

第 1 章 了解 Visual Basic

本章要点

- Visual Basic 简介
- 熟悉 Visual Basic 6.0 集成开发环境
- 怎样学好 Visual Basic

本章以 Visual Basic 6.0 为例，介绍 Visual Basic 的特点、集成开发环境及其相关窗口、菜单和控件，使初学者尽快熟悉 Visual Basic 6.0 的工作环境。

1.1 Visual Basic 简介

1.1.1 Visual Basic 6.0 简介

Visual Basic 6.0（简称 VB6）是 Microsoft 公司于 1998 年推出的可视化开发工具。Visual 意为“可视化的”。它是指开发图形用户界面（GUI）时，无须编写大量代码去描述界面元素的位置和外观，仅通过简单的鼠标拖放操作就可以“所见即所得”的方式设计出标准的 Windows 应用程序界面。Basic 是指 BASIC 语言（Beginner's All-purpose Symbolic Instruction Code，初学者通用符号指令代码），这是一种面向初学者的编程语言。Visual Basic 是基于 BASIC 的可视化程序设计语言，它继承了 BASIC 语言简单易懂的特点，采用面向对象、事件驱动的编程机制，提供了直观的可视化程序设计方法。

在以 Windows 操作系统为平台的众多可视化编程工具中，Visual Basic 是最简单、最容易使用的语言，因此是初学者学习可视化编程语言的最佳选择。Visual Basic 6.0 是目前面向对象开发的主要语言之一，易用性、通用性和开发效率高等特点，这使得 Visual Basic 6.0 特别适合于一般应用程序的开发，成为最流行的 Windows 应用程序开发语言。

1.1.2 Visual Basic 6.0 的三种版本

学习版（Learning）：Visual Basic 6.0 学习版是个人版本，具有建立一般 Windows 应用程序所需要的全部工具。学习版适合于初学者和教学使用。

专业版（Professional）：Visual Basic 6.0 专业版是针对计算机专业人员的，具有某些高级特性，如包括 ActiveX 和 Internet 控件开发工具。专业版适合于专业程序员使用。

企业版 (Enterprise)：Visual Basic 6.0 企业版是最高级的版本，它是企业用户开发分布式应用程序的强大的编程工具，也是目前使用最多的版本。我们在后面的学习以企业版为例介绍 Visual Basic 6.0。

1.1.3 Visual Basic 6.0 的主要特点

1. 面向对象的可视化设计平台

VB 提供的面向对象的可视化设计平台将 Windows 应用程序界面设计的复杂性封装起来。程序员不必为界面设计编写大量代码，只需按照设计方案，用系统提供的工具在界面上“画出”各种对象即可。界面设计的代码将由 VB 自动生成，程序员所需编写的只是实现程序特定功能的那部分代码，从而大大提高了开发效率。

2. 事件驱动的编程机制

VB 通过事件执行对象的操作，即在响应不同事件时执行不同的代码段。事件可以由用户操作（如鼠标或键盘操作等）触发，也可以由系统（如应用程序本身、操作系统或其他应用程序的消息等）触发。

3. 结构化的程序设计语言

VB 具有丰富的数据类型和内部函数，编程语言模块化、结构化、简单易懂。

4. 强大的数据库功能和网络开发功能

VB 可以访问所有主流数据库，包括各种桌面数据库和大型网络数据库。用 VB 可以开发出功能完善的数据库应用程序。Visual Basic 6.0 对后台数据库的访问主要是通过 ADO (ActiveX Data Object) 实现的。ADO 是目前应用范围最广的数据访问接口，在 VB 中可以非常方便地使用 ADO 数据控件和 ADO 编程模型，通过 VB 本身或第三方提供的 OLE DB 和 ODBC 驱动程序访问各种类型的数据库。

Visual Basic 6.0 提供了一系列 Internet 开发工具，可以快速地开发 Web 应用程序，如 DHTML 工具可以使在 Visual Basic 6.0 中编写的程序代码直接用在动态网页设计中。

5. 充分利用 Windows 资源

VB 通过动态数据交换 (DDE)、对象链接与嵌入 (OLE) 和动态链接库 (DLL) 技术实现与 Windows 资源的交互。在 Visual Basic 6.0 中引入的 ActiveX 技术扩展了原有的 OLE 技术，使开发人员摆脱了特定语言的束缚，能够用 VB 开发集文字、声音、图像、动画、电子表格、数据库和 Web 对象于一体的应用程序。

6. 方便实用的程序向导

利用 VB 提供的多种向导可以方便快捷地创建不同类型和功能的应用程序，如应用程序向导、数据窗体向导、数据对象向导、打包和展开向导、工具栏向导、类生成器和 ActiveX 控件接口向导等。

1.2 Visual Basic 能做什么

VB 提供了开发 Windows 应用程序的最迅速、最简捷的方法。不论是资深专业开发人员还是初学者，都可以用 VB 开发应用程序。专业人员可以用 VB 开发出功能完善的大型应用系统，而初学者只要掌握一些关键词就可以建立实用的应用

程序。从开发个人或小组使用的小工具，到大型企业级应用系统，甚至通过 Internet 遍及全球的分布式应用程序，均可以在 VB 提供的工具中各取所需。

事实上，目前在 Windows 平台上运行的各种软件几乎都可以用 VB 编制。一些标准的窗口界面程序，如记事本、画图等，可以用 VB 来完成。常见的游戏软件如五子棋、扑克、扫雷、俄罗斯方块等也可以用 VB 编写。即使是大型的数据处理软件，同样可以用 VB 作为开发工具，世界著名的三大统计软件之一 SPSS 就是用 VB 开发的。这些例子只是 VB 应用中的沧海一粟。从设计新型的用户界面到利用其他应用程序的对象，从处理文字、图像到使用数据库，Visual Basic 提供了完成这些工作的所有工具。只要充分发挥个人的创意，你能想到的程序几乎都可以用 VB 开发出来。

随着 Visual Basic 的不断改进，它已经彻底摆脱了“玩具语言”的形象，成为编制通用 Windows 应用程序、数据库应用程序、多媒体应用程序以及网络应用程序的“几乎无所不能”的理想工具。

1.3 熟悉 Visual Basic 6.0 的开发环境

Visual Basic 6.0 不仅是一种编程语言，而且是集应用程序开发、调试和测试于一体的集成开发环境（IDE）。

1.3.1 启动 Visual Basic 6.0

单击【开始】，指向【程序】，指向【Microsoft Visual Basic 6.0 中文版】，单击【Microsoft Visual Basic 6.0 中文版】命令，即可启动 VB，出现如图 1.1 所示的【新建工程】对话框。该对话框中有三个选项卡：

（1）新建：创建新工程。选项卡中列出了 Visual Basic 6.0 能够建立的应用程序类型，其中【标准 EXE】为默认选项，初学者选择此项即可。

（2）现存：用于选择并打开现有的工程。

（3）最新：列出了最近打开过的工程及其所在文件夹。



图 1.1 新建工程对话框

1.3.2 Visual Basic 6.0 集成开发环境概貌

在【新建窗口】对话框中单击【打开】按钮即可进入 Visual Basic 6.0 的集成开发环境，如图 1.2 所示。

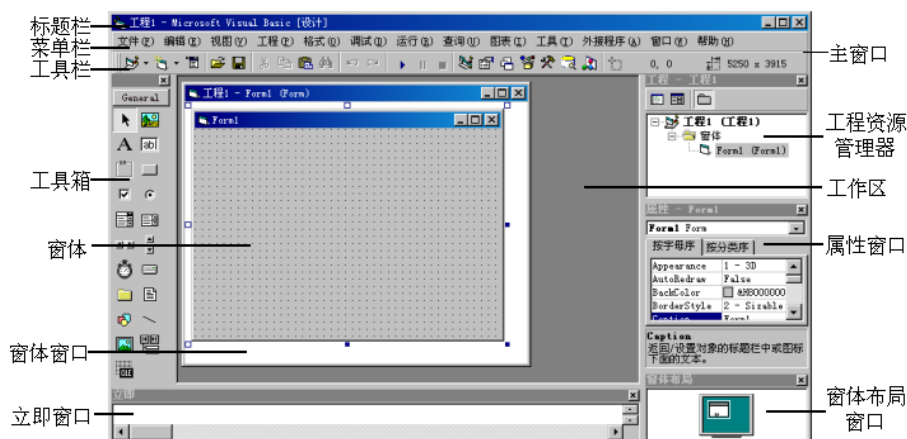


图 1.2 Visual Basic 6.0 集成开发环境

集成开发环境中主要包含以下窗口：主窗口 (Main Windows)、工具箱 (Tool box)、窗体 (Form) 窗口、工程资源管理器 (Project Explorer) 窗口、属性 (Properties) 窗口以及窗体布局 (Form Layout) 窗口等。图 1.2 中除窗体窗口外，其他各窗口均处于“停靠”状态。双击某窗口的标题栏，可使该窗口呈浮动状态，再次双击标题栏可恢复停靠状态。

1.3.3 主窗口

主窗口由图 1.2 所示的集成开发环境顶部的标题栏、菜单栏和工具栏以及下面的工作区组成。

1. 标题栏

图 1.2 标题栏中的标题为“工程 1-Microsoft Visual Basic [设计]”，说明 VB 集成开发环境正处于“工程 1”的设计状态。当进入其他状态时，方括号中的文字会有相应的变化。Visual Basic 6.0 有三种工作模式：设计模式 (Design)、运行模式 (Run) 和中断模式 (Break)。

- 设计模式：可设计用户界面和编写代码，进行应用程序的开发。
- 运行模式：运行应用程序，此时不允许编辑界面和代码。
- 中断模式：应用程序的运行暂停中断，此时可以编辑代码，但不能编辑界面。

2. 菜单栏

菜单栏包括 13 个菜单标题，含有 Visual Basic 6.0 中用到的全部命令：

- 文件：用于新建、打开、保存、添加、移除工程以及生成可执行文件等。
- 编辑：用于代码和控件的编辑。
- 视图：用于显示或切换集成开发环境中的各种窗口以及显示或隐藏特定工具栏。
- 工程：用于工程的管理，如添加或移除窗体、模块和部件，工程属性等。

- 格式：用于窗体中控件的能者为师方式、大小调整、设置间距和锁定等操作。
- 调试：用于应用程序的调试，如断点的设置、变量的监视、单步执行等命令。
- 运行：用于启动、中断和停止应用程序的运行。
- 查询：在建立数据库应用程序时用于设置结构化查询。
- 图表：在建立数据库应用程序时用于编辑图表。
- 工具：用于添加过程、设置过程属性、调用菜单编辑器、设置集成开发环境等。
- 外接程序：用于增加或删除外接程序。
- 窗口：用于相关窗口的开启、关闭和排列。
- 帮助：用于获取相关的帮助信息。

3. 工具栏

利用工具栏可以快速访问常用的菜单命令，默认的工具栏为标准工具栏，如图 1.3 所示。可以通过【视图】 | 【工具栏】菜单中的相关命令自定义工具栏。



图 1.3 标准工具栏

标准工具栏各按钮的功能如表 1.1 所示。

表 1.1 标准工具栏按钮的功能

图标	对应菜单及命令	功能	快捷键
	“文件” 菜单 “添加工程” 命令，	向工程组中添加新工程	
	“工程” 菜单 “添加窗体” 命令	添加一个新窗体到当前工程中	
	“工具” 菜单 “菜单编辑器” 命令	启动菜单编辑器进行菜单编辑	Ctrl+E
	“文件” 菜单 “打开” 命令	打开已有的工程	Ctrl+O
	“文件” 菜单 “保存” 命令	保存当前的工程文件	
	“编辑” 菜单 “剪切” 命令	剪切选定的内容到剪贴板	Ctrl+X
	“编辑” 菜单 “复制” 命令	复制选定的内容到剪贴板	Ctrl+C
	“编辑” 菜单 “粘贴” 命令	将剪贴板中的内容粘贴到当前光标所在位置	Ctrl+V
	“编辑” 菜单 “查找” 命令	查找符合条件的字符串	Ctrl+F
	“编辑” 菜单 “撤销” 命令	撤销上一次的操作	Ctrl+Z
	“编辑” 菜单 “恢复” 命令	恢复刚才撤销的操作	
	“运行” 菜单 “启动” 命令	开始运行程序	F5
	“运行” 菜单 “中断” 命令	暂时中断程序的运行	Ctrl+Break
	“运行” 菜单 “结束” 命令	结束程序的运行	
	“视图” 菜单 “工程资源管理器” 命令	显示工程资源管理器窗口	Ctrl+R
	“视图” 菜单 “属性窗口” 命令	显示属性窗口	F4
	“视图” 菜单 “窗体布局窗口” 命令	显示窗体布局窗口	
	“视图” 菜单 “对象浏览器” 命令	显示对象浏览器窗口，查找所有对象	F2
	“视图” 菜单 “工具箱” 命令	显示工具箱窗口	
	“视图” 菜单 “数据视图窗口” 命令	显示数据视图窗口	
	“视图” 菜单 “Visual Component Manager” 命令	打开可视化组件管理器	

4. 工作区

主窗口工具栏下面的深灰色区域是工作区。工作区是其他各窗口的容器。开发应用程序时可根据程序设计的需要，通过【视图】菜单或工具栏按钮在工作区中显示相关窗口。

1.3.4 窗体窗口

窗体窗口又称为“对象窗口”或“窗体设计器”。通过【视图】|【对象窗口】命令可以打开窗体窗口。窗体窗口是设计用户界面的地方。窗体（Form）就是应用程序的用户界面，是组成应用程序的最基本的元素。一个窗体窗口只含有一个窗体，因此，如果应用程序由多个窗体组成，在设计时就会有多个窗体窗口。每个窗体必须具有惟一的名称，建立窗体时系统默认的窗体名称依次为 Form1、Form2、Form3 等。

1.3.5 工程资源管理器窗口

在 VB 中，工程是指用于创建应用程序的所有文件的集合。工程资源管理器窗口（简称工程窗口）用于显示和管理当前程序中所包含的全部文件，如图 1.4 所示。工程窗口由三部分组成，自上而下分别为标题栏、工具栏和文件列表。

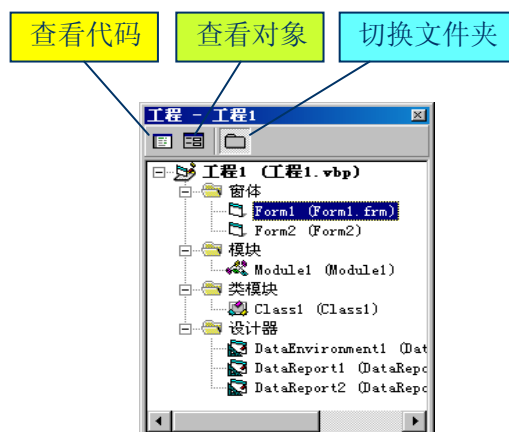


图 1.4 工程资源管理器

1. 标题栏

显示当前工程（组）的名称。

2. 工具栏

由三个按钮组成。【查看代码】按钮用于显示代码窗口，查看和编辑代码；【查看对象】按钮用于显示窗体窗口，查看和编辑正在设计的窗体；【切换文件夹】按钮用于显示或隐藏文件夹。

3. 文件列表

文件列表显示程序中包含的各种文件。每个工程和文件夹前有一个小方框，状态为“+”或“-”，其中“+”为展开按钮，单击后显示相应工程或文件夹包含文件的详细列表。“-”为折叠按钮，单击后隐藏相应工程或文件夹包含文件的详细列表。

从图 1.4 中可以看出，文件详细列表中的每一项由括号内外两部分组成。括号外面的部分为该文件在应用程序内部使用的名称（编写代码时使用）；括号内是该文件保存在磁盘上的文件名，其中有扩展名的（如 Form1.frm）表示已保存过，无扩展名的则表示尚未存盘。

1.3.6 属性窗口

属性窗口如图 1.5 所示，用于设置窗体和控件的属性，如名称、外观、位置、字体等。属性窗口由五部分组成。



图 1.5 属性窗口

1. 标题栏

显示当前工程选定的窗体或控件的名称。

2. 对象下拉列表框

对象下拉列表框中含有当前窗体及其所包含的列表。单击右端的下拉按钮，可列出所有对象以供选择。列表中的每一项（行）代表一个对象，其内容分为左右两部分。左侧以粗体显示的部分为对象名称，右侧以标准字体显示的部分为该对象所属的类。如在图 1.5 所示的属性窗口对象列表框中，左侧的 Form1 为对象名称，右侧的 Form 表示该对象属于窗体（Form）类。

3. 属性显示排列方式

对象列表框下方的两个选项卡用于确定属性显示的排列方式。

- 按字母序：各属性按照英文字母顺序排列。
- 按分类序：各属性按照一定的分类规则顺序排列。

4. 属性列表框

属性列表框用于列出所选对象可以设置的属性及其默认值。当在窗口中或者在属性窗口的对象列表框中选择了相应的对象时，系统就会将此对象的全部可以设置的属性列出，并给出默认值。对于不同对象所列出的属性不同。属性列表分左右两列，左边是各属性的名称，右边是对应的属性值。操作时在左侧选择属性，在右侧设置属性值。

有的属性可以直接输入值，也有些属性不允许输入，只能从给出的选项中选择，或者打开相应的对话框进行设置。

5. 属性说明框

属性说明框在属性窗口的底部，用于显示当前选中的属性的名称，并对其功能进行简要说明。

1.3.7 窗体布局窗口

窗体布局窗口如图 1.6 所示，用于指定程序运行时窗体的初始位置。在窗体布局窗口中有一个模拟显示器，在它的“屏幕”上直观地显示了本程序中各窗体在实际显示器屏幕中的位置和大小。在模拟“屏幕”上，每个窗体都有自己的图标，用鼠标拖动某窗体图标，即可改变该窗体运行时的初始位置。



图 1.6 窗体布局窗口

1.3.8 代码窗口

代码窗口又称为代码编辑器，用于显示和编辑程序代码，如图 1.7 所示。在图 1.2 所示的 VB 集成开发环境中未显示代码窗口。以下几种方法均可以打开代码窗口：

- ① 在窗体窗口双击窗体内部或窗体中的控件；
- ② 在【视图】菜单中选择【代码窗口】命令；
- ③ 在工程窗口单击【查看代码】按钮；
- ④ 在窗体窗口内任意位置右击，在快捷菜单中选择【查看代码】命令；
- ⑤ 按 F7 功能键。

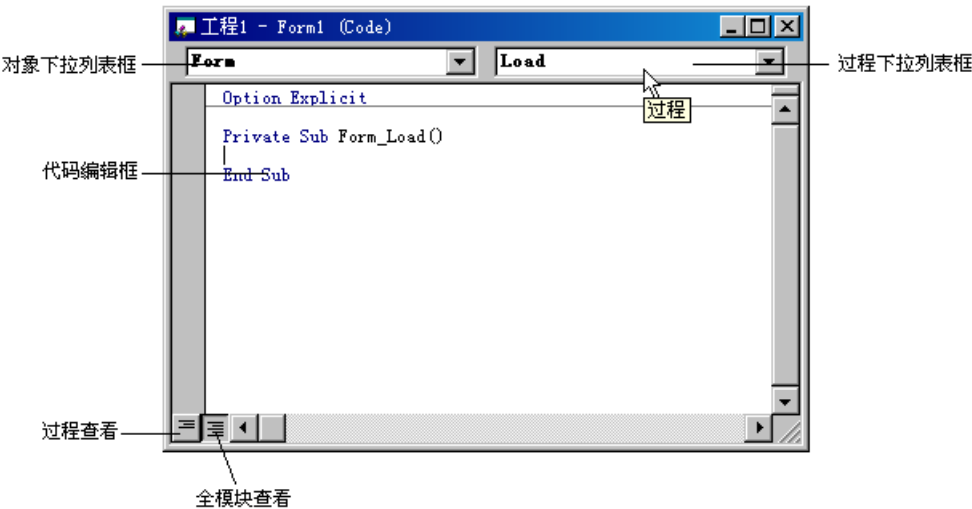


图 1.7 代码窗口

代码窗口主要由以下几部分组成：

对象下拉列表框：含有当前窗体及其所包含的全部对象的列表。单击右端的下拉按钮，可列出所有对象的名称以供选择。其中，窗体对象比较特殊，无论当前窗体的名称如何，在列表中均用 Form 表示窗体对象，而窗体中的控件则一律用控件名称来表示。此外，列表中的“（通用）”表示不属于特定对象的通用代码，一般在此声明模块级变量或用户编写的自定义过程。

过程下拉列表框：列出所选对象的所有事件过程名。当在对象列表框中选择不同对象时，过程列表框中的内容将发生相应的变化。其中“声明”表示模块级变量的声明。

代码编辑框：用于输入和编辑程序代码。

代码查看切换按钮：在代码窗口的左下角有两个按钮，【过程查看】按钮和【全模块查看】按钮，用于切换代码的查看范围。选择【过程查看】按钮时，代码编辑框中只显示选定的过程，所有的编辑操作只针对该过程；选择【全模块查看】按钮时，显示本窗体（模块）的全部代码。

1.3.9 立即窗口

立即窗口如图 1.8 所示。使用立即窗口可以在中断状态下监视对象属性、变量或表达式的值，也可以在设计时查询表达式的值或命令的执行结果。初学者可以在设计时利用立即窗口练习常用函数、语句和表达式的使用。例如，图 1.8 中的第 1~5、5、7、9 行是输入的语句，第 4、6、8、10 行是输出的结果。

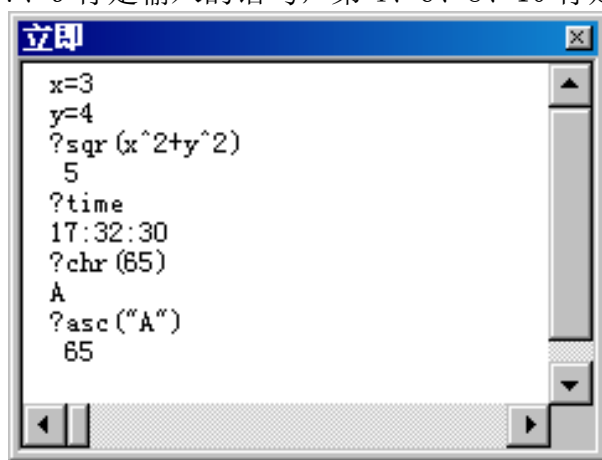


图 1.8 立即窗口

【立即】窗口、【本地】窗口和【监视】窗口三者合称为【调试】窗口，是 VB 集成开发环境所提供的程序调试工具的重要组成部分，打开这些窗口的菜单命令均位于【视图】菜单下。有关调试窗口在程序调试中的具体功能和使用方法将在第 9 章中讨论。

1.3.10 工具箱窗口

工具箱窗口如图 1.9 所示。当新建一个“标准 EXE”工程时，VB 将同时打开如图 1.9 所示的标准工具箱。标准工具箱中含有一个指针图标和 20 个内部（标准）控件的图标。除指针图标外，每一个图标代表一种控件，每个控件都是已经

定义好的对象，它们有自己的属性、方法和事件。将所需控件放在窗体中，就可以构建出实用的应用程序界面。

向窗体中添加控件的方法如下：首先单击工具箱中的控件图标，然后将鼠标指针移向窗体，此时鼠标指针形状变成十字状，按住左键从左上向右下拖动，松开鼠标左键时即可将控件添加到窗体中，矩形的大小代表了界面中控件的大小。也可以双击工具箱中的控件图标，此时控件按默认大小和位置被自动添加到窗体中。

图 1.9 中的控件为内部控件（又称为标准控件），这些控件不能从工具箱中删除。除内部控件外，VB 还提供了许多扩展控件（ActiveX 控件，文件扩展名为 .OCX），它们可以被添加到工具箱中，向工具箱中添加扩展控件方法如下：选择【工程】菜单中的【部件】命令，或者右击工具箱，在快捷菜单中选择【部件】命令，打开如图 1.10 所示的【部件】对话框，在【控件】选项卡的列表中，将所需控件前面的复选框选中（选定标志为“√”），单击【确定】按钮退出对话框后，被选中的控件即可添加到工具箱中。若要删除工具箱中的扩展控件，只需在上述操作中清除选定标志即可。



