话框等。系统内核 KERNEL 支持与操作系统密切相关的功能：如进程加载、文本切换、文件 I/O，以及内存管理、线程管理等。USER 为所有的用户界面对象提供支持，它用于接收和管理所有输入消息、系统消息并把它们发给相应的窗口的消息队列。消息队列是一个系统定义的内存块，用于临时存储消息；或是把消息直接发给窗口过程。每个窗口维护自己的消息队列，并从中取出消息，利用窗口函数进行处理。框图如图 1-3 所示。

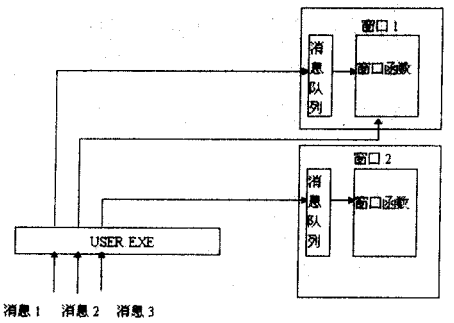

图 1-3 消息驱动模型

3. 图 形 输 出

Windows 程序不仅在输入上与 DOS 程序不同，而且在程序输出上也与 DOS 有着很大不同，主要表现为：

（1）DOS 程序独占整个显示屏幕，其他程序在后台等待。而 Windows 的每一个应用程序对屏幕的一部分进行处理。DOS 程序可以直接往屏幕上输出，而 Windows 是一个多窗口的操作系统，由操作系统来统一管理屏幕输出；每个窗口要输出内容时，必须首先向操作系统发出请求（GDI 请求），由操作系统完成实际的屏幕输出工作。

（2）Windows 程序的所有输出都是图形。Windows 提供了丰富的图形函数用于图形输出，这对输出图形是相当方便的，但是由于字符也被作为图形来处理，输出时的定位要比 DOS 复杂得多。

比如，在 DOS 字符方式下，我们可以写出如下程序用于输出两行文字：

```
printf（“Hello,\n”）；
printf（“This is DOS program.\n”）；
```

而在 Windows 下要输出这两行文字所做的工作要复杂得多。因为 Windows 输出是基于图形的，它输出文本时不会像 DOS 那样自动换行，而必须以像素为单位精确定位每一行的输出位置。另外，由于 Windows 提供了丰富的字体，所以在计算坐标偏移量时还必须知道当前所用字体的高度和宽度。

（3）Windows 下的输出是设备无关的。在 DOS 下编写过 Foxpro 程序的读者常常会有这样的体会，在编写打印报表程序时，要针对不同的打印机在程序中插入不同的打印控制码，用以控制换页、字体设置等选项。这样的程序编写起来繁琐，而且不容易移植（因为换一台不同型号的打印机就要重新修改程序）。而 Windows 下的应用程序使用图形设备接口（GDI）来进行图形输出。GDI 屏蔽了不同设备的差异，提供了设备无关的图形输出能力，Windows 应用程序只要发出设备无关的 GDI 请求（如调用 Rectangle 画一个矩形），由 GDI 去完成实际的图形输出操作。对于一台具有打印矩形功能的 PostScript 打印机来说，GDI 可能只需要将矩形数据传给驱动程序就可以了，然后由驱动程序产生 PostScript 命令绘制出相应的矩形；而对于一台没有矩形输出功能的点阵打印机来说，GDI 可能需要将矩形转化为四条线，然后向驱动程序发出画线的指令，在打印机上输出矩形。当然，这两种输出在用户看来并没有什么区别。

Windows 的图形输出是由图形设备接口（GDI）来完成的，GDI 是系统原始的图形输出库，它用于在屏幕上输出像素、在打印机上输出硬拷贝以及绘制 Windows 用户界面。

GDI 提供两种基本服务：创建图形输出和存储图像。GDI 提供了大量用于图形输出的函数，这些函数接收应用程序发出来的绘图请求、处理绘图数据并根据当前使用设备调用相应的设备驱动程序产生绘图输出。这些绘图函数分为三类：一是文字输出；二是矢量图形函数，用于画线、圆等几何图形；三是光栅（位图）图形函数，用于绘制位图。