图 1.9 工具箱窗口

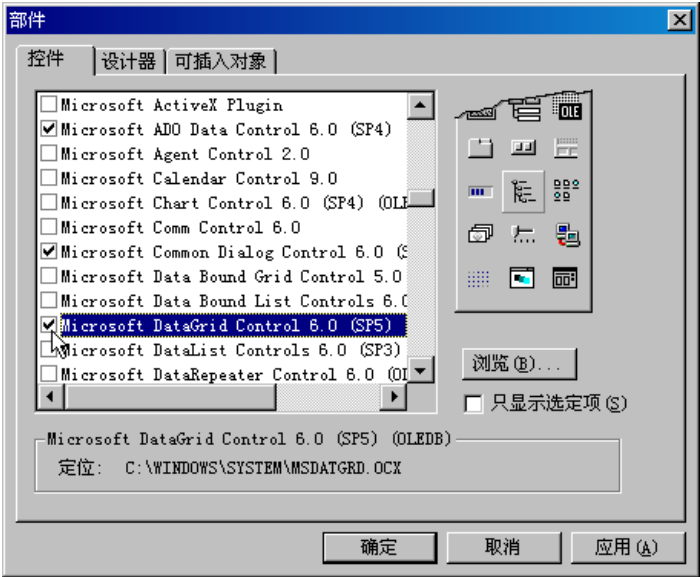


图 1.10 部件对话框

标准工具箱中各图标的含义及其功能

图标	控件名称	说明
	图片框	显示图片或文本。支持使用绘图方法绘图，也可以作为其他控件的容器
	标签	用于输出不希望用户改变的文本
	文本框	用于输入或者编辑文本
	命令按钮	是指定完成某种功能最简单的办法，也是最常用的控件
	框架	是一种容器型的控件，用来将窗体上的控件进行分类整理
	单选按钮	用于建立一系列的选项供用户选择，用户一次只能选择其中一个选项，并且必须选择其中的一个选项
	复选框	用于允许用户同时作多项选择，通常用 Value 属性测试是否被选中
	列表框	以列表的形式显示一系列的数据，用户从中选择一项，而不能直接输入或修改其中的内容
	组合框	同时具有文本框和列表框的特性，用户既可以从列表框中选择一项，也可以在其中的文本框中输入数据
	水平滚动条	提供水平方向的数值范围调整工具
	垂直滚动条	提供垂直方向的数值范围调整工具
	定时器	有规律地以一定的时间间隔激发时钟事件。此控件在运行时不可见
	驱动器列表框	用于列出该计算机所拥有的磁盘驱动器，供用户选择
	目录列表框	用于显示当前磁盘驱动器的目录结构，供用户选择
	文件列表框	用于显示当前目录下的文件清单，供用户选择
	形状	用于构造简单的形状，如圆形、矩形、三角形等
	直线	用于在窗体、图片框和框架等容器中绘制直线
	图像框	与图片框相似，用于显示图像，但不能绘图，也不能作为其他控件的容器
	数据控件	用于访问数据库。该控件不支持 ADO，现已少用
	对象链接与嵌入	用于把其他应用程序的数据嵌入到 Visual Basic 6.0 的应用程序中

1.4 怎样学好 Visual Basic

熟练掌握一种程序设计语言的语法是学好这门语言的第一步，所以掌握 Visual Basic 6.0 语法结构是初学者必须具备的基本功。本章以后各章将分别介绍 Visual Basic 6.0 的语法结构等基础知识。

掌握面向对象的程序设计（OOP）思想是学好 VB 的重要环节。传统的程序设计思想属于程序驱动的过程化程序设计，而 Visual Basic 6.0 的设计观念是事件驱动的面向对象程序设计，所以了解 Visual Basic 6.0 中的对象的概念、用途和功能，是 Visual Basic 6.0 进行程序设计的重要一环。VB 中对象具有属性、方法和事件，了解它们的概念和用途也是学习 VB 程序设计必不可少的重要环节。本书第 2 章将详细讨论对象、属性、方法和事件等概念，并将陆续介绍窗体和控件等对象的实际应用。

多看实例，多做练习。理论学习固然重要，但对于 VB 这种以开发应用为目的的程序设计语言来说，更重要的是掌握它的实际应用。多看一些典型的实例，尤其是教学实例和开发实例，并依照实例进行练习，可以迅速提高自己的编辑能力。当具有一定的编程基础后，可以尝试自行选择或设计应用课题，进行开发应用练习。

要有良好的代码编写能力和习惯，如在程序代码中的必要地方添加注释，采用缩进的代码编写风格。如果您能为一段较复杂的代码添加恰如其分的注释，说明您已经不是一个刚刚跨入 VB 殿堂的新手，您的编程能力已经上了一个新的台阶。此外，还有很重要的一点是对错误的处理，这是很多程序设计人员经常忽略的一个方面。

安装 MSDN (Microsoft Developer Network)，使用 Visual Basic 6.0 提供的帮助，这也是学好 VB 的捷径。

程序设计时要进行系统分析、软件需求分析、软件设计、界面设计、程序编码、测试、维护、编写文档等步骤，具体内容请参考软件工程等相关资料。

【练习】

1. Visual Basic 6.0 有哪些主要特点？
 2. 如何启动 Visual Basic 6.0？
 3. Visual Basic 6.0 集成开发环境中有哪些常用窗口？它们的主要功能是什么？
 4. 工程资源管理器和属性窗口各有哪些组成部分？它们的主要功能是什么？
 5. 如果集成开发环境中的某些窗口已被关闭，如何再将它们打开？
- 如何在工具箱中添加和删除扩展控件？

第 2 章 从零开始编制 VB 程序

本章要点

- 创建简单的应用程序
- 对象的属性、方法和事件
- 窗体的外观设计、文字显示、常用事件和方法以及多窗体程序
- 命令按钮的焦点概念、常用属性、方法和事件
- 标签的外观设计、常用事件和方法，为其他控件提供访问键
- 文本框的简单应用，多行、密码和只读文本框，使用选定的文本
- 开发 VB 应用程序的一般步骤

在第 1 章中，概括介绍了 Visual Basic 6.0 的特点、功能、集成环境以及学好 VB 的方法。从本章开始，将循序渐进地学习如何创建 VB 应用程序。本章主要介绍编制 VB 应用程序的方法、步骤以及窗体和几种基本控件的使用。

2.1 创建最简单的应用程序

要建立应用程序，首先要明确这个应用程序执行后窗体的外观和内容，如有哪些控件，控件的外观，控件之间的关系，对控件进行操作时将发生哪些事件等。本节通过简单的实例说明创建 VB 应用程序的一般步骤。

2.1.1 不编写代码的简单程序

【例 2.1】创建一个无须编写代码的简单程序，程序运行时显示“Hello, World!”。



(1) 创建工程

按 1.3 节所述的方法启动 Visual Basic 6.0，在【新建工程】对话框中选择【标准 EXE】，单击【打开】按钮，进入 Visual Basic 集成开发环境。。

(2) 设计界面

本程序通过标签控件显示文字。双击工具箱标签控件 (Label) 图标，在窗体上添加一个标签 (Label1)。若需调整控件的大小，可以拖动控件四周的八个大小方框（尺寸柄）。若需移动控件，可将鼠标指针指向控件内拖动。窗体大小的调整与控件相似，但只有右下角的三个实心尺寸柄可用。界面设计如图 2.1 所示。

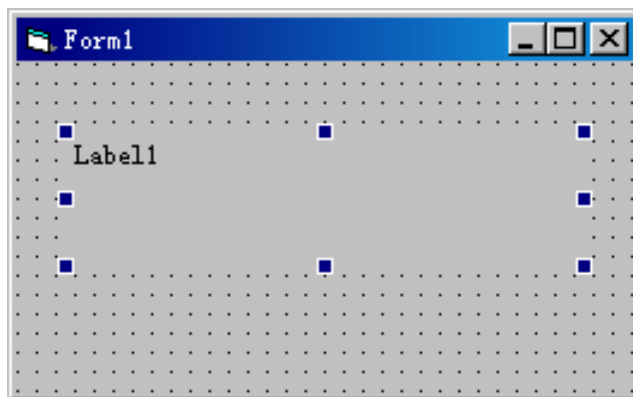


图 2.1 设计界面

(3) 设置属性

在 VB 中，窗体和控件统称为对象。对象建立好后，就要为其设置属性。属性是对象特征的表示，各类对象都有默认的属性值，设置对象的属性是为了使对象符合应用程序的需要。

设置标签属性：在窗体设计窗口选定标签，在属性窗口将 Caption 属性设置为“Hello, World! ”。单击 Font 属性右侧的  按钮，在如图 2.2 所示的【字体】对话框中将字体大小设置为二号。在窗体设计窗口调整标签控件的大小，使“Hello, World! ”显示为一行。通过【格式】菜单中的【在窗体中居中对齐】菜单命令将标签置于窗体中央。

设置窗体属性：单击窗体空白处将其选定，在属性窗口将窗体的 Caption 属性值改为“我的第一个程序”。

设置属性后的用户界面如图 2.3 所示。

图 2.2 【字体】对话框



图 2.3 设置属性

(4) 运行程序

单击工具栏中【运行】按钮 (▶) 或按 F5 键运行应用程序。程序运行结果如图 2.4 所示。



图 2.4 运行程序

(5) 保存工程

选择【文件】菜单中的【保存工程】菜单项或单击工具栏【保存】按钮，打开如图 2.5 所示的【文件另存为】对话框。系统首先要求保存的是所有窗体文件 (.frm)，最后才是工程文件 (.vbp)。VB 窗体的默认文件名默认是 FormX.frm (X 为序号)，工程的文件名是“工程 X.vbp”。建议在保存时采用直观的文件名，如本例将窗体文件命名为 frmHello.frm，工程文件为 Hello.vbp。

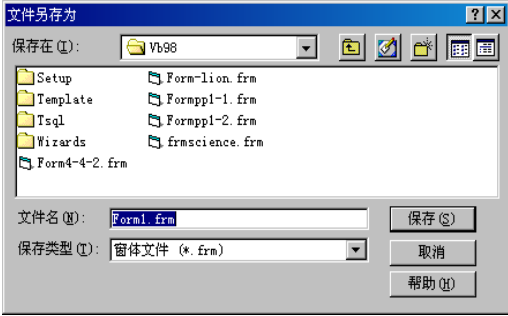


图 2.5 保存文件

至此，一个完整的小程序制作结束。整个过程看似复杂，实际上只需几分钟即可完成。

2.1.2 含有简单代码的程序

上面的程序过于简单，它没有提供与用户交互的功能。要想使应用程序能够响应用户的操作，就需要编写程序代码。在 VB 中，许多功能已封装在对象内部，例如，文本框本身应有各种文本编辑功能，文件列表框具有列出当前目录下的文件的功能。因此在 VB 中只需要编写少量的代码来满足某些特定功能的需求。

【例 2.2】编制一个含有简单代码的程序。程序界面和运行结果如图 2.6 所示。



图 2.6 例 2.2 运行结果

(1) 创建工程

启动 VB 时自动创建新工程的方法如前所述。如果已经进入 VB 集成开发环境，可执行【文件】|【新建工程】菜单命令，创建一个新工程。

(2) 设计界面

按照图 2.6 所示的界面，单击工具箱文本框控件图标，在窗体上画出一个文本框。选择工具箱命令按钮图标，在窗体上画出三个命令按钮。调整好各控件的大小和位置。

(3) 设置属性

窗体和各控件的属性设置如表 2.1 所示。

表 2.1 窗体和控件属性设置

对象 (名称)	属性	属性值
窗体 (Form1)	Caption	欢迎
命令按钮 (Command1)	Caption	显示
命令按钮 (Command2)	Caption	清除
命令按钮 (Command3)	Caption	结束
文本框 (Text1)	Text	
	Font (字号)	二号
	ForeColor	红色
	Alignment	2-Center

(4) 编写代码

在本例中和谐代码的任务是当用户单击【显示】、【清除】和【结束】三个命令按钮时分别做出响应。正如第 1 章所述，VB 采用事件驱动的工作方式，当用户单击某个按钮时，就发生该按钮的单击 (Click) 事件。我们要做的就是处理这一事件，即为事件过程编写代码。

① 为【显示】按钮的单击事件编写代码。在窗体设计窗口双击【显示】按钮，自动打开如图 2.7 所示的代码窗口，光标停留在该按钮的 Click 事件过程中。从图 2.7 中可以看出，在代码编辑区自动插入了按钮 Click 事件的“过程模板”：起始语句、一个空行和结束语句。因此，在编写代码时不必输入事件过程的起始和结束语句，只需编写要进行的操作即可。本例要求当用户单击【显示】按钮时，在文本框中显示“欢迎进入 VB 殿堂！”。在图 2.7 所示的过程模板空行处按 Tab 键（默认缩进 4 个空格），然后输入以下代码：

```
Text1.Text = "欢迎进入 VB 殿堂!"
```

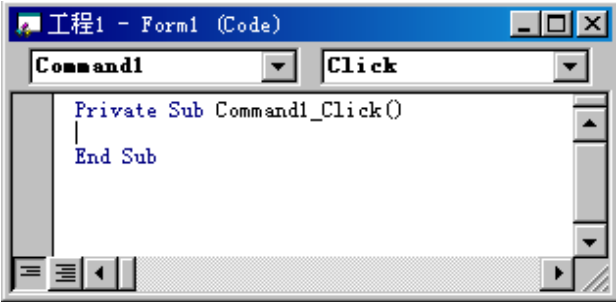


图 2.7 过程模板

输入代码后的结果如图 2.8 所示。

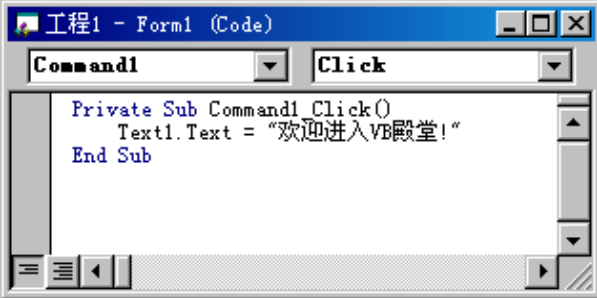


图 2.8 输入代码

② 为【清除】按钮的单击事件编写代码。在代码窗口单击对象列表框右端的下拉按钮，在图 2.9 所示的下拉列表框中选择 Command2, 系统自动插入该按钮单击事件的过程模板，在光标停留处按 Tab 键后输入以下代码：

```
Text1.Text = ""
```

图 2.9 选择编写代码的对象

③ 为【结束】按钮的单击事件编写代码。在对象下拉列表框中选择 Command3, 其单击事件添加以下代码（End 语句用于结束程序运行）：

```
End
```

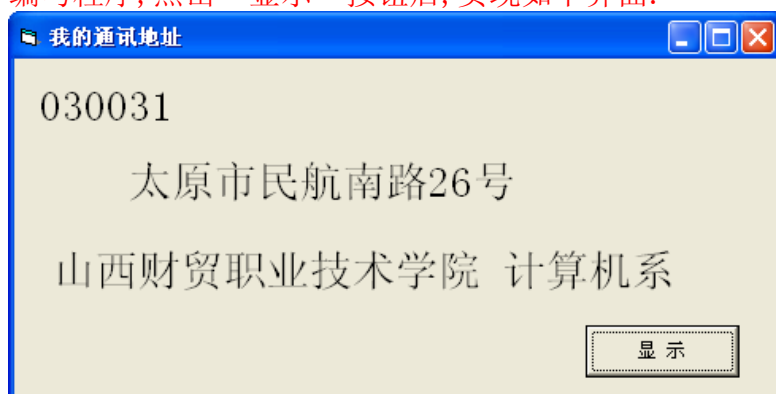
(5) 运行程序

单击工具栏中【运行】按钮（▶）或按 F5 键运行应用程序。运行时单击【显示】按钮，文本框中将显示“欢迎进入 VB 殿堂！”；单击【清除】按钮，将使文本框清空；单击【结束】按钮结束程序运行。

(6) 保存工程

在应用程序设计的任何阶段都可以保存本工程的所有文件，不必等到运行通过后才保存。对于较大的程序，应注意随时存盘，以免因突发事故导致前功尽弃。

【实践】：（请某位同学来做这个程序）
编写程序, 点击”显示”按钮后, 实现如下界面.



2.2 对象

2.2.1 对象是什么

对象 (Object) 的原意是指物体，它是现实世界中事物的抽象表示。对象在实际生活中随处可见。例如，一棵树，一个人，一台计算机，您正在看的这本书等等都是对象。在面向对象的程序设计 (Object Oriented Programming, OOP) 中，对象是具有属性和方法，且能对特定事件做出反应的实体，如窗体、文本框、命令按钮等都是对象。如窗体、文本框、命令按钮都是对象。对象是由代码和数据组合而成的封装体，可以作为一个整体来处理，即对象是数据和代码的集合。对象可以是应用程序的一部分，如控件或窗体，也可以是整个应用程序。对象有 3 个要素：属性、方法和事件。

对象是可以分类的。例如，轿车、卡车、面包车等各种各样的汽车属于汽车这个“类”，张三、李四等个人属于人这个“类”。类（Class）是同种对象的集合与抽象。对象是类的具体化，是类的实例，而类是创建对象实例的模板。对象一旦建立，即可改变其属性。例如，在例 2.2 中，我们使用了三个命令按钮对象，它们都是命令按钮类（Command Button）的一个实例，即用该类模板建立的一个对象。这些对象共享一组由类定义的通用的特征的功能（属性、方法和事件）。但是，每个对象都有它自己的名字，可以放在窗体的不同位置，可以有自己的事件过程等。在 1.3.6 节曾提到，在属性窗口（图 1.5）的对象下拉列表框中可以看到对象的名称及其所属的类，其中对象名称显示为粗体，对象所属的类显示为标准字体。

2.2.2 对象的属性、事件和方法

VB 是以对象为基础的程序设计语言。在 VB 中，窗体、控件等对象具有自己的属性和方法，能对特定事件做出反应。打个比方，有一个小伙子名叫张三，身高 1.80cm，某日在街上散步，走到十字路口时突然看到红色信号灯亮了，这时他停下来，等绿灯亮时再走。在这里，张三这个人就是一个对象，姓名、身高等特征是他的“属性”，走、看、停下来等动作（行为）是他的“方法”，“红灯亮了”是他能够响应的“事件”。下面分别介绍对象的属性、事件和方法。

1. 属性

属性是用来描述对象的数据，可看作静态特征，不同的对象具有不同的属性。通过对属性值的改变，可以使对象的状态发生变化。VB 中的对象都有许多属性，它们是用来描述和反映对象特性的参数，如控件的名称、标题、颜色、字体以及是否可见等。要想学好 VB，必须学会使用属性来定制窗体和窗体中的控件。但这并不是说必须掌握所有属性的用法。对于初学者来说，只需掌握几个最常用的基本属性。一般情况下，对于大多数属性，使用 VB 提供的默认值即可。随着编程经验的增加，可以根据应用程序的需要，逐步掌握某些特殊属性的应用。

（1）属性的设置

在 Visual Basic 6.0 中，对象属性的设置有两种方法：

- ① 在属性窗口直接设置。
- ② 在程序代码中通过赋值实现，格式如下：

[对象名.]属性 = 属性值

若对象是当前窗体，可省略对象名。例如：

```
Text1.Text=" Hello!"
```

’ 将文本框对象 Text1 的 Text 属性设置为 “Hello!”

```
Caption = “欢迎”
```

’ 将当前窗体的 Caption 属性设置为 “欢迎”

在上面的代码中，单引号（’）是注释引导符，它后面的内容是注释。

【演示】修改学生完成的练习题目来演示“属性设置”的两种方法。并提醒学生注意第二种方式是对窗体的属性设置。要求学生思考如何在点击“显示”按钮后，如何更把按钮的 Caption 修改为：“隐藏”。

（2）对象的命名

Name（名称）属性是所有对象都具有的属性，在属性窗口中它位于属性列表框的第一行，代表对象的名称。对象名称主要用于在程序代码中引用对象。在一

个窗体中，每个对象的名称必须保证惟一性，即不得有重名对象。在建立窗体和控件对象时，VB 会自动提供一个默认的名称，如 Form1、Text1、Command3 等，但这些名称未提供有关这个对象的更多信息。为了提高程序的可读性，在较复杂的程序中，对那些有可能在代码中被引用的对象，推荐采用能反映对象类型和功能的名称，如 frmHello、txtAge、cmdEnd 等。其中，前三个字母是 VB 约定的对象名称的前缀，代表对象类型（对象所属的类），用对象类型的缩写表示，紧跟其后的是描述对象功能的部分。对象名称最好采用大小写混合的拼写形式，前缀用小写，功能描述部分的首字母大写，如 cmdOK、txtUser 等。这样做有利于检查错误，且可提高可读性。常用对象类型及前缀见表 2.2。

表 2.2 常用对象类型及前缀

对象类型	前缀	对象类型	前缀	对象类型	前缀
Adodc (ADO 数据控件)	ado	FileListBox (文件列表框)	Fil	Menu (菜单)	menu
CheckBox (复选框)	chk	Form (窗体)	frm	OptionButton (单选按钮)	opt
ComboBox (组合框)	cbo	Frame (框架)	fra	PictureBox (图片框)	pic
CommandButton (命令按钮)	cmd	HScrollBar (水平滚动条)	hsb	RichTextBox (多格式文本)	rtf
CommonDialog (通用对话框)	dlg	Image (图像框)	img	Shape (形状)	shp
DataGrid (数据网格)	dgd	Label (标签)	lbl	TextBox (文本框)	txt
DirListBox (目录列表框)	dir	Line (直线)	lin	Timer (定时器)	tmr
DriveListBox (驱动器列表框)	drv	ListBox (列表框)	lst	VScrollBar (垂直滚动条)	vsb

对于那些在代码中根本不访问的对象（如仅用于显示静态文本的标签控件），则不妨仍采用系统默认名称。此外，修改对象名称最好在开始编写代码之前进行，否则将会给整个应用程序的设计和维

护带来困难。

2. 事件

事件是对象对外部变化的响应，如有人打 110，值班公安人员立即响应。事件是由用户或系统触发，是预先定义好的，可以由对象识别的操作。不同的对象所能识别的事件不同。例如，窗体能识别单击和双击事件，而命令按钮能识别单击却不能识别双击事件。这就像球迷们会为运动员加油，而不会为足球加油，因为不能识别“加油”这种事件。

当在对象上发生了某个事件时，如果要处理这个事件，就必须设计事件处理的步骤。事件处理的步骤称为事件过程（Event Procedure）。VB 程序设计的主要任务就是为对象编写事件过程中的程序代码。

在事件驱动的应用程序中，各事件的发生顺序是任意的，代码不是按照预定的路径执行，而是在响应不同的事件时执行不同的代码（即事件过程）。因此，编程人员只需对每一个对象的特定事件编写相应的代码即可，无须考虑程序的执行顺序。

事件过程的语法如下：

```
Private Sub 对象名_事件名 ([参数表])  
    ' 如果对象是窗体, 则一律用 Form_事件名  
    处理事件的代码  
End Sub
```

例如，在前面的例 2.2 中，单击命令按钮 Command2（“清除”按钮）时，将文本框 Text1 中的内容清空，对应的事件过程如下：

```
Private Sub Command2_Click()  
    Text1.Text = ""  
End Sub
```

3. 方法

方法是对象所具有的动作或功能。是可施加到对象上的各种操作，它是对象本身包含的函数或过程，用于完成某种特定功能，不同类型的对象拥有不同的方法集。例如，调用窗体的 Print 方法，可以在窗体上显示文字，调用窗体的 Move 方法，可以移动窗体的位置。许多方法可以改变对象本身的属性，如用 Move 方法移动窗体时，窗体的 Left 和 Top 属性值就会改变。从本质上讲，对象的方法是封装在对象内部的过程或函数。方法只能在代码中使用，它的用法取决于方法所需参数的个数以及方法是否具有返回值。如果方法不需要参数并且没有返回值，可用以下格式调用对象的方法：

[对象名.]方法名

若省略对象名，则默认为当前窗体。例如：

```
Form1.Show ' 显示窗体 Form1
```

如果方法需要参数，则用下面的格式调用对象的方法：

[对象名.]方法名 参数表

若有多个参数，需用逗号分隔。例如：

```
Print "欢迎!" ' 在当前窗体上显示文字  
' 以下语句将窗体 Form1 移动到屏幕左上角  
Form1.Move 0, 0
```

【实践】在学生完成的窗体中演示以上内容。

2.3 用户界面的载体——窗体

用 VB 创建应用程序的第一步就是创建用户界面。窗体是用户界面的载体，它就像一块画布，是所有控件的容量。编程人员可以根据自己的需要利用工具箱中的控件在“画布”上画出界面。本节讨论窗体的外观设计以及它的常用属性、事件和方法。

2.3.1 窗体的外观设计

窗体的外观是由窗体的属性决定的。在 VB 中，各种对象所包含的属性举不胜举，在这里只介绍常用的属性。从本节开始，我们将根据应用程序设计的需要，陆续介绍一些对象的常用属性，对于已经介绍过的属性，后续章节将不在重复。此外，对象的有些属性与特定事件或方法密切相关，我们将结合事件和方法介绍有关属性。

下面将介绍涉及窗体外观的常用属性，通过设置这些属性，可以使窗体的外观更符合特定应用程序的需要。

1. Caption 标题

Caption 属性用于返回或设置窗体标题栏上显示的文字。除窗体外，许多可视控件（如标签、命令按钮等）也具有 Caption 属性，决定控件表面显示的文字。

注意：该属性的默认值与对象的默认名称相同，正因为如此，初学者最容易将它与 Name（名称）属性混淆。

在代码中访问窗体的 Caption 属性（其他属性和方法与之相似）可用以下几种形式：

```
' 用窗体对象的名称访问其属性
Form1.Caption = "Hello"
' Me 关键字指当前窗体对象
Me.Caption = "Hello"
' 省略对象名称默认为访问当前窗体的属性
Caption = "Hello"
```

Me 关键字在编程时经常使用，它既可以简化代码，也可以提高程序的可读性。

【实践】通过对例 2.2 中窗体的 Caption 的设置，演示以上内容。

2. 背景色和前景色设置

BackColor 属性返回或设置窗体的背景颜色。ForeColor 返回或设置窗体的前景色，即显示在窗体中的文字和图形颜色。大部分可视控件也具有这两个属性。在属性窗口单击这两个属性右侧的下拉按钮，可选择一种颜色。此外，VB 提供了 8 个颜色常数，可在代码中直接用于颜色设置：vbBlack（黑色）、vbRed（红色）、vbGreen（绿色）、vbYellow（黄色）、vbBlue（蓝色）、vbMagenta（洋红）、vbCyan（青色）和 vbWhite（白色）。例如：

```
' 设置背景色为白色
Form1.BackColor = vbWhite
' 设前景色为蓝色
Me.ForeColor = vbBlue
```

【实践】修改例 2.2 中窗体的 BackColor 属性。

3. Left、Top、Height、Width 位置和大小

几乎所有可视控件都具有这几个属性。Left 和 Top 分别表示对象距容器左边界和顶边界的距离，它们决定了对象在容器中的位置。窗体的容器是屏幕，控件的容器通常为窗体，也可以是框架（Frame）、图片框（PictureBox）或选项卡（SSTab）控件。Height 和 Width 分别指定对象的高度和宽度。这四个属性的默认计量单位为缇（twip，1 厘米=567 缇）。

4. ControlBox、MaxButton、MinButton 边框元素

这三个属性决定是否出现窗体的边框元素，均为逻辑值。若 ControlBox 属性值为 True，则在标题栏左端显示控制菜单框（如图 2.10），运行时单击它可打开力中所示的控件菜单；若该属性值为 False，则无控件菜单框，同时隐藏最大化、最小化和关闭按钮。MaxButton 属性值为 True，显示最大化按钮，为 False 则不显示或为灰色（不可用）。MinButton 属性值为 True，显示最小化按钮，为 False 则不显示或为灰色。

5. BorderStyle 边框样式

该属性用于设置窗体边框的样式。属性值有：

0—None：无边框，无标题栏，无法移动及改变大小。

1—Fixed Single：单线边框，可移动，不能改变大小。

- 2—Sizable: 双线边框, 可以移动并可以改变大小, 这是默认值。
- 3—Fixed Dialog: 窗体为固定对话框, 不能改变大小。
- 4—Fixed ToolWindow: 窗体外观与工具箱相似, 有关闭按钮, 不能改变大小。
- 5—Sizable ToolWindow: 窗体外观与工具箱相似, 有关闭按钮, 能改变大小。

该属性在运行时只读。当 BorderStyle 设置为除 2 以外的值时, 系统自动将 MaxButton 和 MinButton 属性设置为 False。

6. Icon 图标

Icon 属性指定窗体处于最小化时显示的图标, 同时也是控制菜单框的图标。在属性窗口中, 可以单击 Icon 设置框右边的按钮, 打开【加载图标】对话框, 选择一个图标文件 (.ico) 载入。若不指定图标, 则使用 VB 默认图标。

7. Picture 背景图片

该属性用于设置窗体中要显示的背景图片。在属性窗口中, 可以单击 Picture 设置框右边的按钮, 打开【加载图片】对话框, 选择一个图形文件载入。也可以在程序代码中通过 LoadPicture 函数加载一个图形文件。

8. WindowState 窗口状态

该属性表示窗体在运行时以什么状态显示。属性值有:

- 0 - Normal: 正常窗口状态。
- 1 - Minimized: 最小化状态, 以图标方式显示。
- 2 - Maximized: 最大化状态, 无边框, 充满整个屏幕。

在代码中设置该属性时, 可以使用数值 0、1、2, 也可以使用 VB 常数 vbNormal、vbMinimized、vbMaximized。例如:

```
Me.WindowState = vbMinimized ' 使窗体最小化
```

```
Me.WindowState = 0 ' 使窗体恢复为正常状态
```

显然, 使用 VB 常数的代码可读性更强。

一般说来, 对象的大部分属性既可以通过属性窗口来设置, 也可以在程序代码中设置。而少数属性只能通过属性窗口设置, 或者只能通过编写程序代码来设置。还有少数属性在属性窗口和代码中均不能设置, 只能在代码中读取。

【实践】: 请某位学生通过对例 2.2 的修改, 然后运行程序, 演示以上内容。可以通过程序方式, 也可以直接在属性窗口中直接修改属性的方式。

2.3.2 在窗体上显示文字

应用程序经常专有权输出信息, 直接在窗体上显示文本就是输出信息的一种方式。

1. Print 方法的初步应用

调用窗体对象的 Print 方法可以在窗体上输出字符串。下面通过一个简单的实例说明如何在窗体上显示文字, 同时复习一下前面介绍过的几个涉及窗体外观的属性。

【例 2.3】 用 Print 方法显示窗体的当前位置。

(1) 设计界面及设置属性

在窗体上放置一个命令按钮 Command1, 将其 Caption 属性设为【改变属性值移动窗体】。将窗体的 MaxButton 属性设为 False (窗体最大化或最小化时,

若通过代码移动窗体位置将会出错)。窗体其他属性的设置: 设 Caption 为【在窗体上显示文字】, BackColor 为白色, ForeColor 为蓝色, Left 和 Top 均为 300; 单击 Font 右侧的按钮, 在【字体】对话框中设置字体为【黑体】, 字号 12。设计界面如图 2.11 所示。

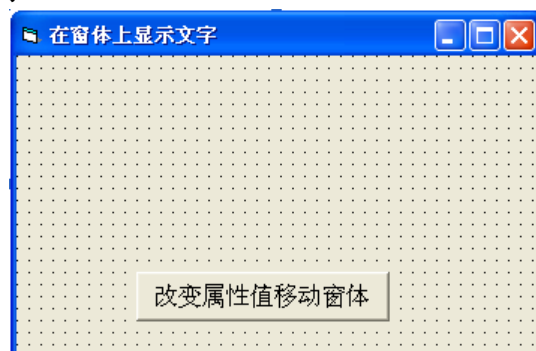


图 2.11 例 2.3 设计时界面

(2) 编写代码

程序代码的任务是单击命令按钮和窗体时改变或恢复窗体位置, 并显示窗体坐标。在 Command1 的 Click 事件中通过改变 Left 和 Top 属性, 使窗体右移、下移各 200 缇。在窗体的 Click 事件中通过改变 Left 和 Top 属性, 使窗体恢复原位。每次移动窗体以及窗体复位时, 用 Print 方法在窗体上显示窗体的当前坐标。

程序运行效果如图 2.12 所示。

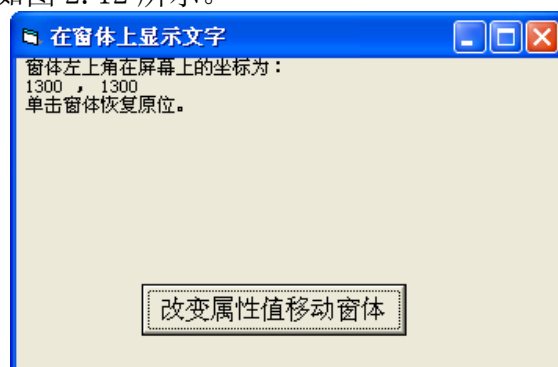


图 2.12 例 2.3 运行时界面

程序代码如下:

```
Private Sub Command1_Click() ' 命令按钮的单击事件
    ' 改变 Left 和 Top 属性值移动窗体
    Me.Left = Me.Left + 200
    Me.Top = Me.Top + 200
    Cls ' 清屏
    Print "窗体左上角在屏幕上的坐标为: "
    Print Me.Left; ", "; Me.Top
    Print "单击窗体恢复原位。"
End Sub

Private Sub Form_Click() ' 窗体的单击事件
    ' 恢复窗体初始位置
    Me.Left = 300
    Me.Top = 300
    Cls
```

```
Print " 窗体左上角在屏幕上的坐标为: "
```

```
Print Me.Left; ", "; Me.Top
```

```
End Sub
```

用 Print 方法在窗体上显示文字的一般语法格式为:

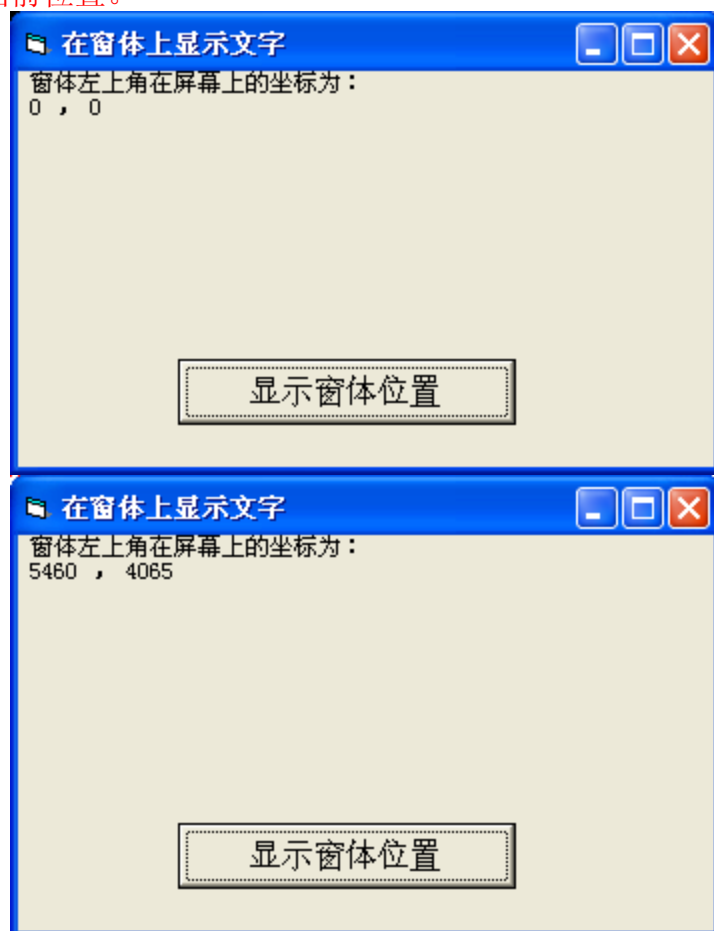
```
[对象名.]Print [输出项列表] [ ; | , ]
```

若省略对象名,则默认为当前窗体。**【输出项列表】**参数是显示在窗体上的文本。如果该参数有多个输出项,必须用分号或逗号分隔。其中分号表示各输出项连续输出,中间无空格;逗号表示各输出项按分区格式输出,每个分区宽度为14个字符。如果省略各参数,则输出一个空行。

Print 方法还有一些特殊用法,将在第4章详细介绍。

在例2.3的程序代码中还使用了Cls方法,该方法用于清除由Print方法生成的文本或绘图方法生成的图形。

【练习】:编写程序,点击“显示窗口位置”按钮后,用Print方法显示窗口在屏幕的位置,然后用鼠标移动窗口后再点击“显示窗口位置”按钮,显示窗口当前位置。



2. Font 属性与字体设置

在例2.1、2.2和2.3中均涉及到了窗体或控件的Font(字体)属性。在这三个示例中,都是通过属性窗口设置的,即单击Font右侧的按钮,在**【字体】**对话框中进行字体设置。在代码中设置字体属性与设置其他属性(如Caption属性等)有所不同。如前面的图2.2所示,要对话框中可以设置字体的多个项目,如字体名称、样式、字号和效果等。这些项目各有其对应的属性,可以在程序代码访问,包括:FontName(字体名称)、FontSize(字号)、FontBold(是否

粗体)、FontItalic (是否斜体)、FontStrikethru (是否加删除线) 和 FontUnderline (是否加下划线)。例如:

```
Me.FontName = "黑体"    ' 设当前窗体字体为黑体
Me.FontSize = 12        ' 字号为 12 磅, 1 磅=20 缇
Me.FontBold = True      ' 加粗
也可以使用另外一种形式, 即在 Font 后加圆点。例如:
Me.Font.Name = "隶书"
Text1.Font.Underline = True
```

说明: 这种形式实际上是在访问 Font 对象的某个属性, 暂时不必关心其细节, 会用即可。

【练习】: 把例 2.3 窗体的字体改为“隶书”并“加粗”。

2.3.3 窗体的加载和卸载

1. 窗体的加载

窗体的加载是指窗体及其所有控件被装入内存, 但界面尚未显示。窗体必须先经过加载阶段才能显示在屏幕上。窗体加载时发生 Load 事件。窗体一旦进入加载状态, Load 事件中的代码就开始执行。因此, 通常在 Load 事件过程中加入窗体的初始化处理代码, 如设置窗体和控件属性的初始值等。

【例 2.4】 在 Load 事件中通过代码为窗体和命令按钮的属性设置初始值, 实现与例 2.3 同样的功能。

新建一个工程, 在窗体上添加一个命令按钮。将窗体的 MaxButton 属性设为 False (该属性运行时只读), 其他属性均不作设置。

双击窗体打开代码窗口, 输入以下代码:

```
Private Sub Form_Load()    ' 设置窗体的属性
    Me.Caption = "在窗体上显示文字"
    Me.FontSize = 12
    Me.FontName = "黑体"
    Me.ForeColor = vbBlue
    Me.BackColor = vbWhite
    Me.Left = 300          ' 设置窗体位置的初始坐标
    Me.Top = 300
    Command1.Caption = "改变属性值移动窗体"
End Sub
```

按钮和窗体单击事件的代码与例 2.3 相同。程序运行效果与图 2.12 相同。

2. 窗体的卸载

窗体的卸载是指窗体被关闭而从屏幕上消失。用户单击窗体上的关闭按钮或在代码中执行 Unload 语句时, 即可卸载窗体。Unload 语句的语法如下:

Unload 对象

例如:

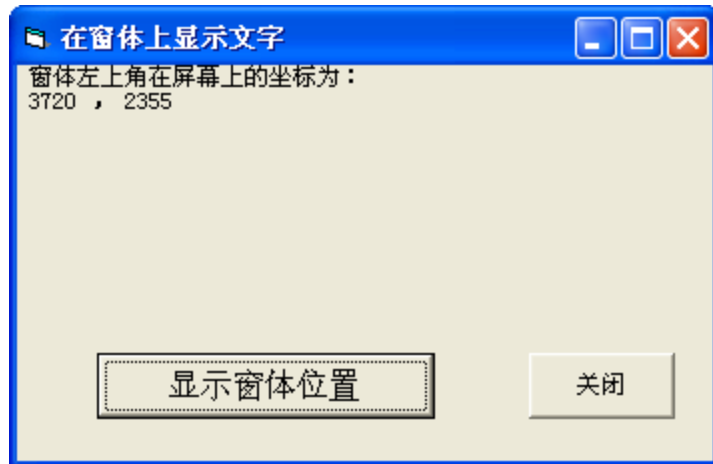
```
Unload Form1
Unload Me
```

窗体卸载前依次发生 QueryUnload 事件和 Unload 事件。这两个事件都有一个参数“Cancel”, 在事件过程中将该参数设为非零值可取消窗体的卸载。如果

需要在窗体卸载时进行一些善后处理（如保存数据或文件等），可以在这两个事件中提示用户，并做出相应的处理。注意不要将 Unload 语句和 Unload 事件混为一谈。

在例 2.2 的【结束】按钮的单击事件过程中，我们曾使用了 End 语句。End 语句直接结束应用程序的运行，不触发 QueryUnload 和 Unload 事件。

【练习】：在例 2.3 中，增加一个“关闭”按钮，点击该按钮时，关闭窗体。



2.3.4 窗体能识别的常用事件

1. 鼠标事件

- Click 事件：单击窗体的空白区域或一个无效控件时发生。
- DblClick 事件：双击窗体的空白区域或一个无效控件时发生。
- MouseDown 事件：当用户在对象上按下鼠标键时触发该事件。
- MouseUp 事件：当用户在对象上释放鼠标键时触发该事件
- MouseMove 事件：当用户在对象上移动鼠标时触发该事件。

2. Activate 和 Deactivate 事件

Activate 是窗体的激活事件，在窗体由非活动窗口变为活动窗口的瞬间发生。Deactivate 事件与 Activate 事件相对，在窗体由活动窗口变为非活动窗口的瞬间发生。

3. ReSize 事件

当窗体第一次显示或改变窗体的大小时发生该事件。利用该事件可以在改变窗体的大小时移动控件或调整其大小。

除上述事件外，窗体还有其他一些较常用的事件，如键盘事件、初始化事件等，将在后续章节中介绍。

2.3.5 窗体可以使用的常用方法

1. Cls 方法

Cls 方法用于清除运行时在窗体或图片框中显示的文本或图形。格式如下：

[对象.]Cls

其中：“对象”为窗体或图片框，若省略则默认为当前窗体。窗体中使用 Picture 属性设置的背景位图和放置在窗体上的控件不受 Cls 方法影响。

2. Move 方法

Move 方法用于移动窗体或控件，并可以改变其大小。格式如下：

[对象.]Move 左边距离[, 上边距离[, 宽度[, 高度]]]

其中：对象可以是窗体以及除菜单以外的所有可视控件，若省略对象则默认为当前窗体。

左边距离、上边距离、宽度、高度均为数值，以 twip 为单位。如果对象是窗体，则“左边距离”和“上边距离”以屏幕左边界和上边界为准，否则以窗体等容器内部的左边界和上边界为准。“宽度”和“高度”指定对象的新宽度和新高度。调用 Move 方法后将自动改变对象的 Left、Top、Width 和 Height 四个属性。

【例 2.5】用 Move 方法移动窗体。

在例 2.3 的窗体中添加一个命令按钮 Command2，设其 Caption 属性为【用 Move 方法移动窗体】，为该按钮的单击事件编写如下代码：

```
Private Sub Command2_Click() ' 使窗体向右、向下各移动 200 缇
    Me.Move Me.Left+200, Me.Top+200
    Cls
    Print " 窗体左上角在屏幕上的坐标为："
    Print Me.Left; ", "; Me.Top
    Print " 单击窗体恢复 Move 方法原位。"
End Sub
```

运行程序后会看到调用 Move 方法与改变属性值具有同样的效果

【例 2.6】使控件大小与窗体大小相适应。

新建工程，在窗体上添加一个文本框 Text1。为窗体的 Resize 事件过程编写如下代码：

```
Private Sub Form_Resize()
    Text1.Move 0, 0, Me.ScaleWidth, Me.ScaleHeight
End Sub
```

说明：当窗体大小改变时，触发 Resize 事件，在该事件过程中调用文本框控件的 Move 方法，使文本框始终充满整个窗体。在上述代码中，使用了窗体的两个特殊属性：ScaleWidth 和 ScaleHeight，它们分别代表窗体内部绘图区域的宽度和高度。

【思考】：如果把代码写入 Load 事件中，执行程序的时候与例 2.6 有什么区别？

2.3.6 多窗体应用程序

多窗体应用程序是指一个应用程序中有多个并列的普通窗体，每个窗体可以有自己的界面和程序代码，完成不同的功能。

1. 添加新窗体

执行【工程】菜单中的【添加窗体】命令或单击工具栏上的【添加窗体】按钮，打开如图 2.13 所示的【添加窗体】对话框，选择【新建】选项卡中的【窗体】图标，并单击【打开】按钮，即可在工程中新建一个空白窗体；若选择如图 2.14 所示的【现存】选项卡，则可以将一个已经做好的窗体添加到当前工程中。

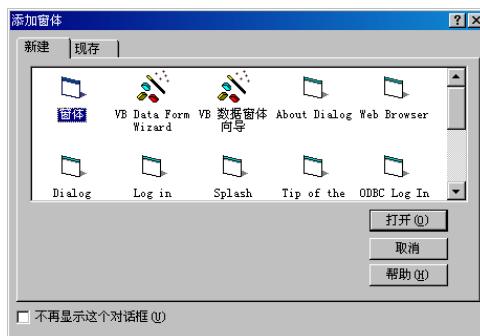


图 2.13 添加窗体对话框【新建】选项卡

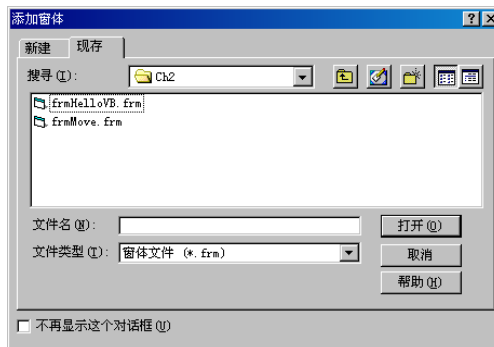


图 2.14 添加窗体对话框【现存】选项卡

添加一个已有（现存）窗体到当前工程时，有两个问题要注意：

- （1）被添加的窗体不能与工程内已有的任何窗体同名，否则无法 。
- （2）添加进来的现存窗体实际上是由多个工程共享，因此，对窗体所做的改变会影响到共享该窗体的所有工程。

2. 设置启动对象

在程序运行过程中，首先执行的对象被称为启动对象。在默认情况下，一个应用程序若含有多个窗体，则第一个创建的窗体被指定为启动对象，即启动窗体。如果要指定其他窗体为启动窗体，可执行【工程】菜单中的【工程属性】命令，打开如图 2.15 所示的【工程属性】对话框，在【启动对象】下拉列表框中选择所需窗体并单击【确定】按钮。

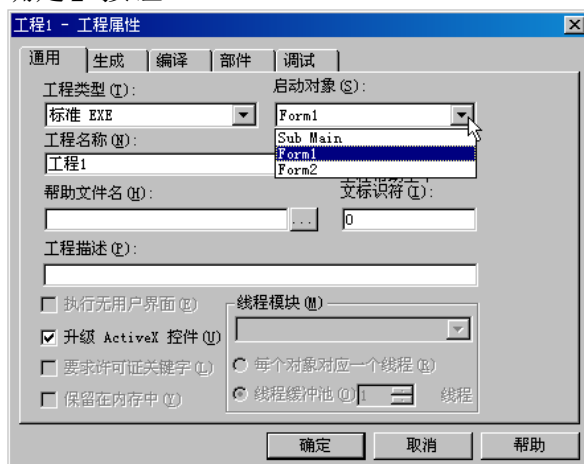


图 2.15 工程属性对话框

从图 2.15 中可以看出，启动对象既可以是窗体，也可以是 SubMain 过程。如果启动对象是 Sub Main 过程，则程序启动时，不加载任何窗体，以后由 Sub Main 过程根据不同的情况决定是否加载哪个窗体。有关 Sub Main 过程的详细内容将在第 8 章讨论。

3. 移除窗体

若当前工程中不再需要某个窗体，可将其移除（不删除磁盘文件）。移除的方法是先从工程资源管理器中选定要移除的窗体，然后执行【工程】菜单中的【移除 xxx】命令。

4. 窗体的显示与隐藏

调用窗体的 Show 方法和 Hide 方法，或者通过代码设置 Visible 属性，可以实现窗体的显示和隐藏。

(1) Show 方法

窗体的 Show 方法用于显示窗体。调用格式为：

[窗体名.]Show [模式 [, 拥有者]]

其中，“模式”参数有两种取值：0 (vbModalless, 默认值) 为非模式窗体，1 (vbModal) 表示模式窗体。模式窗体是指该窗体出现后，用户必须对其作出响应，在关闭该窗体前，不能对本程序中的其他窗体进行操作。非模式窗体则无此限制。“拥有者”参数用于指定被显示窗体的“父”窗体，当“父”窗体最小化或关闭时，被显示窗体也将最小化或关闭。通常将“拥有者”参数设为 Me。例如，在窗体 Form1 的单击事件过程中有如下语句：

Form2.Show vbModalless, Me

其中，Me 表示 Form1，即 Form1 是 Form2 的拥有者。

(2) Hide 方法

Hide 方法用于隐藏窗体。调用格式为：

[窗体名.]Hide

与 Unload 语句不同，Hide 方法只是将窗体暂时隐藏，并未卸载。

(3) Visible 属性

在设计时，通过属性窗口设置 Visible 属性用于指定窗体的可见性，True 为可见，False 为不可见。在代码中将该属性设为 True 与调用不带参数的 Show 方法效果相同，而将该属性设为 False 等同于调用 Hide 方法。

Visible 属性也是各种可视控件的公有属性。

【例 2.7】在程序启动窗体上单击【关于】按钮，通过另一个窗体显示版本信息。

新建工程，将窗体 Form1 改名为 frmMain，设其 Caption 属性为【主窗体】。在窗体上添加一个命令按钮，名称为 cmdAbout，设 Caption 属性为【关于】。添加一个新窗体，设名称为 frmAbout，Caption 属性为【关于】，BorderStyle 属性为 3；在 frmAbout 窗体上添加一个标签，设 Caption 为【多窗体示例 版本：1.0】；添加一个命令按钮，名称为 cmdOk，设 Caption 为【确定】。运行结果如图 2.16 所示。



图 2.16 多窗体程序

frmMain 窗体的【关于】按钮单击事件代码如下：

```
Private Sub cmdAbout_Click()  
    frmAbout.Show 1 ' 显示为模式窗体  
End Sub
```

frmAbout 窗体【确定】按钮单击事件代码如下：

```
Private Sub cmdOk_Click()  
    ' 若仅卸载本窗体则不应使用 End 语句，否则将结束程序  
    Unload Me  
End Sub
```

一个实用的应用程序大多含有多个窗体，本节只是对多窗体应用程序的初步介绍，在后面的章节中，将会介绍更多有关创建实用性多窗体程序的内容。

2.4 让用户下达命令的控件——命令按钮

命令按钮简称按钮。在应用程序中，命令按钮是最常用的控件，前面的示例中就多次使用了命令按钮。使用命令按钮时，通过在它的 Click 事件中编写一段程序，当用户单击这个按钮时，就会启动这段程序，执行某一特定的功能。

2.4.1 焦点

在介绍命令按钮之前，先来了解一下焦点的概念。“焦点”是可视化程序设计中频繁使用的一个术语。所谓焦点是指对象接收鼠标操作或键盘输入的能力。当对象具有焦点时，可以接收用户的输入。在 Windows 平台下，同一时刻只有一个窗口、窗体或控件具有这种能力。例如，在含有多个文本框的窗体上，只有具有焦点的文本框才能接收用户输入的文本。具有焦点的对象通常会以高亮显示的标题或标题栏来表示。

命令按钮可以通过 SetFocus 方法将焦点定位到自身。此外，在程序运行时，还可以使用 Tab 键使焦点在各个对象之间切换。具有焦点的命令按钮的标题周围具有虚线边框。