GDI 识别四种类型的设备：显示屏幕、硬拷贝设备（打印机、绘图机）、位图和图元文件。前两者是物理设备，后两者是伪设备。一个伪设备提供了一种在 RAM 里或磁盘里存储图像的方法。位图存放的是图形的点位信息，占用较多的内存，但速度很快。图元文件保存的是 GDI 函数的调用和调用参数，占用内存较少，但依赖于 GDI，因此不可能用某个设备来创建图元文件，而且速度比位图要慢。

GDI 的图形输出是面向窗口的，面向窗口包含两层含义：

- 每个窗口作为一个独立的绘图接口来处理，有它自己的绘图坐标。当程序在一个窗口中绘图时，首先建立缺省的绘图坐标，原点（0，0）位于窗口用户区的左上角。每个窗口必须独立的维护自己的输出。
- 绘图仅对于本窗口有效，图形在窗口边界会被自动裁剪，也就是说窗口中的每一个图形都不会越出边界。即使想越出边界，也是不可能的，窗口会自动地防止其他窗口传过来的任何像素。这样，你在窗口内绘图时，就不必担心会偶然覆盖其他程序的窗口，从而保证了 Windows 下同时运行多个任务时各个窗口的独立性。

4. 用户界面对象

Windows 支持丰富的用户接口对象，包括：窗口、图标、菜单、对话框等等。程序员只需简单的几十行代码，就可以设计出一个非常漂亮的图形用户界面。而在 DOS 环境下，则需要大量的代码来完成同样的工作，而且效果也没有 Windows 提供的那么好。下面我们介绍一下用户界面对象中的一些术语和相关概念。

- 窗口

窗口是用户界面中最重要的部分。它是屏幕上与一个应用程序相对应的矩形区域，是用户与产生该窗口的应用程序之间的可视界面。每当用户开始运行一个应用程序时，应用程序就创建并显示一个窗口；当用户操作窗口中的对象时，程序会作出相应反应。用户通过关闭一个窗口来终止一个程序的运行；通过选择相应的应用程序窗口来选择相应的应用程序。一个典型的窗口外观如图 1-4 所示。

- 边框

绝大多数窗口都有一个边框，用于指示窗口的边界。同时也用来指明该窗口是否为活动窗口，当窗口活动时，边框的标题栏部分呈高亮显示。用户可以用鼠标拖动边框来调整窗口的大小。

- 系统菜单框

系统菜单框位于窗口左上角，以当前窗口的图标方式显示，用鼠标点一下该图标（或按 ALT+空格键）就弹出系统菜单。系统菜单提供标准的应用程序选项，包括：Restore（还原窗口原有的大小），Move（使窗口可以通过键盘上的光标键来移动其位置），Size（使

用光标键调整窗口大小），Minimize（将窗口缩成图标），Maximize（最大化：使窗口充满整个屏幕）和 Close（关闭窗口）。

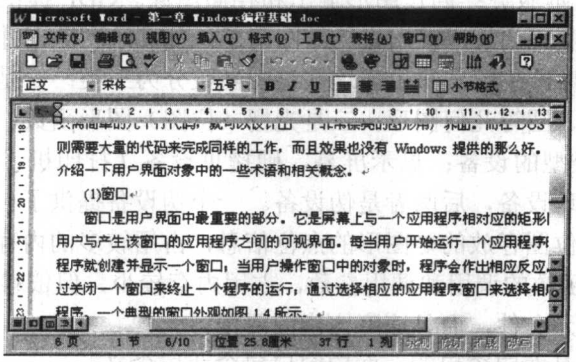

图 1-4　窗口

● 标题栏

标题栏位于窗口的顶部，其中显示的文本信息用于标注应用程序，一般是应用程序的名字，以便让用户了解哪个应用程序正在运行。标题栏颜色反映该窗口是否是一个活动窗口，当为活动窗口时，标题栏呈现醒目颜色。鼠标双击标题栏可以使窗口在正常大小和最大化状态之间切换。在标题栏上按下鼠标器左键可以拖动并移动该窗口，按右键弹出窗口系统菜单。

● 菜单栏

菜单栏位于标题栏下方，横跨屏幕，在它上面列出了应用程序所支持的命令，菜单栏中的项是命令的主要分类，如文件操作、编辑操作。从菜单栏中选中某一项通常会显示一个弹出菜单，其中的项是对应于指定分类中的某个任务。通过选择菜单中的一个项（菜单项），用户可以向程序发出命令，以执行某一功能。如选择“文件->打开…”菜单项会弹出一个打开文件对话框，让用户选择一个文件，然后打开这个文件。