2.4.2 命令按钮的常用属性

1. Caption 属性与访问键

该属性设置显示在按钮上的文字（标题）。设置 Caption 属性时，如果某个字母前面加上“&”，则在程序运行时标题中的该字母即带有下划线，这一字母就成为快捷访问键。所谓快捷访问键是指与 ALT 键同时按下的键，用来打开菜单、执行命令或选择对象。当用户按下 Alt+快捷访问键时，其作用与通过鼠标单击该按钮相同。例如，在命令按钮  中，字母“O”就是快捷访问键，该按钮的 Caption 属性为【确定(&O)】，程序运行时按下 Alt+O 键即相当于单击了该按钮。上述设置访问键的方法也适用于其他具有 Caption 属性的控件。

2. Default 和 Cancel 属性

Default 属性用于设置窗体中的命令按钮是否为默认命令按钮，其值为 False 或 True。如果某个命令按钮的 Default 属性为 True，则在窗体启动后，按 Enter（回车）键就可以立即执行该命令按钮的功能。在同一窗体上只能有一个命令按钮的 Default 属性被设定为 True，当窗体中已经存在一个 Default 属性为 True 的按钮时，其他命令按钮的 Default 属性会自动设为 False。在实际应用中，通常将【确定】按钮的 Default 属性设为 True。

Cancel 属性用来设置窗体中某个命令按钮是否为【取消】按钮，其值为 True 或 False。程序运行后，按 Esc 键与单击活动窗体中 Cancel 属性为 True 的按钮所起的作用相同。同样，在同一窗体上只能有一个命令按钮的 Cancel 属性为 True。

3. Style 和 Picture 属性

Style 属性用于设置命令按钮的外观样式。设置值为 0-Standard（默认值）时，命令按钮为标准样式，不能在其中显示图形或设置背景颜色；设置值为 1-Graphical 时，命令按钮为图形样式，在按钮上可以显示图形或设置背景颜色。

Picture 属性可以指定一个图形文件，用来在命令按钮上显示该文件所对应的图形。要在命令按钮上显示图形，有效的前提是 Style 属性为 1。

4. Enabled 属性

设置命令按钮是否能被按下。当属性值为 True（默认）时，表示命令按钮可以接受用户鼠标或键盘输入来启动它；为 False 时表示按钮不能被按下，这时整个命令按钮以暗淡的颜色显示。在程序执行过程中可以通过修改该属性的值来设置用户的操作权限。

图 2.17 展示了按钮的几种外观。图中三个按钮的 Style 属性均为 1, Picture 属性各设置了一幅图片。按钮①的 Caption 属性为空，按钮②、③的 Caption 属性均为“大笑”，按钮③的 Enabled 属性为 False。



图 2.17 命令按钮示例

2.4.3 命令按钮的常用事件

命令按钮最常用的事件是 Click 事件，可以在该事件中编写代码来处理相应的任务。能触发 Click 事件的操作包括：单击命令按钮；焦点在按钮上时按回车或空格键；使用访问键；在代码中将按钮的 Value 属性设为 True。除 Click 事件外，命令按钮还能接受其他一些事件，但都不常用。

2.4.4 命令按钮的常用方法

命令按钮的常用方法是 SetFocus 方法，使用该方法可以将焦点定位在指定的命令按钮上。其格式为：

对象名.SetFocus

窗体和大多数可视控件也具有 SetFocus 方法。

【演示】修改例 2.1 中的按钮，演示以上内容。

2.5 最简单的文字显示控件——标签

标签控件的功能是显示文本，用户不能直接修改。标签可以用作标题、栏目名称、状态等说明注释文字。标签不能接受焦点。

2.5.1 标签的外观设计

标签的外观是由它的属性决定的。涉及标签外观的很多属性如字体、颜色、大小、位置、是否可见等与窗体的对应属性相同，在此介绍一下其他属性。

1. AutoSize 自动调整大小

该属性设置标签控件是否能自动调整大小以显示所有的内容(Caption 属性值)。如果设置为 True，则自动改变大小以适应内容；设为 False(默认)则保持设计时定义的大小，超出的部分不显示。

2. WordWrap 垂直调整

设置为 True 时，标签可以在垂直方向上改变大小以适应标题内容，但前提是必须设置 AutoSize 的值为 True；设置为 False(默认)时，标签不能在垂直方向上改变大小。

3. BorderStyle、BackStyle 边框和背景样式

BorderStyle 属性设置标签是否具有边框。0：无边框（默认值）；1：有边框。

BackStyle 属性设置背景样式。0：标签透明；1：标签不透明（默认值）。

4. Alignment 对齐方式

设置标签中标题文字的对齐方式。0：左对齐（默认值）；1：右对齐；2：居中。文本框等可视控件也具有此属性。

5. Appearance 立体外观

设置为 1（默认值）时标签为立体效果，设置为 0 时标签为平面效果。

图 2.18 展示了标签的几种外观。

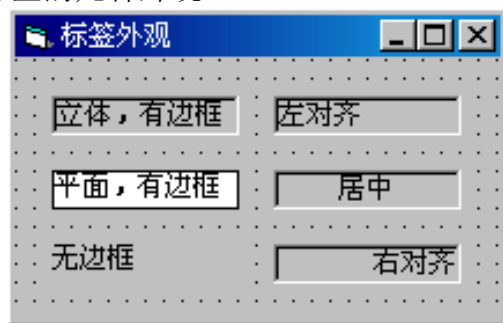


图 2.18 标签控件示例

【例 2.8】用标签制作浮雕效果文字。

利用两个标签控件，在设计时通过白色与黑色错位叠加，实现图 2.19 所示的文字浮雕效果。



图 2.19 浮雕效果文字

(1) 新建工程，在窗体上放置两个标签控件，设 Caption 属性均为【VB6.0 中文版】，字号均为【初号】，字体均为【隶书】。调整标签的宽度和高度，使文字如图 2.19 分两行显示。

(2) 按表 2.3 设置两个标签的其他属性。其中 Left 和 Top 属性是实现错位效果的关键。

表 2.3 标签控件属性设置

控件名	BorderStyle 属性	BackStyle 属性	ForeColor 属性	Left 属性	Top 属性
Label1	1 - Fixed Single	1 - Opaque	白色	324	324
Label2	0 - None	0 - Transparent	黑色	384	372

2.5.2 标签的事件和方法

标签可以接收 Click、Db1Click 和 Change 等事件。但是，在实际应用中，用户很少对标签进行操作，所以，标签的事件很少用到。

标签控件的常用方法有 Move、Refresh 等。Refresh 用于刷新标签的内容。

2.5.3 用标签创建访问键

在讨论用标签创建访问键之前，需要先了解一下对象的“Tab 键次序”和 TabIndex 属性。

1. “Tab 键次序”和 TabIndex 属性

当程序运行时，如果窗体上含有多个控件，则按 Tab 键可以使焦点在各控件之间切换，焦点切换的顺序就是“Tab 键次序”。“Tab 键次序”是由控件的 TabIndex 属性决定的，其值是以 0 为下界的整数。大多数可视控件具有 TabIndex 属性，设计界面时 VB 自动按控件被添加到窗体上的顺序分配 TabIndex 属性值（从 0 开始递增）。

2. 访问键

在 2.4 节我们已经学习了创建快捷访问键的方法。凡是具有 Caption 属性的控件，都可以为其设置快捷访问键。对于那些没有 Caption 属性的控件（如文本框、列表框等），可以将快捷访问键定义在标签的 Caption 属性中，利用标签为其设置快捷访问键。当用户按 ALT+访问键时，由于标签不能接受焦点，因此，焦点将按照“Tab 键次序”自动移动到下一控件处。

用标签为其他控件提供快捷访问键的具体做法是：首先要确保标签的 UseMnemonic 属性为 True(默认值)，然后在标签的 Caption 属性中指定访问键。添加控件时按先标签，后其他控件的顺序成对进行，或者以任意顺序绘制控件，最后将标签的 TabIndex 属性设置为其后控件的 TabIndex 值减 1。

提示：

要想使某个控件在窗体启动时首先获得焦点，可在设计时（或在窗体的 Load 事件中）将其 TabIndex 属性设置为 0。

2.6 最常用的字符输入输出控件——文本框

标签控件只能显示文本，而文本框控件既可以用于显示输出信息，也可以作为控件，在运行时接收用户输入的数据，它是应用程序与用户交互的常用工具。

2.6.1 文本框的简单应用

Text 属性是文本框最重要的属性，文本框中显示的文本即由该属性控制。可以用三种方法设置 Text 属性：设计时在属性窗口设置；运行时通过代码设置；运行时由用户输入。

文本框的 Font（字体）、Alignment（对齐）和 ForeColor（前景色）属性可用于设置文本框中字符的格式。这些属性的功能和用法已台前所述，不再重复。对文本框中字符格式的设置将会影响文本框中的全部内容，不能只对其中的部分内容设置格式。

【例 2.9】用文本框制作如图 2.20 所示的加法器。

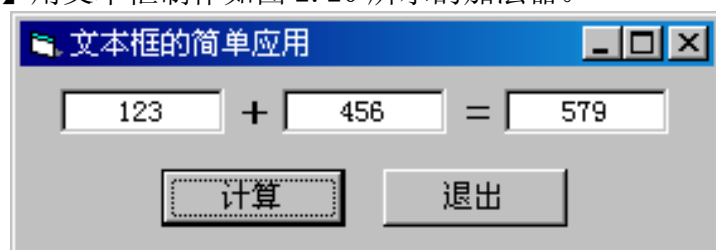


图 2.20 文本框的简单应用

新建工程，设窗体的 Caption 属性为【文本框的简单应用】。在窗体上添加三个文本框，名称分别为 txtAdd1、txtAdd2 和 txtResult，用于输入加数及显示结果。将三个文本框的 Text 属性均设为空，Alignment 属性均设为 2（文字居中）。添加两个标签，Caption 属性分别设为“+”和“=”。再添加两个命令按钮，名称分别为 cmdCalc 和 cmdEnd，Caption 属性分别设为【计算】和【退出】。

为【计算】按钮的单击事件编写如下代码：

```
Private Sub cmdCalc_Click()  
    txtResult.Text=Val(txtAdd1.Text)+ Val(txtAdd2.Text)  
End Sub
```

在【退出】按钮的单击事件中用 End 语句结束运行。

说明：Val 是 VB 内部函数，可将数字字符串转换为数字。详细内容将在第 3 章介绍。

【思考】：修改上例，分别完成加减乘除运算。

2.6.2 创建多行文本框

在默认情况下，文本框只显示单行文本，无滚动条，不支持回车换行。如果文本长度超过文本框的宽度，则只能显示部分内容，此时可用左右箭头键浏览文本。要想使文本框能够显示和输入多行文本，必须对有关属性进行设置。

1. MultiLine 属性

该属性只能在设计时设置。属性值设为 True 时为多行文本框，可以输入或显示多行文本，运行时可以按回车键换行。只要没有水平滚动条，则同时具有自动换行功能，即输入的文本超出文本框右边界时会自动换行。该属性设为 False 时为单行文本框。

2. ScrollBars 属性

设置文本框是否具有垂直或水平滚动条。0-None（默认）：没有滚动条；1-Horizontal：有水平滚动条；2-Vertical：有垂直滚动条；3-Both：同时有水平和垂直滚动条。该属性只能在设计时设置。当文本内容较多时，加入滚动条可便于浏览。

注意：该属性设置为 1、2 或 3 时有效的前提是 MultiLine 属性必须设置为 True。

图 2.21 展示了单行与多行文本框以及滚动条设置的效果。从图中可以看出，单行文本框不支持按回车键换行；在多行文本框中，只要有水平滚动条，就不能实现自动换行。

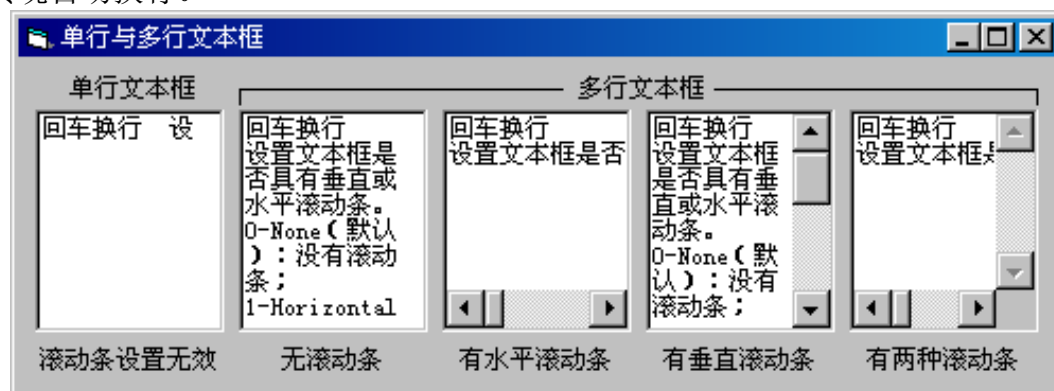


图 2.21 单行与多行文本框

3. 多行文本框中的 Text 属性设置

将文本框的 MultiLine 属性设为 True 后，在属性窗口中的 Text 属性设置与单行文本框有所不同，在 Text 属性右侧多了一个下拉按钮 ，属性值显示为“（文本）”。此时要设置 Text 属性，必须单击下拉按钮，弹出如图 2.22 所示的下拉列表框，然后在其中输入属性值。如果需要换行，不能直接按回车键，而应按 Ctrl+回车键，否则将结束输入。



图 2.22 设置 Text 属性

如果要在代码中设置 Text 属性，可按以下形式编写代码：

```
Text1.Text = "下江陵" & vbCrLf _
    & "朝辞白帝彩云间，" & vbCrLf ...
```

其中，vbCrLf 是 VB 常数，表示回车换行；&为字符串连接运算符；空格加下划线为续行符。

2.6.3 创建密码文本框

在实际应用中经常会看到类似于图 2.23 所示的程序登录界面，用户在密码文本框中输入密码时显示的是星号之类的点位符，而真正的密码被隐藏起来。利用 VB 提供的 PasswordChar 属性和 MaxLength 属性，可以很容易地创建密码文本框。

1. PasswordChar 属性

该属性指定显示在文本框中的字符。例如，若想在密码框中显示星号，则可在属性窗口或代码中将 PasswordChar 属性设为“*”。如图 2.23 所示，无论用户输入什么字符，文本框中都显示星号，但 Text 属性接收的仍是用户输入的实际文本。

注意：要想使该属性有效，必须将 MultiLine 属性设置为 False，即密码框必须是单行文本框。

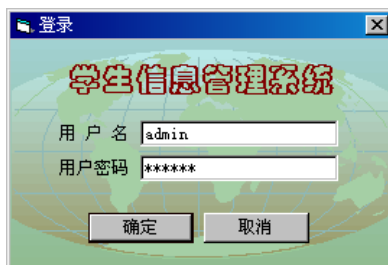


图 2.23 密码框

2. MaxLength 属性

该属性设置文本框中用户可以输入的字符串长度是否有受限制。默认值为 0，表示没有限制。若该属性被设置为大于 0 的整数，则表示文本框中允许输入

的最大字符数。该属性常与 PasswordChar 属性配合使用，限制用户输入密码的长度，也可以单独使用。

【思考】：把密码框与四则运算题目结合，如果密码正确，点击“确定”按钮后，打开四则运算窗体。

2.6.4 创建只读文本框

利用 Locked 属性可以禁止用户编辑文本框内容。该属性的默认值为 False，允许用户编辑文本框内容。若将 Locked 属性设置为 True，用户可以浏览和选定文本框中的文本，但不能作任何变更；可以在文本框中使用【复制】命令，但不能使用【剪切】和【粘贴】命令。Locked 属性只影响运行时的用户操作，在程序中仍可通过代码改变 Text 属性值。

2.6.5 使用选定的文本

在我们日常使用的各种文本编辑工具中，都可以选择全部或部分内容进行复制、剪切等操作，被选定的内容通常以黑（或蓝）底白字显示，称为反相（突出、高亮）显示。

文本框控件实际上是一个简单的文本编辑器，它提供了 SelStart、SelLength 和 SelText 三个特殊属性，可控制文本框中的插入点和文本选定操作。这些属性只能在运行时使用。此外，为了使文本框失去焦点时仍能反相显示选定内容，可使用 HideSelection 属性。

1. SelStart 属性

SelStart 属性是一个数字，用于指示选定文本块的起始位置。如果没有选定的文本，则该属性指示插入点的位置。若设置值为 0，则插入点被置于文本框中第一个字符之前。若设置值大于或等于文本框中文本的长度，则插入点被置于最后一个字符之后。

2. SelLength 属性

该属性用于指定所选文本块的字符个数。若将该属性设置为大于 0 的值 n，则选中并反相显示从当前插入点开始的 n 个字符。

技巧：若设 SelStart=0，SelLength≥文本框中的字符数，则可将文本框中的内容全部选定。

3. SelText 属性

SelText 属性是一个字符串，含有选定的文本。如果没有字符被选定，就是空字符串。对该属性赋值可以替换当前选中的文本，如果没有选中的文本，则在当前插入点处插入文本。

设置了 SelStart 和 SelLength 属性后，VB 会自动将选定的文本送入 SelText 中存放。

4. HideSelection 属性

该属性用于指定当控件失去焦点时选定的文本是否突出显示。True（默认值）：当控件失去焦点时，选定的文本不突出显示。False：当控件失去焦点时，选定的文本仍突出显示。该属性运行时只读。

【例 2.10】在文本框 Text1 中选定文本，通过命令按钮将选定的内容复制或剪切到文本框 Text2 中。

(1) 设计界面及设置属性

在窗体上放置两个文本框 Text1 和 Text2，将它们的 MultiLine 属性均设为 True，ScrollBars 属性均设为 2，HideSelection 属性均设为 False。将文本框 Text1 的 Alignment 属性设为 2。在属性窗口将文本框 Text1 的 Text 属性设为李白的诗“下江陵”。添加三个命令按钮，分别命名为 cmdCopy、cmdCut 和 cmdEnd，设 Caption 属性分别为【复制到目标区】、【剪切到目标区】和【退出】。用两个标签对文本框作简要说明。

(2) 编写代码

【复制到目标区】按钮单击事件的代码如下：

```
Private Sub cmdCopy_Click() ' 复制
    ' 将选中的文本复制到 Text2 中
    Text2.SelText = Text1.SelText
End Sub
```

【剪切到目标区】按钮单击事件的代码如下：

```
Private Sub cmdCut_Click() ' 剪切
    ' 将选中的文本复制到 Text2 中
    Text2.SelText = Text1.SelText
    ' 删除 Text1 中被选中的文本，完成剪切
    Text1.SelText = ""
End Sub
```

在【退出】按钮单击事件中用 End 语句结束运行。程序运行效果如图 2.24 所示。

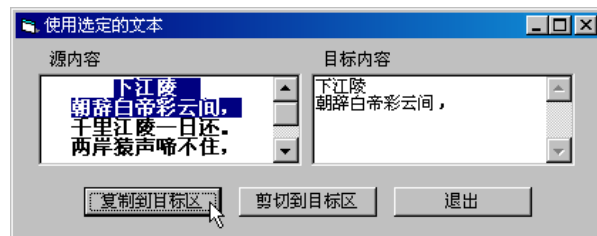


图 2.24 使用选定的文本

除了可利用上述属性处理选定文本外，文本框作为一个小型文本编辑器，不必编写任何代码，即可直接利用快捷菜单（在文本框中右击即可打开）或快捷键（Ctrl+C、Ctrl+X、Ctrl+V）完成对选定文本的复制、剪切和粘贴操作。

2.6.6 文本框的常用事件和方法

文本框支持的事件和方法较多，除了支持鼠标的 Click、blClick 事件外，还支持 Change、GotFocus、LostFocus、KeyPress 等事件和 etFocus 方法

1. Change 事件

当用户在文本框中编辑文本，或在程序中对文本框的 Text 属性进行更改时，将触发其 Change 事件。

2. GotFocus 事件

对象获得焦点事件。当运行程序时用 Tab 键或用鼠标选择对象，或用 SetFocus 方法使光标定位在对象上时，会触发该事件。其他可接收焦点的控件亦可识别此事件。

3. LostFocus 事件

对象失去焦点事件。当按下 Tab 键或用鼠标选择其他对象使光标离开当前对象时触发该事件。其他可接收焦点的控件亦可识别此事件。

4. KeyPress 事件

当焦点在当前文本框中时，在键盘上按下某个按键则触发该对象的 KeyPress 事件。该事件返回一个参数 KeyAscii，表示所按下的键的 ASCII 码。

与命令按钮一样，该方法是将光标移动到指定的文本框中，使其获得焦点。这是文本框比较常用的方法，当在窗体中建立了多个控件后，可以使用该方法把光标置于所需要的文本框中。其他可接收焦点的控件也具有此方法。

2.7 开发 VB 应用程序的一般步骤

在本章的前面几节中，我们通过若干例题介绍了如何开发一个 VB 程序，在此，对 VB 应用程序的开发步骤作一小结。

2.7.1 设计应用程序界面

应用程序界面是人机交互的接口，创建应用程序界面是 VB 程序设计的第一步，可分为创建工程、添加控件和调整控件等步骤。

1. 创建工程

创建工程可以在启动 VB 时进行，如果已经进入 VB 集成开发环境，可执行【文件】菜单中的【新建工程】命令，创建一个新工程。新工程的默认名称为“工程 1”，VB 自动为新工程添加一个默认名称为 Form1 的窗体。一个工程中可以有多个窗体，其中有一个为启动窗体。

2. 添加控件

根据应用程序功能的需要，通过工具箱中的按钮在窗体上添加控件。可以单击工具箱中的相应控件图标后在窗体上画出控件，也可以直接双击工具箱中的控件图标进行添加。

3. 调整控件

为了使程序界面更加美观实用，往往需要对窗体和控件的大小及位置进行调整。

(1) 选定控件

要选定单个控件时单击它即可。若要同时选定多个控件，可以采用两种方法：

①拖动鼠标，将欲选定的控件包围在一个矩形虚框内。

②选定第一个对象，按住 Ctrl 键，再依次单击其他要选定的控件。

选定多个对象后，可以整体移动并可统一设置其格式和某些属性，以减少重复操作。在一级被选定的控件中，最后被选定的是主控件，它周围为实心尺寸柄，而其他被选定的控件周围为空心尺寸柄。进行格式设置时，将以主控件为标准。若要改变主控件，可在选定多个控件后，再单击其中的某个控件既可将其设置为主控件。

(2) 复制与删除控件

- 复制：选定控件，单击工具栏【复制】按钮，再单击【粘贴】按钮，系统将弹出对话框询问是否创建控件数组的对话框，单击【否】按钮则复制出名称不同而其他属性均相同的控件。有关控件数组的详细内容将在第 7 章讨论。

- 删除：选定控件，按 Del 键即可。

(3) 调整控件的大小

选定对象，拖动尺寸柄，调整到所需大小为止。

(4) 统一尺寸

当希望窗体上多个控件具有相同大小时，可以选择【格式】菜单中的【统一尺寸】命令来实现。根据需要选择子菜单项【宽度相同】、【高度相同】或者【两者都相同】。

(5) 对齐控件

当希望窗体上多个控件整齐排列时，可以选择【格式】菜单中的【对齐】命令来实现。对齐方式有：左、居中、右、顶端、中间、底端对齐以及对齐到网格。

(6) 锁定控件

为了防止在设计应用程序的过程中不小心移动调整好的控件，可将调整好的控件锁定。选择【格式】菜单中的【锁定控件】命令，此时窗体上的所有控件都被锁定，不能在窗体上用鼠标直接拖动控件。需要解锁时，重复执行上述菜单命令即可。

(7) 其他格式

利用【格式】菜单还可以对选定控件设置间距、在窗体上的居中方式和前后层次等。

2.7.2 设置属性

应用程序的用户界面设计好后，即可通过属性窗口来设置对象的属性。在属性窗口中设置的属性值只是初始设置，在程序中可以对这些属性值进行修改。

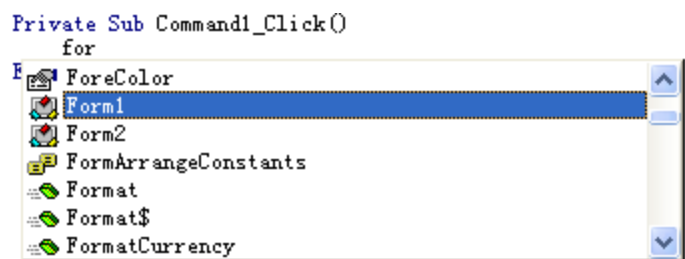
2.7.3 编写代码

编写代码是开发 VB 程序最关键的步骤，其中主要是编写事件过程代码，实现应用程序的特定功能。下面介绍一下利用“自动完成关键字”功能快速输入代码的技巧。

VB 代码编辑器具有自动完成关键字的功能。若某些关键字或对象名称较长，或者忘记了它们的完整拼写形式，只记得其前几个字母时，利用“自动完成关键字”的功能即可快速、准确地输入关键字或对象名称。有两种操作方法：

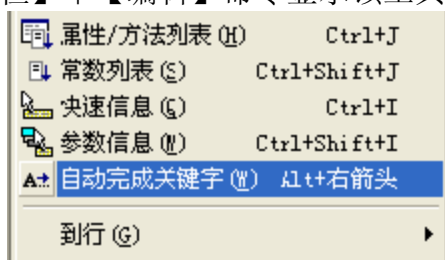
(1) 用快捷键

先输入关键字或对象名称的前几个字符，然后按 Alt+→ 键，此时在插入点处将会出现如图 2.26 所示的快速列表，用 ↓ 或 ↑ 键选中所需关键字或对象名称，然后按 Tab 键或其他分隔符（如空格、圆点“.”、逗号、等号、非字母运算符等），即可准确无误地输入该关键字或对象名称。双击快速列表中的关键字或对象名亦可完成输入。



(2) 用工具栏按钮

先输入关键字或对象名称的前几个字符，然后单击【编辑】工具栏按钮（图 2.27），亦可打开图 2.26 所示的快速列表。若【编辑】工具栏未显示，可通过选择【视图】|【工具栏】|【编辑】命令显示该工具栏。



例如，某命令按钮的名称为 cmdQuestion，要将其 Endabled 属性设置为 False，可按以下步骤操作：

输入“cmdq”，按 Alt+→ 键，在快速列表中选中该按钮，输入句点“.”，输入“e”，输入赋值号“=”，按 Tab 键结束。此时将自动生成以下代码：

```
cmdQuestion.Endabled=False
```

2.7.4 调试运行程序

完成代码编辑后，即可按 F5 键或单击工具栏【运行】按钮运行程序。当程序有误出现错误提示对话框后，不要急于关闭它，应弄清楚对话框中给出的错误类型的性质后再关闭。可以通过【调试】菜单的各种调试选项来设计程序，尽可能发现程序中存在的错误和问题。有关程序调试的详细内容将在第 9 章讨论。如果程序出现死循环，无法结束运行时，可按 Ctrl+Break 键中断程序，然后单击工具栏中的结束按钮结束程序运行，并修改错误。

2.7.5 保存工程及生成可执行文件

为了避免因突发事故导致前功尽弃，应注意随时保存工程中的所有文件。VB 工程中的常见文件类型主要有工程文件（.vbp）、窗体文件（.frm）、二进制窗体文件（.frx，主文件名与窗体文件同名，多用于存放图像等信息）和标准模块文件（.bas）等。

程序调试运行通过后可以将其生成可执行文件（.exe），后者可以在操作系统中直接运行，不必启动 VB 集成开发环境。生成可执行文件的方法是：选择【文件】菜单中的【生成…EXE】命令，打开【生成工程】对话框，选择文件存放位置并指定文件名后，单击【确定】按钮。

1. 什么是对象？举例说明对象的属性、事件和方法。

2. 对象的 Name（名称）属性与 Caption 属性有何区别？
3. 如何在窗体上显示文字？如何清除窗体上的文字？在多窗体程序中如何设置启动窗体？
4. 怎样为命令按钮设置访问键？怎样为文本框设置访问键？
5. 要在命令按钮上添加图片应当对什么属性进行设置？若已在规定的属性里装入某个图形文件，但按钮仍不能显示图形，应如何修改？
6. 文本框和标签的主要区别是什么？
7. 用标签制作阴影文字。
8. 如何将文本框设置成多行文本框使其显示垂直滚动条。
9. 制作一个密码文本框，输入密码时只显示 # 号，密码的长度不得超过 16 个字符。
10. 简述开发 VB 应用程序的一般步骤。

第 3 章 VB 语言基础

本章要点

- Visual Basic6.0 的数据类型
- Visual Basic6.0 的变量和常量，变量的命名规则和声明方法
- Visual Basic6.0 的运算符及表达式的用法
- Visual Basic6.0 的常用内部函数的用法
- Visual Basic6.0 的编码规则

在第 2 章中，介绍了编制 VB 应用程序的方法、步骤及几种基本控件的使用，可利用这些控件快速地编写简单的小程序。但要编写实用的程序，还要学习 Visual Basic 程序设计语言，包括基本语句、函数、过程等。本章主要介绍 VB 的数据类型、表达式、编码规则等语言基础知识。

3.1 数据类型

在 Visual Basic6.0 中，为解决各类实际问题，要采用各种不同的数据类型。不同的数据类型所表达的数据范围、精度、所占的存储空间和可以进行的运算均不相同。Visual Basic6.0 不但提供了丰富的标准数据类型，还可以由用户自己定义数据类型。

3.1.1 标准数据类型

Visual Basic6.0 所提供的标准数据类型有数值型、逻辑型、日期型、字符型、可变型、对象型等几种。

1. 数值型数据（Numeric）

数值型数据可分为两大类：整型和实型。

(1) 整型

整型表示的就是整数，整数运算速度快、精确，但表示数的范围小。根据所表示的数的范围不同，又可以分为三种类型：字节型 (Byte, 1 字节)、整型 (Integer, 2 字节)、长整型 (Long, 4 字节)。

字节型 (Byte)：为无符号整数。在计算机内用一个字节表示，其取值范围为 0~255。

整型 (Integer)：指在计算机内用两个字节来存储的整数，其取值范围为 -32768~+32767。

长整型 (Long, 4 字节)：指在计算机内用四个字节来存储的整数，其取值范围为 -2147483648~+2147483647。

在 Visual Basic 6.0 中也可以用八进制和十六进制表示整数。为了与十进制有所区别，八进制数必须在其数字前冠以符号 "&" 或者 "&O"；十六进制数必须在其数字前冠以符号 "&H"。而且，在用八进制和十六进制表示长整型数据时，必须在数字结尾加以符号 "&"。例如：&147、&O147、&147& 分别是八进制整数、八进制整数和八进制长整型数；&H147、&H852F7& 分别是十六进制整数和十六进制长整型数。

十进制：默认。

八进制：前加 &O；例：&O123

十六进制：前加 &H；例：&HAB123

(2) 实型

实型表示的就是实数，实数采用浮点表示形式，表示数的范围大，但有误差，且运行速度慢。根据所表示的数的范围和精度的不同，又可以分为三种类型：单精度实型 (Single)、双精度实型 (Double)、货币型 (Currency)。

单精度实型 (Single)：在计算机内用四个字节表示的实数，其取值范围为 $\pm 1.401298E-45 \sim \pm 3.402823E38$ ，最多有 7 位有效数字。当要赋给单精度变量的有效数字超过 7 位时，超出部分会自动四舍五入。如果该数的整数部分超过了 7 位，该数将用科学记数法表示。形式如下： $\pm aE \pm c$ ，表示 $\pm a \times 10^{\pm c}$ 。其中 a 表示该数的尾数部分，可以是整数或实数；E 表示该数是单精度实数，也可以用 e 表示；c 表示该数的指数部分，只能是整数。例如：12345.678 表示为单精度实数应该为 12345.68。12345678 表示为单精度实数应该为 1.234568E7，即 1.234568×10^7 。

双精度实型 (Double)：在计算机内用八个字节表示的实数，其取值范围为 $\pm 4.94065645841247DE-324 \sim \pm 1.79769313486232D308$ ，最多有 15 位有效数字。当要赋给双精度变量的有效数字超过 15 位或者该数的整数部分超过了 15 位时，处理方法与单精度数相同，超出部分会自动四舍五入。需要注意：双精度实数科学记数法与单精度数有所不同，尾数与指数之间用 D 或者 d 来间隔，形如： $\pm aD \pm c$ ，表示 $\pm a \times 10^{\pm c}$ 。

货币型 (Currency)：是一种特殊的实数，它是专为处理货币而设计的数据类型。它采用定点表示形式，在计算机内一般用八个字节来表示，小数点的右边保留 4 位，小数点的左边最多可以达到 15 位，其取值范围为 -922337203685477.5808~922337203685477.5807。

2. 逻辑型数据 (Boolean)

逻辑型数据 (Boolean) 只有两个值：True (真) 和 False (假)。逻辑型数据在计算机内用两个字节保存。可以把它们转换成数值型数据，此时，True

为-1, False 为 0。也可以把其他类型的数据（数值型或由数字组成的字符串型数据）转换为逻辑型数据，此时非 0 的数据转换为 True, 0 转换为 False。

转换：

逻辑 $\Rightarrow$ 整型：True $\Rightarrow$ -1; False $\Rightarrow$ 0;

其他 $\Rightarrow$ 逻辑：非 0 $\Rightarrow$ True; 0 $\Rightarrow$ False

3. 日期型数据 (Date)

日期型数据 (Date) 在计算机内一般用八个字节的浮点数来表示，其取值范围为：日期：公元 100 年 1 月 1 日~9999 年 12 月 31 日；时间：0:00:00~23:59:59。数值型的数据也可以转换为日期型数据。此时，该数据的整数部分表示从 1899 年 12 月 30 日起经过的天数，负数表示在其之前的天数；小数部分表示时间，午夜为.0，中夜为.5。

表示：

① 两端用#将日期文字括起来；例如：#January 19, 2002#；#10/12/2002#；#2001-12-31 12:30:00PM#

② 用数字序列表示（小数点左侧为日期，右侧为时间，.0 为午夜，0.5 为中午 12 点；负数表示 1899 年 12 月 30 日前）。例如：-3.0 表示 1899 年 12 月 27 日 00:00:00；1.5 表示 1899 年 12 月 31 日 12:00:00。

4. 字符串型数据 (String)

字符串型数据 (String) 是用双引号括起来的一串字符组合，引号内的字符可以是字母、各种符号和汉字。Visual Basic6.0 中，字符串分为两种类型：定长字符串和变长字符串。

(1) 定长字符串

事先定义字符串的长度（即字符串内所含字符的个数），在和谐运行过程中，始终保持其长度为不变的字符串。例如：假设定义变量 Str1 为长度为 8 的定长字符串，当给 Str1 赋值为 Very 时，Str1 的值为 Very，用四个空格添满不足部分。当给 Str1 赋值为 Very Good 时，Str1 的值为 Very Goo，右边的多出的字符被截去。Str1 始终保持其长度为 8 不变。

(2) 变长字符串

字符串的长度不固定，随着对字符串变量赋值，它的长度可以发生变化。例如：假设定义变量 Str2 为长度为变定长字符串，当给 Str2 赋值为 Very 时，Str2 的值为 Very，其长度为 4。当给 Str2 赋值为 Very Good 时，Str2 的值为 Very Good，其长度为 9。

5. 变体型数据 (Variant)

变体型数据 (Variant) 是一种特殊的数据类型，也称为可变型数据。一个变体型的变量能够存储所有系统定义类型的数据，可以随着为它所赋值的类型而改变自身类型。系统默认的数据类型是变体型。代表各种类型（定长 String 和用户自定义型除外），可用 VarType 函数检测其类型，可自动转换。

变体型数据 有三个特殊的值，分别为

(1) Empty：还没有为赋值。它不同于数值 0、长度为 0 的字符串” ” 和空值 Null，后三者都是有特定的值的。

(2) Null：通常用于数据库应用程序，表示未知数据或丢失的数据。

(3) Error：是特定值，指出已发生的过程中的错误状态。

6. 对象型数据 (Object)

对象型数据（Object）作为 32 位地址来存储，该地址可以引用当前应用程序或其他应用程序中的对象。可以用 Set 语句指定一个被声明为 Object 的变量，去引用应用程序所能识别的任何实际对象。对象型在 Visual Basic6.0 的较高层次的编程中使用。

Visual Basic6.0 的基本数据类型总结见表 3.1。

表 3.1 Visual Basic6.0 的基本数据类型

数据类型	关键字	类型符	字节数	取值范围
字节型	Byte		1	0~255
整型	Integer	%	2	-32768~32767
长整型	Long	&	4	-2147483648~2147483647
单精度实型	Single	!	4	$\pm 1.40\text{E}-45 \sim \pm 3.40\text{E}38$
双精度实型	Double	#	8	$\pm 4.94\text{D}-324 \sim \pm 1.80\text{D}308$
货币型	Currency	@	8	-922337203685477.5808~922337203685477.5807
字符串型	String	\$	不确定	定长 0~65535 个字符，变长 0~2 ³¹ -1 个字符
逻辑型	Boolean		2	True 和 False
日期型	Date		8	日期从 100 年 1 月 1 日到 9999 年 12 月 31 日 时间从 00:00:00 到 23:59:59
可变型	Variant		待分配	
对象型	Object		4	任何对象引用

3.1.2 自定义数据类型

自定义数据类型是由用户自己定义的数据类型，它由若干标准数据类型组成。自定义类型数据也称记录类型数据。自定义数据类型通过 Type 语句来实现。定义语句格式为：

```
[Private|Public] Type 类型名
    元素名 As 数据类型
    . . .
End Type
```

说明：Private|Public 是两个可选项，它们表示自定义的数据类型的作用范围，如果省略，默认为 Public。若在窗体模块中声明自定义类型，必须用 Private。将在后面的章节中详细讲解。

例如：

```
Private Type Student
    Name As String
    Birth As Date
    Age As Integer
End Type
```

定义了一个名为 Student 的数据类型，该类型有三个元素 Name、Birth 和 Age，它们分别为字符串类型、日期型和整型。定义了 Student 类型之后，就可以定义该类型的变量。

理解助记：

数据类型就像衣服的型号，不同型号的衣服有不同的尺寸大小，只适合某种身材的人穿。小个头穿大号衣服，只是浪费国家的布匹，不会挤破衣服，但大个头穿小号衣服，会挤破衣服，所以大个头决不能穿小号衣服。买衣服时，只要记住适合自己的衣服型号，就可以很方便地买到所需的衣服，但小朋友不知道自己的衣服型号，需要大人帮他们去选择。数据就像小朋友，需要你根据它的特点，选择适合它的自己的衣服型号，即数据类型。在选择时，将较小的数存放在较大的空间中没什么问题，只是浪费部分存储空间，但较大的数绝不能用较小的空间来存放。

3.2 变量与常量

在程序的运行过程中，有些值是固定不变的，有些值是可以发生变化的。例如：在求圆面积的程序中，有语句 $s=3.14*r*r$ ，圆半径 r 是可以变化的，圆面积 s 随着圆半径 r 的改变而改变，也是可以变化的。而 3.14 是固定不变的。因此，从另外一个角度，可以将数据分为变量和常量。

3.2.1 变量

在程序的运行过程中，其值是可以发生变化的量称为变量（从另一个角度讲，变量是存放数据的内存单元的名称）。每一个变量都有名字和数据类型。变量名用来唯一地标志每一个变量，数据类型则表明了该变量的类型。实际上，变量就是命名的内存存储单元，它可以暂时保存程序处理过程中的数据，如输入的数据、参加运算的数据、运算结果等。变量一般要先声明，再使用。

一个变量的值可以随时改变，因此一个变量中能存放多个值，但一个变量在任一时刻只能存放一个值，当给变量赋新值时，会替换原来的旧值，即变量“喜新厌旧”。只要变量存在，同时没有赋新值给变量，变量的值一直保存，可以多次读取。可以说，变量值“取之不尽”，而且每个变量均有自己的数据类型和唯一的变量名。

理解助记：

变量就像旅馆里的客房，每个房间都有自己的房间号和配置标准，它专给旅客临时居住，虽然房间里的住客经常在变，但任何时候只能住一批人。旅馆里的客房与变量的不同点是，只有客房内所住的人退房后，才能住下一批人，不会因为下一批人来而将前一批人下赶走，不像变量那样“喜新厌旧”。一般常量就像人的住所，比较固定，不用记房间号；符号常量就像旅馆的包间，虽然也是旅馆的一间客房，且也有记房间号，但它的住客不会变，包间需要先登记，符号常量需要先声明。

1. 变量的声明

(1) 显式声明

格式如下：

```
Public | Dim | Static | Private 变量名 As 数据类型[, 变量名 As 数据类型...]
```

或者：

Public | Dim | Static | Private 变量名<类型符>[, 变量名<类型符>...]

说明:

(1) 变量声明后, 系统会自动赋初值, 数值型变量系统置初值为 0, 字符串变量系统置初值为空串。

(2) As 数据类型 可省, 若省略数据, 则声明的变量为变体型变量。

(3) 如果要在同一个声明语句中声明多个变量, 则各个变量间用逗号隔开, 且每个变量均要单独声明数据类型。

如 Dim sum As Single, ave As Single 与 Dim sum, ave As Single 完全不同。前者声明了两个单精度变量 sum 和 ave, 而后者声明了一个变体型变量 sum 和一个单精度变量 ave。

例如:

```
Dim A As Integer, B As Long
```

语句声明了两个变量 A 和 B, 其中 A 为整型, B 为长整型。

```
Dim C!
```

语句声明了一个变量 C, 类型为单精度型。

```
Dim myname As String '声明变长字符串变量 myname
```

```
Dim no As String*8 '声明定长(8 个字符)字符串变量 no
```

声明变量后即可为其赋值, 格式为:

变量名 = 表达式

(2) 隐式声明

在 Visual Basic6.0 中, 允许对变量进行隐式声明, 即不对变量进行声明而直接引用, 此变量将被默认为 Variant 数据类型, 且没有确定的初值。

例如, 有如下事件过程:

```
Private Sub Command1_Click()
```

```
B = -12.3
```

```
Print B
```

```
End Sub
```

上述事件过程中, 变量 B 没有声明就使用, 可以认为它就是隐式声明的变量。但是, 这样容易因为写错变量名而引起麻烦。例如: 在应用程序中声明了变量 Student, 但是在后面的程序段中误写为 Studnt, 系统会认为是有一个隐性声明的变量 Student, 从而造成错误。为了避免这种错误, 将变量的声明方式设置为强制声明, 主要有两种方法来实现: 一种方法是可以在模块的声明中加入语句“Option Explicit”, 强制编译器发现所有未声明的变量。另一种方法是选择【工具】—【选择】命令, 弹出“选项”对话框, 单击【编辑器】标签, 选中“要求声明变量”复选框。如图 3.4 所示, 在代码窗口中会自动出现“Option Explicit”。例如, 下面的程序段在执行时将产生图 3.1 所示的错误提示:

```
Option Explicit
```

```
Private Sub Form_Click()
```

```
A = 25.8 '隐性声明变量 A, 在这里不允许
```

```
Print A
```

```
End Sub
```



图 3.1 错误提示

理解助记：

声明变量相当于在旅馆里开房，旅客入住时，旅店要根据旅客的特点和要求，确定旅客的房间号（变量名称）和房间类型（变量的数据类型）。

2. 变量的命名规则

在 Visual Basic6.0 中变量的命名要遵循以下的规则：

- （1）变量名要以字母或汉字（建议少用）开头，不能以数字或下划线开头。
- （2）变量名一般由字母、数字、汉字和下划线组成，不得含有+、-、*、/、\$、&、%、!、#、?、小数点或逗号等字符。
- （3）变量名的长度不得超过 255 个字符。
- （4）变量名不得与 Visual Basic6.0 中的关键字重名，如 Type、Public、Sin 等。）），然后根据人的年龄大小，确定是购买大人衣服还是小孩衣服（整型还是实型）最后根据人的身材来确定具体的衣服大小（具体数据类型）。变量常用的数据类型的选择方法如下图所示。

3. 变量的作用范围和存活期

变量的作用范围又称作用域。根据变量声明的位置和声明符的不同，VB 将变量分为过程级变量（或称局部变量）、模块级变量和全局变量，如表 3.2 所示。

表 3.2 变量的作用范围

作用范围	声明位置	声明符	可见范围
过程级	事件过程、通过程或函数过程	Dim、Static	在声明变量的过程或函数中
模块级	窗体/模块的“通用-声明”部分	Dim、Private	在声明变量的整个窗体或模块中
全局	模块的“通用-声明”部分	Public	整个工程

除作用范围外，变量还有存活期（或称有效期），在存活期内变量能够保持它们的值。全局变量在应用程序运行期间始终有效，窗体的模块级变量在窗体卸载前一直有效。用 Dim 关键字声明的局部变量仅当过程被执行时才存在。当一个过程执行完毕，它的局部变量的值就已经不存在，而且变量所占据的内存也被释放。当下一次执行该过程时，它的所有局部变量将重新初始化。如果要保留局部变量的值，可在过程内部用 Static 关键字将局部变量声明为静态变量。

【例 2.1】使用静态变量统计鼠标单击窗体的次数。步骤如下：

（1）新建一个标准 EXE 工程。

（2）双击窗体，打开代码编辑器窗口。在对象列表中选择“Form”，在过程列表中选择“Click”。在“Form_Click（）”过程模板中添加语句形成如下代码段：

```
Private Sub Form_Click()
```

```

Static countDanji As Integer ' 声明静态变量
countDanji = countDanji + 1
Print "已单击的次数为："; countDanji
End Sub

```

程序运行结果如图 3.2 所示。

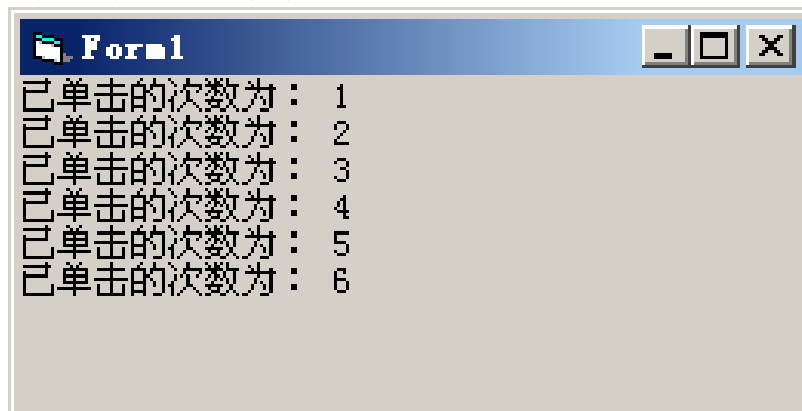


图 3.2 例 2.1 运行结果

3.2.2 常量

常量就是在程序运行过程中，其值不会发生变化的量。Visual Basic6.0 中，常量有两种：普通常量和符号常量。

1. 普通常量

普通常量可以分为数值常量、字符串常量、逻辑常量和日期常量等。

数值常量：即数学中的常数。例如：0，-7.368，&369，45E-6 等都是数值常量。

字符串常量：用双引号括起来的字符序列。例如：“ ”，“How Are You”，“山东教育学院”等都是字符串常量。

逻辑常量：只有两个值 True（真）和 False（假）。

日期常量：用于表示某一具体的日期和时间。可以有多种表示形式，但必须把日期和时间用符号#括起来。例如：#2004/5/3#，#May 5, 2004#，#04-5-5 15:12:40#等都是日期常量。

2. 符号常量

符号常量是用一个符号来表示一个固定不变的量，它有两种来源：用户自定义和系统内部定义。

用户自定义符号常量的方法如下：

```
Const 符号常量名 [As 数据类型] = 表达式
```

用[]括起来的部分表示是可选的。表达式的值就是符号常量所表示的值，一旦定义了一个符号为常量，就不能再为它赋值。例如：

```
Const PI As Single = 3.14159 ' 定义 PI 为符号常量, As Single
```

表示定义符号常量为单精度类型，可以省略以后要使用 3.14159 的地方，可以用 PI 来代替。

在 VB 6.0 中，已经预定义了大量的符号常量，这些符号常量一般以小写的 vb 为前缀，如 vbRed 表示红色，vbGreen 表示绿色，vbNormal 表示正常，vbCRLF

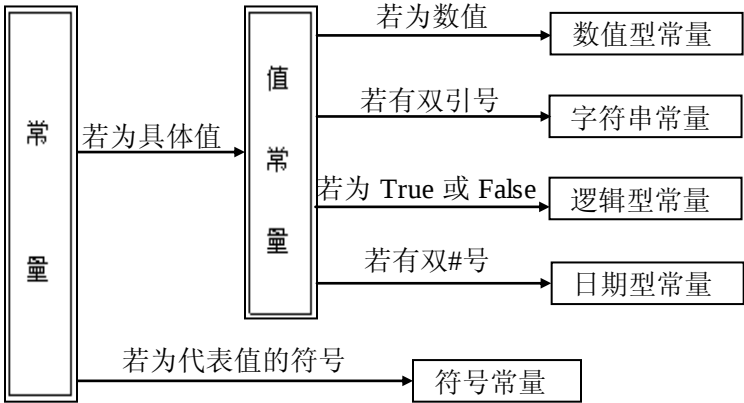
表示回车换行。系统内部定义的符号常量是由控件或应用程序提供的，在【对象浏览器】窗口可以查到它们。

例如如下事件过程：

```
Private Sub Form_Click() ' 单击窗体使之最大化
    Form1.WindowState = vbMaximized
End Sub
Private Sub Form_DblClick() ' 双击窗体
    Form1.WindowState = vbMinimized ' 最小化
End Sub
```

过程中的VbMaximized（将窗口最大化）、vbMinimized（将窗口最小化）就是系统内部定义的符号常量。

常量的分类及其特征如下图所示：



3.3 运算符与表达式

和其他程序设计语言一样，VB 也具有丰富的运算符，来形成各种表达式，实现各种运算。

3.3.1 运算符

1. 算术运算符

算术运算符包括：+（加）、-（减）、*（乘）、/（除）、\（整除）、Mod（取余）、-（负号）、^（指数）。算术运算符进行简单的算术运算，运算对象是数值型数据。

+（加）、-（减）、*（乘）、/（除）和-（负号）与数学中的运算规则相同，但是要注意在表达式中*（乘）号不能省略，且注意它的写法。如：求圆周长的公式必须写为 $2*3.14*r$ 的形式，而不能将其中的乘号省略。

VB6.0 中有两种特殊的除法运算：\（整除）和 Mod（取余）。整除运算就是对两数进行除法运算后取商的整数部分。取余运算就是对两数进行除法运算后取商的余数部分。若参与运算的两个数都是整数，则可直接进行运算。若参与运算的两个数中有实数，则先将实数的小数部分进行四舍五入，再进行运算。例如： $6.8 \backslash 3$ 的结果为 2， $6.8 \text{ Mod } 3$ 的结果为 1。