一般地，以“…”结尾的菜单项文本表明选择该项时会弹出一个对话框，让用户输入信息，然后执行操作，如“文件->打开…”。若不以“…”结尾，则表明选择该菜单项直接执行一个动作，如“编辑”菜单下的“粘贴”菜单项。若一个菜单项呈现灰色，则表明该菜单当前不可用。有时菜单项上还有加速键，加速键是一种键盘组合，它是菜单项的一种替代方式，可以让用户通过键盘直接发出命令；在键盘上按下这一键盘组合，就等效于选择了相应的菜单。如“粘贴（P）CTRL+V”，就表示粘贴操作的快捷键是 CTRL+V，按下 CTRL+V 就执行粘贴操作。

● 工具条

工具条一般位于菜单栏下方，在它上面有一组位图按钮，代表一些最常用的命令。工具条可以显示或隐藏。让鼠标在某个按钮上停一会儿，在按钮下方会出现一个黄色的小窗口，里面显示关于该按钮的简短说明，叫做工具条提示（ToolTip）。用户还可以用鼠标拖动工具条将其放在窗口的任何一侧。

● 客户区

客户区是窗口中最大的一块空白矩形区域，用于显示应用程序的输出。例如，字处

理程序在客户区中显示文档的当前页面。应用程序负责客户区的绘制工作，而且只有和该窗口相对应的应用程序才能向该用户区输出。

● 垂直滚动条和水平滚动条

垂直滚动条和水平滚动条分别位于客户区的左侧和底部，它们各有两个方向相反的箭头和一个深色的长度可变的滚动块。可以用鼠标选中滚动条的箭头上下卷滚（选中垂直滚动条时）或水平卷滚（选中水平滚动条时）客户区的内容。滚动块的位置表示客户区中显示的内容相对于要显示的全部内容的位置，滚动块的长度表示客户区中显示的内容大小相对于全部内容大小的比例。

● 状态栏

状态栏是一般位于窗口底部，用于输出菜单的说明和其他一些提示信息（如鼠标器位置、当前时间、某种状态等）。

● 图标

图标是一个用于提醒用户的符号，它是一个小小的图像，用于代表一个应用程序。当一个应用程序的主窗口缩至最小时，就呈现为一个图标。

● 光标

Windows 的光标是显示屏上的一个位图，而不是 DOS 下的一条下划线。光标用于响应鼠标或其他定位设备的移动。程序可以通过改变光标的形状来指出系统中的变化。例如，程序常显示一个计时的光标，用于指示用户一些漫长的操作正在进行之中。程序也可以通过改变光标让用户知道程序进入了一种特殊模式，例如，绘图程序经常改变光标来反映被绘制对象的类型，是直线还是圆或其他。

● 插入符

插入符（caret）是一个微小并闪烁的位图，作为一个键盘控制的指针。控制键盘输入的窗口可以创建一个插入符去通知用户：窗口现在可以进行键盘输入。在 DOS 环境下，一般使用“光标”作为键盘指针，而在 Windows 中，“光标”被作为鼠标指针。

应用程序必须维护这个插入符。在 Windows 中，在一个时间只允许有一个插入符存在。因此，要使用插入符号作为键盘指针的应用程序必须在取得焦点时创建一个插入符号，并在失去焦点后删除它。

● 对话框

对话框是一种特殊的窗口，它提供了一种接收用户输入、处理数据的标准方法。特别的，当用户输入了一个需要附加信息的命令时，对话框是接收输入的标准方法。比如，假设用户要求系统打开一个文件，对话框就可以提供一个让用户从一组文件中选择一个文件的标准方法。如前所述，在一般情况下，在选择菜单名字后面跟着省略号（…）的菜单项通常会弹出一个对话框。图 1-5 给出了查找对话框的一个例子。

● 控件

在图 1-5 中，查找对话框是一个独立的窗口，它显示信息并接收用户的输入。在对话框中，还包含了许多小的窗口，这些窗口被称为控件。控件是应用程序用来获得用户特定信息的窗口，比如要打开文件的名字或自动换行的设置等。应用程序也会通过控件获取所需的信息，以便控制程序的某种属性，如自动换行特性的开关。