^（指数）对数据进行指数运算。例如： 3^2 的结果为 9， $8^{(1/3)}$ 的结果为 2。若参与运算的两个数都是

2. 关系运算符

关系运算符又称比较运算符。关系运算对两个进行比较，运算的结果是逻辑型的值。当关系成立时，结果为 True；当关系不成立时，结果为 False。

Visual Basic6.0 中的关系运算符如表 3.3 所示，其中前六个运算符较为常用，它们的运算对象是数值型数据或字符型数据。若参与运算的两个数都是数值型数据，则按照数值的大小来比较。。若参与运算的两个数都是字符型数据，按照字符的 ASCII 码值的大小从左到右逐个字符进行比较，直到碰到第一个不相同的字符为止。ASCII 码值大的字符串大。例如：“abxd”<“abcd”的结果为 False。

表 3.3 VB 6.0 关系运算符

运算符	意义	示例	返回值
=	等于	"ABC"="ABF"	False
>	大于	"ABC">"AF"	False
>=	大于等于	"f" >= "Fgh"	True
<	小于	25<45.5	True
<=	小于等于	23<=23	True
<>	不等于	"XYZ"<>"xyz"	True
Like	使用通配符匹配比较	"WXYZ" Like "*X*"	True
Is	引用对象比较	object1 Is object2	由对象引用的当前值决定

Like 运算符用于字符串的模糊比较，可以与通配符“*”、“？”、“#”等结合使用。“*”可匹配多个任意字符，“？”可匹配单个任意字符，“#”可匹配单个数字字符。

Is 运算符用于比较两个的引用变量。

3. 逻辑运算符

逻辑运算符包括 And（逻辑与）、Or（逻辑或）、Not（逻辑非）、Xor（异或）、Eqv（逻辑等价）、。逻辑运算符是对操作数进行逻辑运算，运算的结果为逻辑型数据。当逻辑关系成立时，运算结果为 True；当逻辑关系不成立时，运算结果为 False。

（1）And（逻辑与）

当且仅当参与运算的两个操作数都为 True 时，运算结果为 True。否则，只要参与运算的两个操作数中有任何一个为 False 时，运算结果都为 False。例如： $(3>2) \text{ And } (4=4)$ 的结果为 True， $(2>5) \text{ And } (6<7)$ 和 $(2>5) \text{ And } (6>7)$ 的运算结果均为 False。

（2）Or（逻辑或）

当且仅当参与运算的两个操作数都为 False 时，运算结果为 False。否则，只要参与运算的两个操作数中有任何一个为 True 时，运算结果都为 True。例如：，

(2>5)Or(6>7) 的运算结果为 False, (3>2)Or(4=4) 和 (2>5)Or(6<7)的结果均为 True。

(3) Not (逻辑非)

Not 是单目运算符。当参与运算的操作数为 False 时, 运算结果为 True。当参与运算的操作数为 True 时, 运算结果为 False。例如: Not (2=2) 运算结果为 False, Not (2>3) 运算结果为 True。

(4) Xor (异或)

当且仅当参与运算的两个操作数相异时, 即其中一个为 True, 另一个为 False, 运算结果为 True。否则, 当参与运算的两个操作数都相同时, 即它们都为 True 或者它们都为 False, 运算结果都为 False。例如: (2>5)Xor(6<7)的结果为 True, (3>2)Xor(4=4)的运算结果为 False。

(5) Eqv (逻辑等价)

当且仅当参与运算的两个操作数都相同时, 即它们都为 True 或者它们都为 False, 运算结果为 True。否则, 当且仅当参与运算的两个操作数相同异时, 即其中一个为 True, 另一个为 False, 运算结果为 False。例如: (2>5) Eqv (6<7)的结果为 False, (3>2) Eqv(4=4)的运算结果均为 True。

(6) Imp (蕴含)

当且仅当参与运算的两个操作数中, 第一个数为 True, 第二个操作数为 False 时, 运算结果为 False。否则, 运算结果为 True。例如: (2<3) Imp (2=2), (2>3) Imp (2<>2)的结果为 True, (2<3) Imp (2<>2))的运算结果为 False。

运算符	NOT 非	AND 与	OR 或	XOR 异或	EQV 等价	IMP 蕴含
优先级	1	2	3	3	4	5

4. 字符串运算符

字符串运算符包含“&”、“+”两个运算符, 它们的作用是将两个操作数连接起来, 成为一个字符串, 操作对象可以是字符串、数值型和可变型数据。

其中“+”是直接将两个字符串从左到右原样连接, 生成一个新的字符串。参与运算的两个数据必须是字符型数据。例如” This is a “+” book.” 的运算结果为一个新的字符串” This is a book.”。当两个操作数均为数值型时, 相当于算术加; 当数字字符和数值型连接时, “+”将字符型转换为数字型, 然后相加; 当非数字字符+数值型(类型不匹配)时, 出错。

“&”是将参与运算的两个数据强制性地按字符串类型连接在一起, 生成一个新的字符串。参与运算的两个数据可以是字符型、数值型和可变型数据。例如: ” Visual Basic” &6.0 的运算结果为一个新的字符串” Visual Basic 6.0”。

注意: 若在变量后使用运算符“&”时, 应在变量和“&”之间加一个空格。因为“&”既是字符串运算符, 又是长整型类型符, 当变量名和“&”连在一起时, Visual Basic 6.0 会把它作为类型符号处理。

3.3.2 表达式

1. 表达式

由常量、变量、函数、运算符以及括号连接起来的有意义的式子称为表达式。
要注意的是：常用的一些数学表达式要写为 Visual Basic 6.0 的表达式形式。
例如：数学表达式 $38ab+lnx$ 的 VB 表达式为：

```
38 * a * b + Log(x)
```

2. 类型转换

在表达式中经常会出现不同类型数据混合运算的情形。此时，需要按一定的规则进行类型转换。转换的方法有两种：系统自动转换和使用转换函数转换。

对于算术运算，当参与运算的数据具有不同的精度时，系统将自动进行类型转换。系统规定运算结果的数据类型以精度高的类型为准。

转换函数将在后面的章节中学习。

3. 运算符的优先级

当表达式中有多个运算符时，此时表达式要按运算符的优先级来进行运算。在 Visual Basic 6.0 的表达式中，运算按照括号、函数、算术运算、字符串运算、关系运算、逻辑运算的顺序进行。Visual Basic 6.0 中运算符的优先级如表 3.4 所示。

表 3.4 运算符的优先级

分类	运算符	类内优先级	总体优先级
算术运算符	^	高	高 ↓ 低
	- (负号)	↓ 低	
	*, /		
	\		
	Mod +, -		
字符串运算符	&, + (字符串连接)	同级	
关系运算符	>, >=, <, <=, =, <>, Like, Is	同级	
逻辑运算符	Not	高	↓ 低
	And	↓ 低	
	Or		
	Xor		
	Eqv		
	Imp		

3.4 常用内部函数

Visual Basic 6.0 中有两类函数：内部函数和用户自定义函数。自定义函数在以后的章节中讨论，这里着重讨论常用内部函数。

Visual Basic 6.0 提供了大量的内部函数（或标准函数）供用户在编程时使用。在程序中要使用一个函数时，只要给出函数名的相应的若干参数，就可以得到一个函数值。用户在编程时可以直接调用内部函数。Visual Basic 6.0 的内部函数包括：数学函数、字符串函数、转换函数、时间/日期函数、随机函数和格式输出函数等。下面就介绍一些常用的内部函数，其他函数可以查阅相关的资料。

以下叙述中，我们用 n 表示数值表达式，c 表示字符串表达式，d 表示日期表达式。若函数名后有 \$ 符号，则表示该函数返回值为字符串。

3.4.1 数学函数

函数	功能	示例	结果
----	----	----	----

Sin	返回弧度的正弦	Sin(1)	.841470984807897
Cos	返回弧度的余弦	Cos(1)	.54030230586814
Atn	返回用弧度表示的反正切值	Atn(1)	.785398163397448
Tan	返回弧度的正切	Tan(1)	1.5574077246549
Abs	返回数的绝对值	Abs(-2.4)	2.4
Exp	返回 e 的指定次幂	Exp(1)	2.71828182845905
Log	返回一个数值的自然对数	Log(1)	0
Rnd	返回小于 1 且大于或等于 0 的随机数	Rnd	0~1 之间的随机数
Sgn	返回数的符号值	Sgn(-100)	-1
Sqr	返回数的平方根	Sqr(16)	4
Int	返回不大于给定数的最大整数	Int(3.6)	3
Fix	返回数的整数部分	Fix(-3.6)	-3

3.4.2 转换函数

函数	功能	示例	结果
Asc(c)	将字符 c 转换成 ASCII 码值	Asc("A")	65
Chr(n)	将 ASCII 码值转换成字符	Chr(66)	"B"
LCase(c)	将字符串 c 中大写字母转换为小写字母	LCase("ABCabc")	"abcabc"
UCase(c)	将字符串 c 中小写字母转换为大写字母	LCase("ABCabc")	"ABCABC"
Str(n)	将数值 n 转换成字符串	Str(2.71828)	" 2.71828"
Val(c)	将数字字符串转换成数值	Val("3.141593")	3.141593

3.4.3 字符串函数

函 数	说 明	示例	结果
Len(C)	返回字符串的长度	Len("MyName=张三")	9
Left\$(C,N)	返回从字符串左边开始的指定数目的字符	Left\$("MyName",2)	"My"
Right\$(C,N)	返回从字符串右端开始的指定数目的字符	Right\$("MyName",4)	"Name"
Mid\$(C,N1[,N2])	返回从字符串指定位置开始的指定数目的字符	Mid\$("MyName",2,3)	"yNa"
Ltrim\$(C)	返回删除字符串左端空格后的字符串	LTrim\$(" MyName")	"MyName"

RTrim\$(C)	返回删除字符串右端空格后的字符串	RTrim\$(" Good ")	" Good"
Trim\$(C)	返回删除字符串前导和尾随空格后的字符串	Trim\$(" Good ")	"Good"
String\$(N, C)	返回包含一个字符重复指定次数的字符串	String\$(2, "ABCD")	"AA"
Space\$(N)	返回由指定数目空格字符组成的字符串	Space\$(5)	" "
InStr(n1, c1, c2)	返回字符串在给定的字符串中出现的开始位置	c= "student" InStr(3, c, "t")	7

3.4.4 时间/日期函数

函数	功能	示例	结果
Time	返回系统时间	Time	18:17:28
Date	返回系统日期	Date	2004-08-16
Year(d)	返回参数 d 的年号	Year(#2004/08/16#)	2004
Month(d)	返回参数 d 的月份号	Year(#2004/08/16#)	8
Day(d)	返回参数 d 的日期号	Day (#2004/08/16#)	16
WeekDay(d)	返回参数 d 的星期号	WeekDay (#2004/08/16#)	2

3.4.5 格式输出函数 Format

格式：Format(<表达式>, <格式字符串>)

功能：按格式字符串指定的格式输出表达式的值。

说明：表达式可以是数值型、字符串型、日期型数据。格式字符串中的常用格式说明符见表 3.5。

下面是 Format 函数的示例，运行结果如图 3.3 所示。

```
Private Sub Form_Click()
    Print Format(2.71828, "#####.##")
    Print Format(2.71828, "00000.00")
    Print Format(271828, "$##,###,###.##")
    Print Format(0.18, "###.###")
    Print Format(0.18, "0.000E+00")
    Print Format(Time, "ttttt")
    Print Format(Date, "dddddd")
End Sub
```

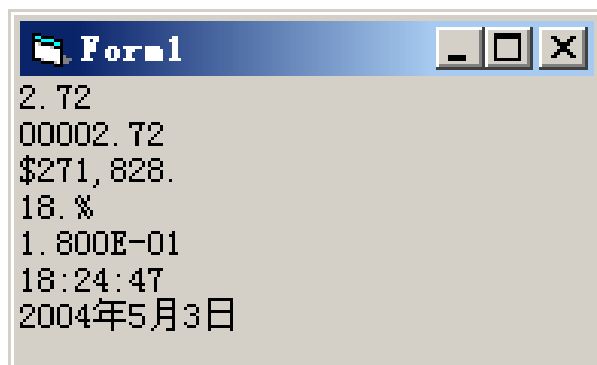


图 3.3 Format 函数示例

表 3.5 常用格式说明符

格式说明符	意义
#	数字占位符。显示一个数字或者什么都不显示，若小数点前后实际位数少于“#”的个数，不补 0
0	数字占位符。显示一个数字，若小数点前后实际位数少于“0”的个数，补 0
.	小数点占位符
,	千位分隔占位符
%	百分比号占位符，表达式的值自动乘以 100
\$	美元号占位符，常位于数值的首位
+,-	正负号占位符
E+,E-	指数符号占位符
@	字符占位符，显示字符或空白
ttttt	5 个 t，将时间按完整的时：分：秒格式显示
dddddd	6 个 d，将日期按控制面板中设置的长日期格式显示

3.5 编码规则

VB 和其他各种程序设计语言一样，编写代码时也有一定的收发室规则。主要规则如下：

1. VB 代码不区分字母的大小写

VB 系统对用户输入的程序代码进行自动转换，转换规则如下：

(1) 代码中输入的关键字，首字母总被转换成大写，其余字母被转换成小写。

(2) 若关键字由多个英文单词组成，则将每个英文单词的首字母转换成大写。

(3) 对于用户自己定义的变量、过程名，VB 以第一次定义的为准，以后输入的相同的变量、过程名向第一次输入的形式转换。

2. 语句书写自由

(1) 在同一语句行上，可以书写多个语句，语句之间用冒号“：”分隔。

(2) 一个单行语句可以分为若干行书写，书写时在本行后面加上续行符，即加上空格和下划线“_”。

(3) 一行最多允许书写 255 个字符。

3. 可以在语句中给出注释

为了便于程序的阅读、调试和维护，可以在语句中给出注释。注释的引导符有两种形式。

(1) 注释以 Rem 开头。如：Rem 例子一。“例子一”为注释内容，用 Rem 引出。

(2) 以单引号 “'” 引导注释内容。“'” 引导的注释内容可以放在语句的后面。

另外，可以使用【编辑】工具栏的【设置注释块】按钮和【解除注释块】按钮为选中的多行语句设置和取消注释。

4. 使用缩排格式

在结构化编码约定中，除了在程序的适当位置给出注释外，还应使用缩进的代码格式来反映程序的逻辑结构和嵌套。采用缩进格式可以使程序代码层次分明、结构清晰，便于阅读、理解和维护。默认情况下，一一行代码的起始位置按 Tab 键可缩进 4 个空格，按回删键 (BackSpace) 取消缩进。也可以【编辑】工具栏的【缩进】按钮和【凸出】按钮为选中的多行语句设置和取消缩进。

5. 语句可以使用行号与标号

在 VB6.0 的源程序中，保留了行号与标号的使用，但这不是必须的。行号由数字组成，标号是以字母开始，以冒号结束的字符串。它们一般用在转向语句中，但对结构化程序设计语言，转向语句应限制使用。

6. 显式声明变量

尽管 VB6.0 允许隐式声明变量，但是为了避免写错变量名引起麻烦，导致难以查找的错误，建议养成对变量进行显式声明的良好习惯。如前所述，在模块的声明段中加入 Option Explicit 语句，可以强制编译器发现所有未声明的变量。如果要在以后新建的所有模块中自动插入 Option Explicit 语句，可以执行【工具】菜单中的【选项】命令，打开如图 3.4 所示的【选项】对话框，在【编辑器】选项卡中选中【要求变量声明】复选框，单击【确定】按钮完成设置即可。

此外，本书在说明语法格式时采用以下约定：

- 必选项：用尖括号<>括起来，或不使用任何符号。
- 可选项：用方括号[]括起来。
- 多选项一：各选项之间用竖线|隔开。

例如：[对象.]Print [输出项列表][;|,]

上述符号只是为了便于解释语法格式，在实际代码中不应包含这些符号。

【练习】

1. 下列哪些符号是合法的变量名？

VB258、Sgn、88Ai、A\B、取消、Visual Basic

2. 下列符号哪些是常量，哪些是变量？

123、PI、True、“正确”、Good、8!、6e-5

3. 计算下列表达式的值。

- (1) 6>8
- (2) 21/2
- (3) 17\5
- (4) 9.8 Mod 5*2
- (5) True Xor Not 10
- (6) 8=6 And 8<6
- (7) Not 3>1 Imp 1<2
- (8) #5/552004#-5
- (9) "Sum" & 2001
- (10) "BG" +147

4. 求下列函数的值。
- (1) Len(“Hello, 济南铁职院 “)
 - (2) Right(“98765”, 3)
 - (3) Ltrim(“ 6982 ”)
 - (4) String(3, “ Good”)
 - (5) InStr(2, “ asdfasdf”, “ as”)
 - (6) Chr(“76”)
 - (7) Fix(15.86)
 - (8) Lcase(“3721efda”)
 - (9) Str(23.4567)
 - (10) Month(#5/4/2004#)
 - (11) Year(#05-08-04#)
5. 对于没有赋初值的变量，系统默认的值是什么？
6. 写出要产生下列随机数，所需的表达式。
- (1) 产生一个在区间(0, 20)内的随机数。
 - (2) 产生一个在区间[40, 65]上的随机整数。
 - (3) 产生一个两位的随机整数。
 - (4) 产生 C~K 内的随机字母。
7. 设 Y 是一个正实数，对 Y 的第四位小数四舍五入，应该怎样实现？

第 4 章 程序结构

本章要点：

- 顺序结构程序设计的方法及应用
- 选择结构程序设计的方法及应用
- 循环结构程序设计的方法及应用

程序结构是指程序中命令或语句执行的流程结构。Visual Basic 6.0 程序设计语言的控制结构与其他高级语言的程序控制结构一样，采用结构化的程序设计方法。使用结构化程序设计方法设计的程序结构清晰，易读性强，也易于查错和排错。结构化程序设计方法有三种基本控制结构：顺序结构、选择结构和循环结构。

4.1 顺序结构

顺序结构是指程序中的语句按出现的先后顺序依次执行，中间没有分支、循环和转移。顺序结构是一种线形结构，也是程序设计中最简单、最常用的基本结构，所有程序都包含这种结构。一些简单的程序可以只用顺序结构来编写，如冷面几章中介绍的示例程序都是仅采用了顺序结构。顺序结构如图 4.1 所示。

对于一个只包含顺序结构的程序来说，信息的一般流程为：输入—加工处理—输出。因此，在顺序结构中的典型语句主要是赋值语句、输入输出语句等。在

VB 中，信息的输入可以通过文本框等控件实现，输出可以通过标签、文本框等控件或 Print 方法实现。此外，VB 还提供了输入对话框和输出（消息）对话框，可以方便地完成输入和输出操作（有关对话框的使用将在第 5 章介绍）。

在第 3 章和第 3 章的讨论中已经多次使用过赋值语句和 Print 方法，从它们的应用中可以看出，赋值语句和 Print 方法均具有运算功能，是直接利用表达式对信息进行加工处理的有效手段。下面对赋值语句和 Print 方法作进一步说明，并简要介绍可对某个对象执行一系列操作的 With 结构。

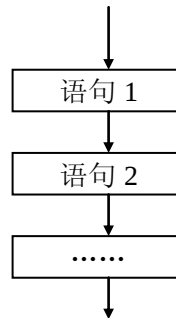


图 4.1 顺序结构

【讲授、举例】本部分内容主要以讲授为主，针对顺序结构，用生活中的例子来进行说明。

4.1.1 赋值语句

赋值语句是程序设计中最常用、最基本的语句，用于为变量或对象的属性赋值，格式如下：

格式 1：变量名 = 表达式

格式 2：[对象名.]属性名 = 表达式

在格式 2 中，若省略对象名，则默认对象为当前窗体。

在赋值语句中，表达式只能出现在赋值运算符（=）的右侧。表达式的数据类型应当与变量或属性的数据类型兼容。

当赋值号两侧的数据类型不同时，VB 按以下规则处理：

- （1）若两侧同为数值型但精度不同，则按左侧变量或属性的精度转换。
- （2）当左侧变量或属性为数值型时，若右侧不厌其烦 数字字符串，则转换为数值型再赋值；若表达式为空串或含有非数字字符，则出现错误。
- （3）非字符型表达式赋值给字符型变量或属性时均转换为字符型。
- （4）当数值型赋值给逻辑型时，0 转换为 False，非 0 转换为 True；当逻辑型转换为数值型时，False 转换为 0，True 转换为-1。

对象类型变量的赋值比较特殊，需要使用 Set 关键字，格式如下：

Set 对象变量名 = 表达式

4.1.2 Print 方法

1. 用 Print 方法输出数据

Print 方法用于在窗体、图片框或打印机等对象上输出数据，格式如下：

[对象名.]Print [输出项列表][{;|,}]

说明：

对象名：可以是窗体、图片框或打印机等对象，若省略对象名，则在当前窗体上输出数据。

输出项列表：要输出的内容（表达式）。若有多个输出项，可用逗号或分号隔开。

分号（；）：各输出项连续输出，中间无空格。

逗号（，）：各输出项按分区格式输出，即将一个输出行以 14 个字符的宽度为单位分成若干区段（称为“打印区”），每个区段输出一个表达式的值。

如果调用 Print 方法的语句以分号或逗号结束，则下一次执行 Print 方法时将在同一行输出；否则，每执行一次 Print 方法即自动换行。

Print 方法在 Form_Load 事件过程中不起作用。如果要在该事件中显示数据，必须在该过程内加上 Form.Show 方法或把窗体的 AutoRedraw 属性设置为 True。

技巧：在输入 Print 关键字时可以只输入问号（?），VB 会自动将其翻译成 Print。

2. 与 Print 方法有关的函数

VB 提供了 Spc 和 Tab 两个函数，用于配合 Print 方法对输出进行定位。

（1）Spc 函数

格式：Spc(n)

Spc 函数用于插入 n 个空格。例如：

```
Print "你好!"; Spc(8); "Hello!"
```

输出结果：

```
你好!      Hello!
```

（2）Tab 函数

格式：Tab[(n)]

Tab 函数用于将输出位置定位于第 n 列。若省略参数 n，则将插入点移动到下一个打印区的起点（此时与逗号作用相似）。如果 n 小于当前显示位置，则将输出位置移到下一行第 n 列。例如，若在窗体的 Form_Click 事件中加入以下代码，则：单击窗体后输出如图 4.2 所示的结果。

```
Print "1234567890"  
Print "Hello"; Tab(10); "China"  
Print "Hello"; Tab; "China"  
Print "Hello"; Tab(4); "China "
```



图 4.2 Tab 函数示例

【演示】建立窗体演示以上内容，并进行提问。

4.1.3 With 结构

使用 With 结构可以对某个对象执行一系列语句，而不必重复指出该对象的名称。With 结构的格式如下：

```
With 对象
    语句块
End With
```

例如，要改变一个对象的多个属性，可以在 With 结构中添加为该对象的多个属性赋值的语句，此时只需引用对象一次而不是在每个属性赋值时都要引用它。下面的示例说明了如何使用 With 结构来给同一个对象的几个属性赋值。

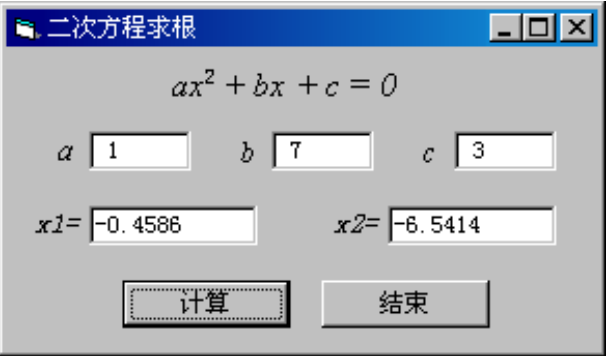
```
' 为文本框字体对象的多个属性赋值
With Text1.Font
    ' 下面只需输入圆点即可引用对象的属性或方法
    .Name = "隶书"      ' 字体名
    .Size = 12          ' 字号
    .Bold = True        ' 粗体
    .Italic = True      ' 斜体
    .Underline = True   ' 下划线
End With
```

注意：当程序一旦进入 With 结构，对象就不能改变。因此不能用一个 With 结构来设置多个不同的对象。

下面通过实例说明顺序结构程序设计的特点。

【例 4.1】求一元二次方程 $ax^2+bx+c=0$ 的根。

程序执行结果如下：



设计步骤如下：

(1) 新建工程，在窗体上添加五个文本框、两个命令按钮和五个标签。文本框均采用默认名称，Text 属性均为空。其他对象的属性设置如表 4.1 所示。

表 4.1 例 4.1 对象属性

对象 (名称)	属性	属性值	对象 (名称)	属性	属性值
窗体 (Form1)	Caption	二次方程求根	标签 (Label3)	Caption	c
命令按钮 (cmdCalcu)	Caption	计算	标签 (Label4)	Caption	x1
命令按钮 (cmdEnd)	Caption	结束	标签 (Label5)	Caption	x2
标签 (Label1)	Caption	a	标签 (Label6)	Caption	$ax+bx+c=0$
标签 (Label2)	Caption	b	标签 (Label7)	Caption	2

注：标签 Label17 用于显示二次方程的指数，六号字。

【实践】要求某位学生来建立程序运行的界面。代码部分由教师来完成。

(2) 编写代码。双击【计算】按钮，打开代码窗口，输入以下代码：

```
Private Sub cmdCalcu_Click()  
    Dim a As Single, b As Single, c As Single  
    Dim D As Single  
    Dim x1 As Single, x2 As Single  
    a = Val(Text1.Text)  
    b = Val(Text2.Text)  
    c = Val(Text3.Text)  
    D = b * b - 4 * a * c          ' 二次方程求根的判别式  
    x1 = (-b + Sqr(D)) / (2 * a) ' 用求根公式计算 x1 和 x2  
    x2 = (-b - Sqr(D)) / (2 * a)  
    ' 显示结果，最多保留 4 位小数  
    Text4.Text = Format(x1, "0.####")  
    Text5.Text = Format(x2, "0.####")  
End Sub
```

在【结束】按钮的单击事件中用 End 语句结束程序运行。

运行程序，单击【计算】按钮即可解出方程的根。运行效果如图 4.3 所示。

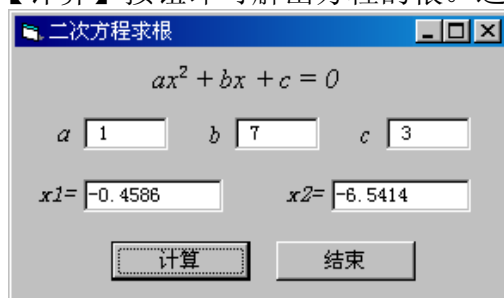


图 4.3 二次方程求根

【提问】：上面的程序是有什么缺陷的。

当 $a=0$ 或判别式（变量 D）的值小于 0 时，将出现错误，导致程序崩溃。因此，应设法判断变量 a 和 D 的值所处的范围，并作相应的处理。这不是顺序结构所能完成的任务，需采用下面将要介绍的选择结构才能实现。

4.2 选择结构

仅用顺序结构编写的程序一般比较简单，只能完成一些简单的任务，能够处理的问题类型也很有限。在实际应用中，有很多问题需要判断是否满足某种条件，从而进行不同的处理。选择结构可根据条件来选择控制程序的执行顺序，增加应用程序的灵活性。

选择结构的特点是：根据所给定的条件的真假，选择执行不同的语句。VB 中的选择结构主要是通过 If 语句和 Select Case 语句实现的。

4.2.1 If 语句

If 语句有多种结构形式，可根据实际应用的需要选用不同的形式。

1. If...Then 结构

If...Then 结构的流程如图 4.4 所示。该结构表示“如果条件满足就执行 Then 后边的语句，否则不执行任何操作”。语法格式如下：

(1) 单行形式

If 条件 Then 语句

(2) 块（多行）形式

If 条件 Then

语句块

End If

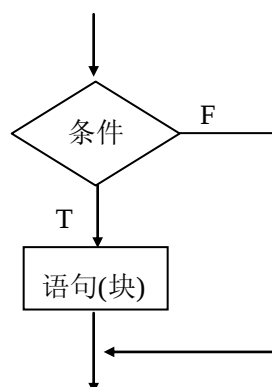


图 4.4 If...Then 结构

一般情况下，“条件”是运算结果为逻辑值的表达式，若表达式的值为 True，则条件成立，若表达式的值为 False，则条件不成立。“条件”也可以是运算结果为数值的表达式，此时 VB 将表达式的值解释成逻辑值：0 为 False，非零为 True。

需要注意的是，If...Then 的单行格式不用 End If 语句，整个语句必须写在一行上。如果条件为真时需要执行多条语句，所有语句必须在同一行上并且以冒号分开，如：

If 条件 Then 语句 1：语句 2：语句 3

为了使程序便于维护，提高其可读性，建议尽量不要采用这种单行多句的形式。如果需要在 Then 关键字后面执行多条语句，应使用块形式的 If...Then...End If 结构。

【例 4.2】编写程序，在文本框中输入一个整数，判断该数是不是偶数。

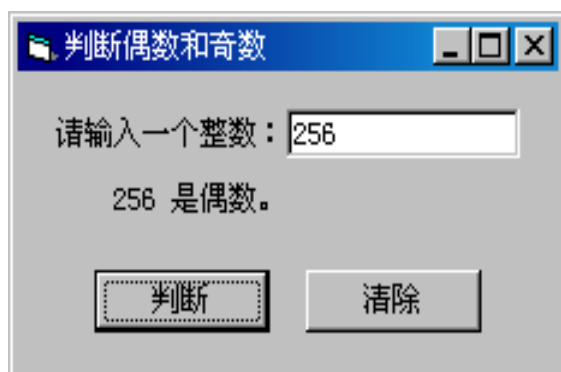


图 4.5 If...Then 示例

分析：判断某整数的奇偶性，可以检查该数能否被 2 整除。若某数能被 2 整除，则该数为偶数。

设计步骤如下：

- (1) 建立用户界面，设置对象属性。各对象的属性如表 4.2 所示。

表 4.2 例 4.2 对象属性

对象 (名称)	属性	属性值	对象 (名称)	属性	属性值
窗体 (Form1)	Caption	判断偶数和奇数	文本框 (Text1)	Text	
标签 (Label1)	Caption	请输入一个整数	命令按钮 (Command1)	Caption	判断
标签 (Label2)	Caption		命令按钮 (Command2)	Caption	清除

- (2) 编写程序代码。【判断】命令按钮的 Click 事件代码如下：

```
Private Sub Command1_Click()
    Dim x As Integer
    x = Val(Text1.Text)
    If x Mod 2 = 0 Then
        Label2.Caption = x & " 是偶数。"
    End If
End Sub
```

“清除”命令按钮 Click 事件的代码如下：

```
Private Sub Command2_Click()
    Text1.Text = ""
    Label2.Caption = ""
End Sub
```

【例 4.3】限制用户输入合法数据。

文本框是常用的数据输入控件，为了防止用户输入错误数据，可以在文本框的 KeyPress 事件中检查用户的按键。KeyPress 事件有一个 KeyAscii 参数，用于传送或改变用户按键的 ASCII 码。在 KeyPress 事件过程中将 KeyAscii 设置为 0 即可取消按键。在本例中，要求只能输入数字，若输入非数字字符则取消本次按键。代码如下：

```
Private Sub Text1_KeyPress(KeyAscii As Integer)
    If Not IsNumeric(Chr(KeyAscii)) And _
        KeyAscii <> 8 Then KeyAscii = 0
End Sub
```

说明：IsNumeric 是 VB 内部函数，用于判断参数是否为数字。“8”是回删键 Backspace 的 ASCII 码。

将上述代码应用到例 4.2 中观察其效果。

2. If...Then...Else 结构

上述 If...Then 结构实际上只提供了一种选择，即条件满足时执行指定的代码，条件不满足时执行任何操作。如果需要在条件不满足时执行另外一些语句，可以使用 If...Then...Else 结构。该结构的流程如图 4.6 所示，表示如果条件满足就执行 Then 后边的语句，否则就执行 Else 后的语句。语法格式如下：

(1) 单行形式

If 条件 Then 语句 1 Else 语句 2

(2) 块形式

If 条件 Then

语句块 1

Else

语句块 2

End If

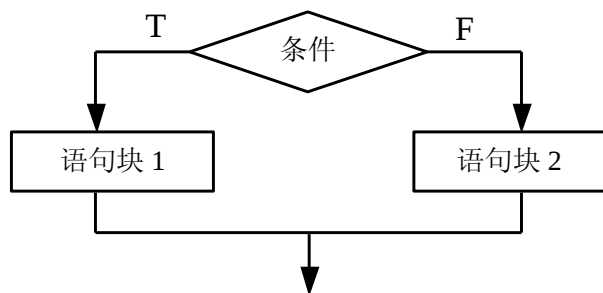


图 4.6 If...Then...Else 结构的流程

当程序运行到 If 语句时，首先测试条件，如果条件为 True，则执行 Then 之后的语句块 1，然后越过语句块 2，执行后续语句；如果条件为 False，则执行 Else 子句的语句块 2，然后执行后续的语句。下面通过实例说明该结构的应用。

【例 4.4】改进例 4.2，使程序能判断整数的奇偶性。

分析：判断某数的奇偶性，就是检查该数能否被 2 整除，若能被 2 整除，该数为偶数，

程序界面设计同例 4.3，只需将【判断】按钮单击事件的代码作如下改动。

```
Private Sub Command1_Click()
    Dim x As Integer
    x = Val(Text1.Text)
    If (x Mod 2) = 0 Then
        Label2.Caption = x & " 是偶数"
    Else ' 增加 Else 子句，显示奇数
        Label2.Caption = x & " 是奇数"
    End If
End Sub
```

运行结果如图 4.8 所示。

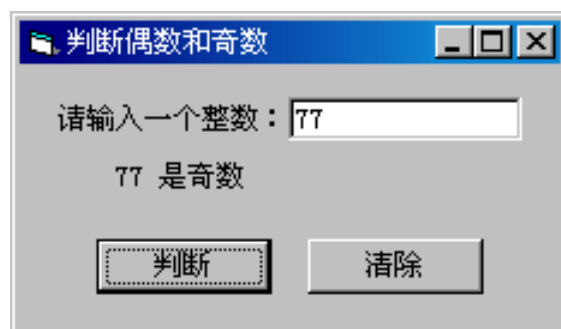


图 4.8 If...Then...Else 示例

3. If...Then...ElseIf 结构

上述 If 结构只能根据单一条件决定程序的流程。在处理实际问题时，常常需要对多种条件进行边疆的判断，并根据判断的结果执行不同的操作。此时可以使用 If...Then...ElseIf 结构。该结构的流程如图 4.9 所示。语法格式如下：

```

If 条件 1 Then
    语句块 1
ElseIf 条件 2 Then
    语句块 2
. . .
ElseIf 条件 n Then
    语句块 n
Else
    语句块 n+1
End If

```

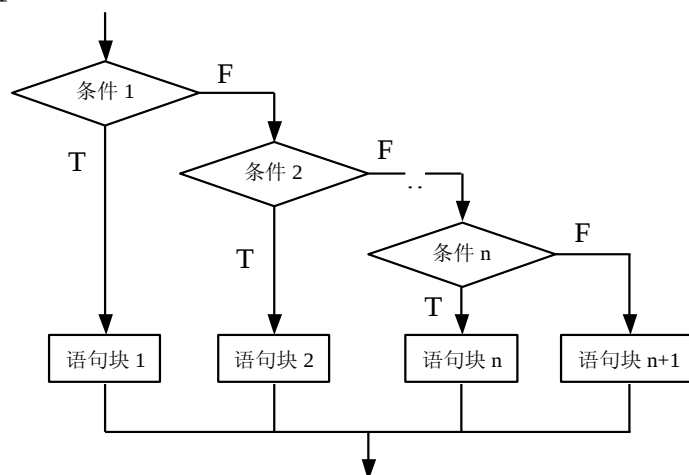


图 4.9 If...Then...ElseIf 结构的流程图

该结构的作用是根据不同条件执行指定的语句块。当程序运行到 If 语句时，首先测试条件 1，如果条件 1 为 True，则执行语句块 1，然后 End If 之后的语句；如果条件 1 为 False，就测试条件 2，依次类推，直到找到一个为 True 的条件就执行相应的语句块。如果条件都不为 True，则执行 Else 后的语句块 n+1。

下面通过实例说明 If...Then...ElseIf 结构的应用。

【思考】改进例 4.1 求一元二次方程 $ax^2+bx+c=0$ 的根的程序，修正其中的缺陷。

分析：首先应判断方程中二次项的系数 a ：若 $a=0$ ，则不是二次方程，不作求根运算；若 $a \neq 0$ ，则根据判别式 b^2-4ac 的值求根。方程的根有三种情况：

- ① $b^2-4ac=0$ ，方程有两个相等的实根；
- ② $b^2-4ac>0$ ，方程有两个不等的实根；
- ③ $b^2-4ac<0$ ，方程无实根，有两个共轭复根。

程序界面设计与例 4.1 相同。根据以上分析，将【计算】按钮单击事件过程的代码修改如下：

```
Private Sub cmdCalcu_Click()
    Dim a As Single, b As Single, c As Single
    Dim D As Single
    Dim x1, x2 '将 x1、x2 声明为变体型，可以存放不同类型的数据
    a = Val(Text1.Text)
    b = Val(Text2.Text)
    c = Val(Text3.Text)
    D = b * b - 4 * a * c '二次方程求根的判别式
    If a = 0 Then 'a=0，不是二次方程
        x1 = ""
        x2 = ""
    ElseIf D = 0 Then 'a≠0，D=0，方程有相等二实根
        x1 = -b / (2 * a)
        x2 = x1
    ElseIf D > 0 Then 'a≠0，D>0，方程有不等二实根
        x1 = (-b + Sqr(D)) / (2 * a)
        x2 = (-b - Sqr(D)) / (2 * a)
    Else
        x1 = "无实根" 'a≠0，D<0，无实根
        x2 = x1
    End If
    '显示结果，最多保留 4 位小数
    Text4.Text = Format(x1, "0.####")
    Text5.Text = Format(x2, "0.####")
End Sub
```

在上述代码中，变量 D 存放判别式 (b^2-4ac) 的值。实际上，当 $D=0$ 或 $D>0$ 时，都是用求根公式计算 x_1 和 x_2 ，因此可将两个 ElseIf 子句合并：

```
ElseIf D >= 0 Then 'D≥0，方程有两个实根
    x1 = (-b + Sqr(D)) / (2 * a)
    x2 = (-b - Sqr(D)) / (2 * a)
```

【练习】设计一程序，完成以下计算：企业发放的奖金根据利润提成。利润(I) 低于或等于 10 万元时，奖金可提 10%；利润高于 10 万元，低于 20 万元时，低于 10 万元的部分按 10%提成，高于 10 万元的部分，可提成 7.5%；20 万到 40 万之间时，高于 20 万元的部分，可提成 5%；40 万到 60 万之间时高于 40 万元的部分，可提成 3%；60 万到 100 万之间时，高于 60 万元的部分，可提成 1.5%，高于 100 万元时，超过 100 万元的部分按 1%提成，从输入当月利润 I，求应发放奖金总数？

4. If 语句的嵌套

嵌套是指在一个控制结构中插入另一个控制结构。If 语句的嵌套是指在一个 If 语句中插入另一个 If 语句。内嵌的 If 语句可以出现在关键字 Then 或 Else 之后的语句块中。If 语句嵌套常用于复杂的多分支选择，它的一般形式如下：

```
If 条件 1 Then
    ...
    If 条件 2 Then
        ...
    End If
    ...
End If
```

例如，例 4.5 中二次方程求根的 If 语句可以改为下面的形式：

```
If a = 0 Then          ' a=0, 不是二次方程
    x1 = ""
    x2 = ""
Else                  ' a≠0
    If D >= 0 Then      ' 内嵌 If 语句开始。 D≥0, 方程有两个实根
        x1 = (-b + Sqr(D)) / (2 * a)
        x2 = (-b - Sqr(D)) / (2 * a)
    Else
        x1 = "无实根"   ' D<0, 无实根
        x2 = x1
    End If              ' 内嵌 If 语句结束
End If
```

从上面的示例中可以看出，当条件为假又内嵌 If 语句时（If 语句内嵌在 Else 子句中），可以用 If...Then...ElseIf 结构代替 If 语句的嵌套。这两种结构均可实现多分支选择。

If 语句的嵌套使用时应注意 If 语句与 End If 的配对关系，多个 If 嵌套中，End If 与它最近的 If 配对。为了增强程序的可读性，嵌套结构的书写应采用锯齿型缩进。

4.2.2 Select Case 语句

当对同一个表达式的多种不同取值情况作不同处理时，用 If...Then...ElseIf 结构或 If 语句嵌套会使程序显得较为杂乱，而用 Select Case 语句（情况语句）可使程序更加清晰易读。

Select Case 语句的语法格式如下：

```
Select Case 测试表达式
    Case 值表 1
        语句块 1
    Case 值表 2
        语句块 2
    ...
    Case 值表 n
        语句块 n
```

Case Else
 语句块 n+1
 End Select

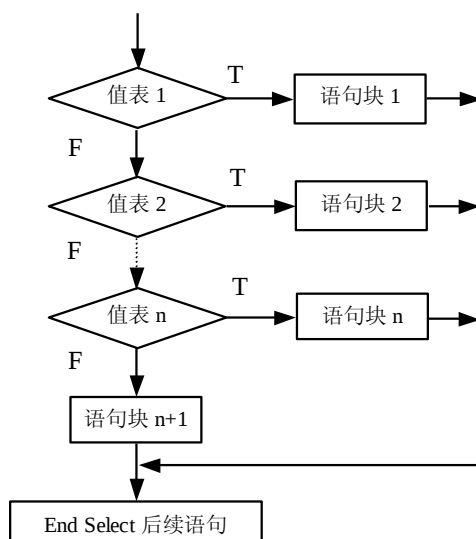


图 4.10 Select Case 语句的流程

Select Case 语句的流程如图 4.10 所示。该语句的功能是根据“测试表达式”的值，选择执行第一个符合条件的语句块。执行该语句时，首先求出“测试表达式”的值，然后依次将该值与结构中各 Case 子句的“值表”进行比较。如果相等，就执行与该 Case 相关联的语句块，然后执行 End Select 后面的语句。如果没有匹配的“值表”，则执行 Case Else 后面的语句块，然后执行 End Select 后面的语句。如果有多个“值表”与测试值匹配，则只有第一个匹配的 Case 子句的有效。

Select Case 语句中的“测试表达式”可以是变量、算术表达式或字符串表达式。它的类型必须与 Case 子句后面“值表”的类型相同。

Case 子句后面“值表”可以是单一值或一组值的列表，有三种表示形式。“值表”由以下形式的表达式组成：

形式	示例	说明
(1) 表达式	Case 100+a	数值或字符串表达式
(2) 一组用逗号分隔的枚举值	Case 2, 4, 6, 8	测试表达式的值等于 2, 4, 6, 8 之一
(3) 表达式 1 To 表达式 2	Case 1 To 10	$1 \leq \text{测试表达式的值} \leq 10$
(4) Is 关系运算符表达式	Case Is > 20	测试表达式的值 > 20

1. 表达式

可以是常量、变量、算术表达式或字符串表达式。如果有多个表达式，则表达式之间要用逗号隔开，例如：表达式 1，表达式 2，…。其中逗号表示“或者”，即只要其中有一个表达式的值与测试值匹配，就执行该 Case 子句后的语句块。例如：

```

Select Case iMonth ' iMonth 为变量，用作测试表达式
  Case 1, 3, 5, 7, 8, 10, 12 ' “值表” 为常量列表
    Print iMonth & " 月：31 天"
  Case 4, 6, 9, 11
    Print iMonth & " 月：30 天"
  Case 2

```

```

        Print iMonth & " 月：平年 28 天，闰年 29 天"
    Case Else
        Print " 出错"
    End Select

```

2. 表达式 1 To 表达式 2

表示取一定范围的值，其中，表达式 1 的值必须小于表达式 2 的值。若指定多个取值范围，则用逗号隔开。例如：

```

Case 0 To 100          ' 测试表达式的值在 0~100 之间
Case "a" To "z", "A" To "Z"      ' 所有小写和大写字母
Case "0" To "9"          ' 所有数字字符

```

3. Is 关系运算符表达式

Is 关键字可以配合关系运算符来指定一个数值范围。如果在编写代码时未 Is，则 VB 行动在关系运算符前面插入 Is 关键字。若指定多个取值范围，也要用逗号隔开。例如：

```

Case Is > 100
Case Is > 100, Is < 0

```

上述三种形式既可以单独使用，也可以混合使用。例如：

```

Case 1, 3, 5, 7, 8, 9 To 20, Is > 30

```

【例 4.6】设计一个程序，输入成绩，根据以下条件判断成绩的等级：

```

90~100  等级 A
80~89   等级 B
70~79   等级 C
60~69   等级 D
60 以下 等级 E

```

(1) 新建工程，在窗体中添加两个标签，一个文本框，一个标题为“显示等级”的命令按钮。各对象的属性设置如表 4.3 所示。

表 4.3 例 4.3 对象属性

对象（名称）	属性	属性值	对象（名称）	属性	属性值
标签(Label1)	Caption	请输入成绩	文本框(Text1)	Text	
标签(Label2)	Caption		命令按钮(Command1)	Caption	显示等级

(2) 为【显示等级】按钮的单击事件编写如下代码：

```

Private Sub Command1_Click()
    Dim Score As Integer
    Score = Val(Text1.Text)
    Select Case Score
        Case 90 To 100
            Label2.Caption = Score & "分为等级 A"
        Case 80 To 89
            Label2.Caption = Score & "分为等级 B"
        Case 70 To 79
            Label2.Caption = Score & "分为等级 C"
        Case 60 To 69
            Label2.Caption = Score & "分为等级 D"
    End Select
End Sub

```

```

        Case 0 To 59
            Label2.Caption = Score & "分为等级 E"
        Case Else
            Label2.Caption = "成绩有误，请重新输入！"
        End Select
    End Sub

```

程序运行如图 4.11 所示。



图 4.11 Select Case 示例

从以上示例可以看出，使用 Select Case 语句实现多分支结构更加直观，程序可读性强。但应注意，并非所有的多分支结构均可用 Select Case 语句代替 If...Then...ElseIf 语句。当对多个变量或表达式进行条件判断时（如例 4.5），仍然要用 If...Then...ElseIf 结构。

【练习】用 Select Case 语句完成利润分配题目。

4.3 循环结构

在编程中经常遇到这样的情况：某一类问题的计算或处理方法完全一样，只是要求重复计算或处理多次，例如求若干数的总和，统计学生各分数段的人数，银行存款利率的计算等。类似这样的问题，就要用到循环结构。

所谓循环结构是指对同一程序段重复执行若干次，被重复执行的语句块称为循环体。循环体的执行与否以及次数多少视循环类型和条件而定。VB 中常用的循环语句有 For...Next 语句和 Do...Loop 语句。For...Next 循环用于已知循环次数的情况下，而 Do...Loop 循环主要用于不知道循环次数的情况下，在给定的条件满足时执行循环体。下面分别介绍这两种循环结构。

4.3.1 For...Next 循环

For...Next 循环简称 For 循环。如果知道循环要执行多少次时，就可以使用 For 循环。For 循环使用一个循环变量（计数器）控制循环体的执行次数。每执行一次循环之后，循环变量的值就会自动增加或者减少。

For 循环的语法格式如下：

```

For 循环变量 = 初值 To 终值 [Step 步长]
    [循环体]
Next [循环变量]

```

1. 格式说明:

循环变量: 必须为数值型。For 和 Next 关键字后面的循环变量必须相同。

初值和终值: 均为数值型, 可以是数值表达式。

步长: 数值型, 可以是数值表达式, 默认值为 1。若步长为正数, 应设初值 $\leq$ 终值; 若步长为负数, 应设初值 $\geq$ 终值, 否则循环体不会被执行。步长不应为 0, 否则程序将陷入无限循环(死循环)。

循环体: 在循环中被执行的语句块。若循环体中无语句, 则为空循环。在循环体中可根据条件加入 Exit For 语句强制退出循环。Exit For 通常出现在选择结构中。

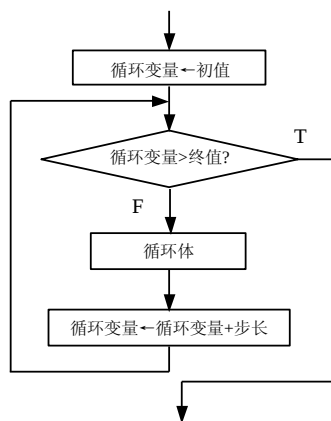


图 4.12 For 循环

For 循环的流程如图 4.12 所示, 执行过程如下:

- (1) 循环变量被赋初值。
- (2) 若步长为正, 判断循环变量的值是否大于终值, 若步长为负正, 判断循环变量的值是否小于终值。
- (3) 若判断结果为真, 则退出循环, 否则执行循环体中的语句。
- (4) 遇到 Next 则把循环变量加上步长。
- (5) 重复步骤 (2) ~ (4)。

2. For 循环应用实例

【例 4.7】编制程序, 计算 $1+2+3+\dots+200$ 的整数和。在窗体上放置一个命令按钮, 编写如下代码:

```
Private Sub Command1_Click()
    Dim Sum As Integer, i As Integer
    Sum = 0
    For i = 1 To 200
        Sum = Sum + i
    Next i
    Print "1 + 2 + ... + 200 = " & Sum
End Sub
```

运行结果如图 4.13 所示。

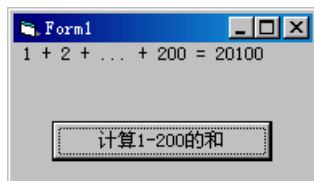


图 4.13 求 1~200 之和

提示：累加和连乘（如求某数的阶乘 $n!$ ）是程序设计中的常用算法，均可采用循环结构实现。在进入循环之前，应当为存放累加和的变量（如本例中的 Sum）或存放连乘积的变量设置初始值，通常将累加变量设置为 0，将连乘变量设置为 1。

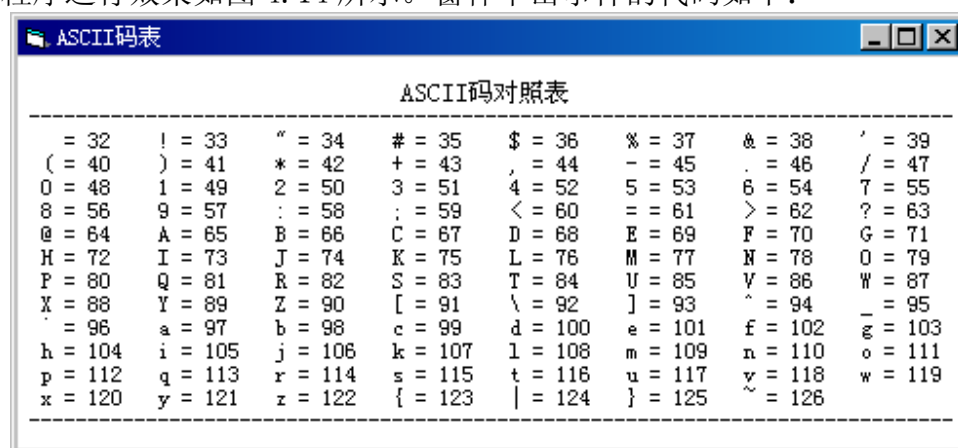
【练习】编写程序，在两个文本框中输入两个整数，然后计算从第一个整数到第二个整数的和。