控件总是与其他窗口连用，典型的是对话框，但也可以用在普通窗口之中。常见的

控件有：按钮、编辑框、列表框、组合框、静态文本等等。

- 消息框

是用于给用户一些提示或警告的窗口。例如，能够在应用程序执行某项任务过程中出现问题时通知用户。图 1-6 所示的对话框警告用户输入了一个不合法的文件名。

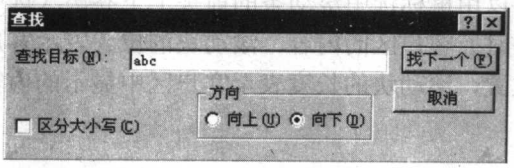

图 1-5 查找对话框

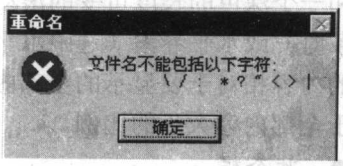

图 1-6 消息框

5. 资 源 共 享

对于 DOS 程序来说，它运行时独占系统的全部资源，包括显示器、内存等，在程序结束时才释放资源。而 Windows 是一个多任务的操作系统，各个应用程序共享系统提供的资源，常见的资源包括：设备上下文，画刷，画笔，字体，对话框控制，对话框，图标，定时器，插入符号，通信端口，电话线等。

Windows 要求应用程序必须以一种能允许它共享 Windows 资源的方式进行设计，它的基本模式是这样的：

- 向 Windows 系统请求资源。
- 使用该资源。
- 释放该资源给 Windows 以供别的程序使用。

即使最有经验的 Windows 程序员也常常会忽略第三步。如果忽略了这一步，轻则当时不出错，但过一会儿程序运行出现异常情况，或干扰别的程序正常运行；重则立即死机，比如设备上下文没有释放时。

在 Windows 应用程序设计中，CPU 也是一种非常重要的资源，因此应用程序应当避免长时间地占用 CPU 资源（如一个特别长的循环）；如果确实需要这样做，也应当采取一些措施，以让程序能够响应用户的输入。主存也是一个共享资源，要防止同时运行的多个应用程序因协调不好而耗尽内存资源。

应用程序一般不要直接访问内存或其他硬件设备，如键盘、鼠标、计数器、屏幕或串口、并口等。Windows 系统要求绝对控制这些资源，以保证向所有的应用程序提供公平的不中断的运行。如果确实要访问串并口，应当使用通过 Windows 提供的函数来安全的访问。

6. Windows 应用程序组成

前面介绍了 Windows 应用程序的特点，现在让我们看看编写一个 Windows 程序需要做哪些工作。编写一个典型的 Windows 应用程序，一般需要：

- C，CPP 源程序文件：源程序文件包含了应用程序的数据、类、功能逻辑模块（包括事件处理、用户界面对象初始化以及一些辅助例程）的定义。
- H，HPP 头文件：头文件包含了 CPP、C 源文件中所有数据、模块、类的声明。当一个 CPP、C 源文件要调用另一个 CPP、C 中所定义的模块功能时，需要包含

那个 CPP、C 文件对应的头文件。

- 资源文件：包含了应用程序所使用的全部资源定义，通常以.RC 为后缀名。

这里说的资源不同于前面提到的资源，这里的资源是应用程序所能够使用的一类预定义工具中的一个对象，包括：字符串资源、加速键表、对话框、菜单、位图、光标、工具条、图标、版本信息和用户自定义资源等。

其中 CPP、C 和头文件同 DOS 下的类似，需要解释的是资源文件。在 DOS 程序设计过程中，所有的界面设计工作都在源程序中完成；而在 Windows 程序设计过程中，像菜单、对话框、位图等可视的对象都是在资源文件中定义的。

1.1.3 对象化的 Windows 编程

面向对象技术是目前流行的系统设计开发技术，它包括面向对象分析和面向对象程序设计。面向对象程序设计技术的提出，主要是为了解决传统程序设计方法：结构化程序设计所不能解决的代码重用问题。

结构化程序设计从系统的功能入手，按照工程的标准和严格的规范将系统分解为若干功能模块，系统是实现模块功能的函数和过程的集合。由于用户的需求和软、硬件技术的不断发展变化，按照功能划分设计的系统模块必然是易变的和不稳定的。这样开发出来的模块可重用性不高。

面向对象程序设计从所处理的数据入手，以数据为中心而不是以服务（功能）为中心来描述系统。它把编程问题视为一个数据集合，数据相对于功能而言，具有更强的稳定性。

面向对象程序设计同结构化程序设计相比最大的区别就在于：前者首先关心的是所要处理的数据，而后者首先关心的是功能。

面向对象程序设计是一种围绕真实世界的概念来组织模型的程序设计方法，它采用对象来描述问题空间的实体。关于对象这一概念，目前还没有统一的定义。一般认为，对象是包含现实世界物体特征的抽象实体，它反映了系统为之保存信息和（或）与它交互的能力。它是一些属性及服务的一个封装体，在程序设计领域，可以用“对象=数据+作用于这些数据上的操作”这一公式来表达。

类是具有相同操作功能和相同的数据格式（属性）的对象的集合。类可以看做抽象数据类型的具体实现。抽象数据类型是数据类型抽象的表示形式。数据类型是指数据的集合和作用于其上的操作的集合，而抽象数据类型不关心操作实现的细节。从外部看，类型的行为可以用新定义的操作加以规定。类为对象集合的抽象，它规定了这些对象的公共属性和方法；对象为类的一个实例。苹果是一个类，而放在桌上的那个苹果则是一个对象。对象和类的关系相当于一般的程序设计语言中变量和变量类型的关系。

消息是向某对象请求服务的一种表达方式。对象内有方法和数据，外部的用户或对象对该对象提出的服务请求，可以称为向该对象发送消息。合作是指两个对象之间共同承担责任和分工。

1. 面向对象的编程方法具有四个基本特征

● 抽象

抽象就是忽略一个主题中与当前目标无关的那些方面，以便更充分地注意与当前目标有关的方面。抽象并不打算了解全部问题，而只是选择其中的一部分，暂时不用部分细节。比如，我们要设计一个学生成绩管理系统，考查学生这个对象时，我们只关心他的班级、学号、成绩等，而不用去关心他的身高、体重这些信息。抽象包括两个方面，一是过程抽象，二是数据抽象。过程抽象是指任何一个明确定义功能的操作都可被使用者看做单个的实体看待，尽管这个操作实际上可能由一系列更低级的操作来完成。数据抽象定义了数据类型和施加于该类型对象上的操作，并限定了对象的值只能通过使用这些操作修改和观察。

● 继承

继承是一种联结类的层次模型，并且允许和鼓励类的重用，它提供了一种明确表述共性的方法。对象的一个新类可以从现有的类中派生，这个过程称为类继承。新类继承了原始类的特性，新类称为原始类的派生类（子类），而原始类称为新类的基类（父类）。派生类可以从它的基类那里继承方法和实例变量，并且类可以修改或增加新的方法使之更适合特殊的需要。这也体现了大自然中一般与特殊的关系。继承性很好地解决了软件的可重用性问题。比如说，所有的 Windows 应用程序都有一个窗口，它们可以看做都是从一个窗口类派生出来的。但是有的应用程序用于文字处理，有的应用程序用于绘图，这是由于派生出了不同的子类，各个子类添加了不同的特性。

● 封装

封装是面向对象的特征之一，是对象和类概念的主要特性。封装是把过程和数据包围起来，对数据的访问只能通过已定义的界面。面向对象计算始于这个基本概念，即现实世界可以被描绘成一系列完全自治、封装的对象，这些对象通过一个受保护的接口访问其他对象。一旦定义了一个对象的特性，则有必要决定这些特性的可见性，即哪些特性对外部世界是可见的，哪些特性用于表示内部状态。在这个阶段定义对象的接口。通常，应禁止直接访问一个对象的实际表示，而应通过操作接口访问对象，这称为信息隐藏。事实上，信息隐藏是用户对封装性的认识，封装则为信息隐藏提供支持。封装保证了模块具有较好的独立性，使得程序维护修改较为容易。对应用程序的修改仅限于类的内部，因而可以将应用程序修改带来的影响减少到最低限度。

● 多态性

多态性是指允许不同类的对象对同一消息作出响应。比如同样的加法，把两个时间加在一起和把两个整数加在一起肯定完全不同。又比如，同样的选择编辑-粘贴操作，在字处理程序和绘图程序中有不同的效果。多态性包括参数化多态性和包含多态性。多态性语言具有灵活、抽象、行为共享、代码共享的优势，很好地解决了应用程序函数同名问题。

2. 面向对象程序设计具有许多优点

- 开发时间短，效率高，可靠性高，所开发的程序更强壮。由于面向对象编程的可重用性，可以在应用程序中大量采用成熟的类库，从而缩短了开发时间。
- 应用程序更易于维护、更新和升级。继承和封装使得应用程序的修改带来的影响