【例 4.8】制作 ASCII 码对照表。ASCII 码（美国信息交换标准码）是 7 位二进制字符集，用来表示标准美制键盘上的字母、符号以及控制字符。其中，可打印字符的编码值范围为 32~126（32=空格）。利用 Chr 函数可以将字符代码转换为对应的字符。

（1）设计界面。程序的界面很简单，窗体上无任何控件。将窗体的背景色设为白色，Caption 属性设为 **【ASCII 码表】**。

（2）编写代码。由于可打印字符的 ASCII 码对照表具有明确的起止范围，因此，很适于用 For...Next 循环制作。单击窗体时，在循环中用 Print 方法将 ASCII 字符及其代码直接显示在窗体上，格式为“字符=字符代码”，每行显示 8 个字符及其代码，用 Tab 函数控制每个 ASCII 码的打印位置。

程序运行效果如图 4.14 所示。窗体单击事件的代码如下：



= 32	! = 33	" = 34	# = 35	\$ = 36	% = 37	& = 38	' = 39
(= 40	) = 41	* = 42	+ = 43	, = 44	- = 45	. = 46	/ = 47
0 = 48	1 = 49	2 = 50	3 = 51	4 = 52	5 = 53	6 = 54	7 = 55
8 = 56	9 = 57	: = 58	; = 59	< = 60	= = 61	> = 62	? = 63
@ = 64	A = 65	B = 66	C = 67	D = 68	E = 69	F = 70	G = 71
H = 72	I = 73	J = 74	K = 75	L = 76	M = 77	N = 78	O = 79
P = 80	Q = 81	R = 82	S = 83	T = 84	U = 85	V = 86	W = 87
X = 88	Y = 89	Z = 90	[= 91	\ = 92	] = 93	^ = 94	_ = 95
` = 96	a = 97	b = 98	c = 99	d = 100	e = 101	f = 102	g = 103
h = 104	i = 105	j = 106	k = 107	l = 108	m = 109	n = 110	o = 111
p = 112	q = 113	r = 114	s = 115	t = 116	u = 117	v = 118	w = 119
x = 120	y = 121	z = 122	{ = 123	= 124	} = 125	~ = 126	

图 4.14 ASCII 码对照表

```
Private Sub Form_Click()
    Dim intASC As Integer
    Dim i As Integer
    Cls
    Print
    Me.FontSize = 10 ' 设置字号
    Print Tab(29); "ASCII 码对照表"
    Me.FontSize = 9
    ' String 函数返回指定数目的重复字符
    Print " "; String$(79, "-")
    ' intASC 为循环变量，并代表要打印的 ASCII 码
    For intASC = 32 To 126
        Print Tab(10 * i + 3); _
            Chr(intASC); " = "; intASC;
        i = i + 1
        If i = 8 Then ' 每行显示 8 个 ASCII 码
```

```

        i = 0
        Print
    End If
Next intASC
Print vbCr; " "; String$(79, "-")
End Sub

```

4.3.2 Do...Loop 循环

Do...Loop 循环简称 Do 循环，主要在循环次数未知时使用。Do 循环有两类语法形式，即前测型循环和后测型循环。

前测型循环（先判断，后执行）

Do [{While|Until} 条件]

[循环体]

Loop

后测型循环（先执行，后判断）

Do

[循环体]

Loop [{While|Until} 条件]

说明：

（1）前测型循环先判断条件，如果为真，执行循环体，否则退出，因此有可能一次也不执行循环体；后测型循环先执行循环体，然后判断条件，因此至少执行一次循环体。

（2）While 关键字是指当条件为真时执行循环体；Until 与之相反，条件为假时执行循环体，直到条件为真时退出循环。二者在功能上并无本质区别，只要将条件取反，就可以互相取代。例如，Do While x>=10 与 Do Until x<10 是等价的。

（3）在循环体中可以插入 Exit Do 语句，随时跳出循环。Exit Do 通常用于条件判断之后（如 If...Then）。

（4）如果省略了“{While|Until} 条件”子句，则为无条件循环，此时应在循环体内适当位置插入 Exit Do 语句，否则会陷入死循环。

使用 While 关键字的前测型和后测型 Do 循环的流程如图 4.15 和图 4.16 所示。使用 Until 关键字的 Do 循环与之相似，只需将图中的“T”和“F”互换位置即可。

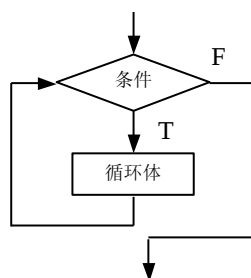


图 4.15 Do While...Loop 循环

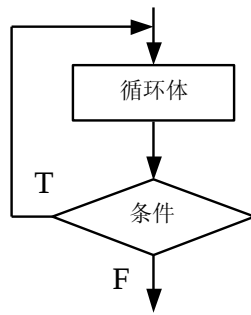


图 4.16 Do...While Loop 循环

下面通过实例说明 Do 循环的应用。

【例 4.9】用 Do While...Loop 循环计算 1 到 200 的奇数和（1+3+5+...+199）。

在窗体上放置一个命令按钮，编写如下代码：

```

Private Sub Command1_Click()
    Dim Sum As Integer, i As Integer
    Sum = 0
    i = 1
    Do While i <= 200
        Sum = Sum + i
        i = i + 2
    Loop
    Print " 1 + 3 + 5 + ... + 199 = " & Sum
End Sub
  
```

运行结果如图 4.17 所示。

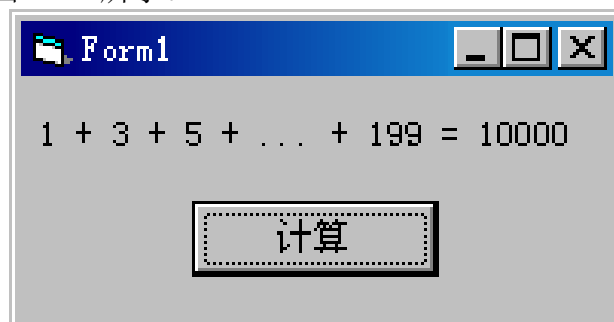


图 4.17 求 1~200 奇数和

上述功能亦可用 For 循环实现：

```

For i = 1 To 200 Step 2
    Sum = Sum + i
Next
  
```

由此可见，Do 循环完全可以代替 For 循环。尽管如此，在已知循环次数的情况下，还是应当使用 For 循环，它将使程序更加简洁，效率更高。

【例 4.10】用“辗转相除法”求两个自然数 m 和 n 的最大公约数。

用“辗转相除法”求最大公约数的计算方法如下：

- (1) 两数相除（m 除以 n），取余数 r；
- (2) 若 $r \neq 0$ ，则将除数改作被除数，余数改作除数（ $m \leftarrow n, n \leftarrow r$ ），重复步骤（1）、（2），直到 $r = 0$ 为止；
- (3) 最后一次相除时所用的除数就是最大公约数。

程序设计：本例采用 Do...Loop Until 循环实现。程序界面及运行效果如图 4.18 所示。各对象的属性设置如表 4.4 所示。

表 4.4 例 4.10 对象属性

对象（名称）	属性	属性值	对象（名称）	属性	属性值
窗体（Form1）	Caption	最大公约数	标签（Label1）	Caption	输入两个自然数
文本框（Text1）	Text		标签（Label2）	Caption	M=
文本框（Text2）	Text		标签（Label3）	Caption	N=
命令按钮（Command1）	Caption	计算	标签（Label4）	Caption	

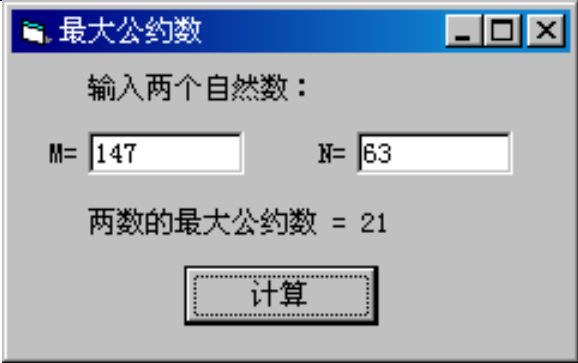


图 4.18 求最大公约数

【计算】按钮单击事件的代码如下：

```
Private Sub Command1_Click()  
    Dim m As Integer, n As Integer, r As Integer  
    m = Val(Text1.Text)  
    n = Val(Text2.Text)  
    ' 若数据超出有效范围，退出本过程  
    If m <= 0 Or n <= 0 Then Exit Sub  
    Do  
        ' 求最大公约数  
        r = m Mod n  
        m = n  
        n = r  
    Loop Until r = 0 ' r=0 时退出循环  
    ' 退出循环时, m 中存放的是最后的除数, 即最大公约数  
    Label4.Caption = "两数的最大公约数 = " & m  
End Sub
```

【总结】

- 1、对于绝大部分的程序，两种语句都可以互换使用。
- 2、循环程序两种写法：穷举法与迭代法。

4.3.3 循环的嵌套

在循环结构中可以嵌套任何循环结构，也可以嵌套选择结构。

【例 4.11】求 100~200 之间的素数。

【分析】：素数或称质数，是指一个大于1的整数，除了1和它本身以外不能被其他正整数整除，这个数就是素数。判断一个数 n 是否为素数，只要依次用 $2 \sim \sqrt{n}$ 作除数去除 n ，若 n 不能被其中任何一个数整除，则 n 即为素数。

让学生首先考虑如何判断一个数是否是素数。如果只是判断一个数是否是素数，我们完全可以用一层循环解决问题。再要求学生考虑如果需要判断多个有规律的数的解决方案。

本题的求解可用双重循环实现，外循环遍历 100~200 之间的所有整数，内循环判断各数是否为素数。

程序设计：在窗体上添加一个命令按钮，设 Caption 属性为【开始】。运行时单击该按钮后用 Print 方法显示 100~200 之间的素数。【开始】按钮单击事件的代码如下：

```
Private Sub Command1_Click()  
    Dim n As Integer, i As Integer, j As Integer  
    Dim flag As Boolean '判断 n 是否为素数的标志  
    Print vbCrLf; Tab(8); "100~200 之间的素数"; _  
        vbCrLf; String(35, "-")  
    For n = 100 To 200 '外循环遍历 100~200 之间所有整数  
        flag = True '先假定 n 为素数  
        For i = 2 To Int(Sqr(n)) '内循环判断 n 是否为素数  
            If n Mod i = 0 Then '若 n 能被 i 整除，不是素数  
                flag = False '修改标志  
                Exit For '退出内循环  
            End If  
        Next i  
        If flag Then '若 n 为素数，显示（每行 7 个数）  
            Print n;  
            j = j + 1  
            If j Mod 7 = 0 Then Print  
        End If  
    Next  
    Print String(35, "-")  
End Sub
```

程序运行效果如图 4.19 所示。



图 4.19

【例 4.12】制作如图 4.20 所示的“九九乘法表”。

九九乘法表									
1×1=1									
2×1=2	2×2=4								
3×1=3	3×2=6	3×3=9							
4×1=4	4×2=8	4×3=12	4×4=16						
5×1=5	5×2=10	5×3=15	5×4=20	5×5=25					
6×1=6	6×2=12	6×3=18	6×4=24	6×5=30	6×6=36				
7×1=7	7×2=14	7×3=21	7×4=28	7×5=35	7×6=42	7×7=49			
8×1=8	8×2=16	8×3=24	8×4=32	8×5=40	8×6=48	8×7=56	8×8=64		
9×1=9	9×2=18	9×3=27	9×4=36	9×5=45	9×6=54	9×7=63	9×8=72	9×9=81	

图 4.20 九九乘法表

【分析】：“九九乘法表”由 9 行 9 列等式组成，若以变量 i 代表行号，变量 j 代表列号，则所有等式均可表示为： $i * j = \text{乘积}$ 。在图 4.18 中，每行内等式的个数等于该行的行号。对这种具有明显行列规律的问题，通常采用 For...Next 双重循环解决。设外循环的循环变量为行号，内循环的循环变量为列号，在内循环中输出一行中的各列，退出内循环后行号加 1，输出下一行。

程序设计：设窗体的背景色（BackColor 属性）为白色。程序运行时单击窗体后调用窗体的 Print 方法直接在窗体上显示乘法表，在 Print 方法中用 Tab 函数控制各列等式的输出位置。窗体单击事件的代码如下：

```
Private Sub Form_Click()
    Dim strS As String, i As Integer, j As Integer
    Cls
    Print vbCr; Tab(41); "九九乘法表"
    Print " " & String$(89, "-")
    For i = 1 To 9      ' 外循环变量 i 为乘法表的"行"
        For j = 1 To i  ' 内循环变量 j 为乘法表的"列"
            strS = i & "×" & j & "=" & i * j  ' 行列相乘
            Print Tab((j - 1) * 10 + 3); strS; ' 显示
        Next j
        Print
    Next i
    Print " " & String$(89, "-")
End Sub
```

1. 指出赋值语句中的错误。

- (1) $a+b=x+y$
- (2) $x="123" + "x"$ (x 为整型变量)
- (3) $y=" "$ (y 为单精度型变量)
- (4) $3x=x3$
- (5) $y=\text{Sqr}(-5)*x$

2. 编制一个温度转换程序，实现摄氏温度 $^{\circ}\text{C}$ 与华氏温度 F 的相互转换。相

关公式为：
$$^{\circ}\text{C} = \frac{5}{9}(F - 32) \qquad F = \frac{9}{5}^{\circ}\text{C} + 32$$

3. 编制程序，通过文本框输入 a 、 b 、 c 三个数，用标签显示最大数和最小数。

4. 编制程序，通过文本框输入年份和月份，显示该月的天数。注意判断年份是否为闰年：年号能被 4 整除，但不能被 100 整除，或者年号能被 400 整除的年份为闰年。

5. 设计一个程序，通过文本框输入两个正整数 M 和 N ($M < N$)，计算从 M 到 N 的整数和以及偶数和，用标签显示结果。

6. 编制程序，通过文本框输入自然数 n，计算其阶乘 $n!$ ($n! = 1 \times 2 \times 3 \times \cdots \times n$)。

7. 编制一个将十进制整数转换为二进制数的程序。

8. 在窗体上显示如图 4.21 所示由*组成的菱形图案。要求用两种方法实现：

(1) 用单层循环结合 Spring 函数实现；

(2) 不使用 Spring 函数，用双重循环实现。

考虑：把 C 语言中学习过的程序，用 VB 实现？

第 5 章 与用户对话

本章要点

- 输入对话框的创建和使用
- 消息对话框的创建和使用
- 文件对话框、颜色对话框、字体对话框、打印对话框、帮助对话框等通用对话框的创建和使用
- 自定义对话框的创建和使用

通过前面几章的学习，同学们已经具备了基本的语法知识，足以向开发专业性应用程序进军了。从功能上看，一个完整的应用程序通常包括三部分：数据输入、数据处理和数据输出。将要加工的数据从外部设备(如键盘)输入到计算机中，并将处理结果输出到指定设备(如显示器)是程序设计语言所应具备的基本功能。VB 的输入输出有着十分丰富的内容和形式，它提供了多种手段，既可通过各种控件实现输入输出操作，又可通过特定函数和语句与用户对话，使输入输出更加灵活、方便和直观。本章主要讨论各种对话框的创建和使用。

5.1 输入对话框

在 VB 中，数据的输入的方式主要有两种。一种方式是使用文本框等具有输入功能的控件，这是在前面的章节中多次采用的方式，它的优点是功能较强，形式灵活，缺点是程序设计的步骤较多。另一种方式是利用输入框函数 InputBox 通过输入对话框输入数据，它只需一行代码即可实现输入窗体的功能，减少了编程的工作量。。

InputBox 的语法格式为：

InputBox (提示信息[, 对话框标题][, 默认值][, X, Y])

该函数的功能是产生一个对话框，作为输入数据的界面，等待用户输入并返回所输入的内容。

函数中各参数的作用如下：

提示信息：字符串表达式，在对话框内显示提示信息，最大长度为 1024 个字符。如果包含多行，可以在各行之间用回车符 Chr(13)、换行符 Chr(10) 或回车换行符的组合 Chr(13) & Chr(10) 来分隔，对应的 VB 常数分别为 vbCr、vbLf 和 vbCrLf。

对话框标题：字符串表达式，显示在标题栏中作为对话框的标题。若省略，则默认为当前工程的名称。

默认值：字符串表达式，显示在对话框的文本框中，在没有其他输入时作为默认输入值使用。默认为空。

X、Y：均为数值表达式，分别指定对话框左边和上边与屏幕左边和上边的距离，单位为缇 (twip)。如果省略，对话框在水平方向居中，垂直方向约为屏幕的上三分之一处显示。X 和 Y 应成对出现，否则无效。

说明：

(1) 在对话框中，如果用户单击【确定】按钮，则 InputBox 函数返回文本框中所有内容；如果单击【取消】按钮，则函数返回零长度的字符串。

(2) 除提示信息外，其他参数均为可选参数。如果指定了后面的参数而省略前面的参数，则必须保留前面的逗号。例如：

```
strNo = InputBox("输入编号", , "001")。
```

(3) 使用 InputBox 函数一次只能输入一个数据。如果要输入多个数据，则必须多次使用该函数。

【例 5.1】通过 InputBox 函数输入两个数字存入变量 a、b，然后将其互换。在窗体上添加一个命令按钮，设 Caption 为【输入数字】。按钮单击事件代码如下：

```
Private Sub Command1_Click()  
    Dim a, b, t  
    Cls  
    a = Val(InputBox("请输入 a :", "输入数字"))  
    b = Val(InputBox("请输入 b :", "输入数字"))  
    Print " 交换前: a ="; a; ", b ="; b  
    t = a: a = b: b = t ' 交换  
    Print " 交换后: a ="; a; ", b ="; b  
End Sub
```

运行结果如图 5.1 所示，左图为 InputBox 函数生成的对话框，右图为窗体显示结果。

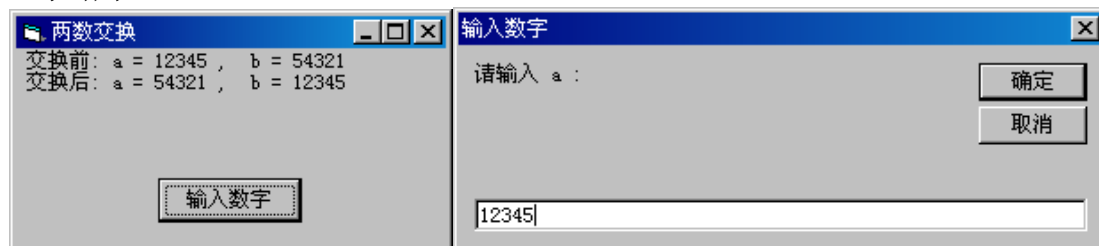


图 5.1 InputBox 函数示例

【练习】编写程序，利用 InputBox 函数输入一个正整数，然后对该正整数分解质因数，并在窗体中输出结果。

5.2 消息对话框

在应用程序中，常常需要在屏幕上显示诸如提示、询问、警告或错误等信息，对用户的操作作出提醒或反馈。例如，在使用 Word 编辑文档时，若尚未保存就单击了关闭按钮，系统会弹出如图 5.2 所示的消息对话框。对话框实际上是一种特殊的窗体，如果由我们自己来编制，则需在窗体上添加按钮、标签和图标等，当然还需要设置属性，编写若干代码。这种利用窗体设计器设计对话框的方法比较复杂且浪费时间。为此，VB 提供了 MsgBox 函数(语句)，专门用来生成消息对话框(简称消息框)，它可以向用户传送信息，并可以通过用户在对话框中选择的按钮识别用户所作的响应，作为程序继续执行的依据。使用 MsgBox 函数，仅用一行代码，既可快速生成多种形式的消息对话框。

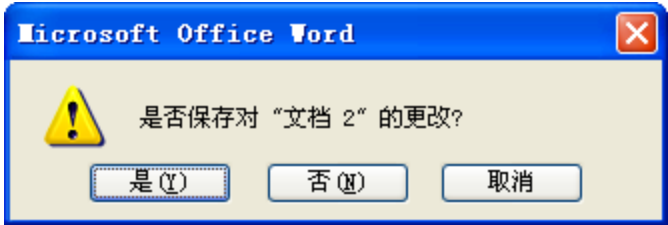


图 5.2 消息对话框

5.2.1 MsgBox 函数

该函数能够在对话框中显示信息，等待用户选择按钮，并返回一个整数指明用户单击了哪个按钮。语法格式如下：

变量 = MsgBox (提示信息[, 按钮] [, 对话框标题])

参数说明：

【提示信息】和【对话框标题】参数的作用与 InputBox 函数的对应参数相同。

【按钮】参数为数值表达式，是可选项，用来控制在对话框内显示的按钮种类和数量以及图标类型。该参数的值由四类数值相加产生，这四类数值分别表示按钮的类型、显示图标的种类、默认按钮的位置和消息框的强制返回特征，如表 5.1 所示。

表 5.1 “按钮”参数的设置值及意义

分类	VB 符号常量	按钮值	含义
按钮类型	vbOkOnly	0	只显示【确定】按钮
	vbOkCancel	1	显示【确定】和【取消】按钮
	vbAbortRetryIgnore	2	显示【终止】、【重试】和【忽略】按钮
	vbYesNoCancel	3	显示【是】、【否】和【取消】按钮
	vbYesNo	4	显示【是】和【否】按钮
	vbRetryCancel	5	显示【重试】和【取消】按钮
图标类型	vbCritical	16	显示 Critical Message 图标
	vbQuestion	32	显示 Warning Query 图标
	vbExclamation	48	显示 Warning Message 图标
	vbInformation	64	显示 Information Message 图标
默认按钮	vbDefaultButton1	0	第一个按钮是默认值
	vbDefaultButton2	256	第二个按钮是默认值
	vbDefaultButton3	512	第三个按钮是默认值
强制返回类型	vbApplicationModal	0	应用模式。用户响应消息框之前，当前应用程序处于暂停状态
	vbSystemModal	4096	系统模式。用户响应消息框之前，所有应用程序处于暂停状态

【按钮】参数由上面四类数值相加组成，其组成原则是：从每一类中选择一个值，把这几个值加在一起就是该参数的值。若省略某类数值，则默认该类数值为 0。不同的组合会得到不同的结果，如果省略【按钮】参数，则只显示【确定】按钮且无图标。例如，【按钮】参数被设为以下三种组合时，将显示图 5.3 所示的效果。

(1) $0+16+0+0=16$ ：显示【确定】按钮和严重警告图标，默认按钮为【确定】按钮，如图 5.3 所示。

(2) $3+32+256+0=291$ ：显示【是】、【否】、【取消】三个按钮以及询问图标，默认按钮为【是】按钮，如图 5.3 所示。

(3) $2+48+0+0=50$ ：显示【终止】、【重试】、【忽略】三个按钮以及警告错误图标，默认按钮为【终止】按钮，如图 5.3 所示。

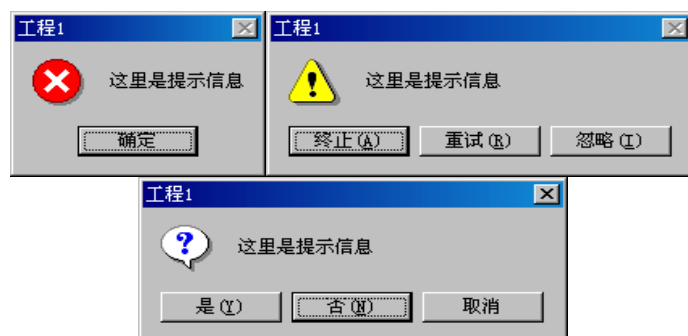


图 5.3 MsgBox 函数【按钮】参数示例

【按钮】参数若采用 VB 符号常量来表示则更加直观。例如：

`vbYesNo + vbQuestion`

MsgBox 函数可以通过返回值判断用户选择了哪一个按钮，对应情况如表 5.2 所示。该返回值用来作为程序继续执行的依据，通常用选择结构根据返回值决定后面的操作。

表 5.2 MsgBox 函数的返回值

符号常量	返回值	含义
vbOk	1	选择【确定】按钮
vbCancel	2	选择【取消】按钮
vbAbort	3	选择【终止】按钮
vbRetry	4	选择【重试】按钮
vbIgnore	5	选择【忽略】按钮
vbYes	6	选择【是】按钮
vbNo	7	选择【否】按钮

【例 5.2】用 MsgBox 函数建立如图 5.4 所示的“退出”对话框。

在例 5.1 的窗体中增加一个命令按钮，设 Caption 属性为“退出”。在该按钮的单击事件中用 Unload 语句卸载窗体：

Unload Me

为窗体的 Unload 事件编写如下代码：

```
Private Sub Form_Unload(Cancel As Integer)
    Dim MyExit As Integer
    MyExit = MsgBox("确实想退出吗？", _
        vbYesNo + vbQuestion, _
        "退出") ' 默认按钮为【否】
    If MyExit = vbNo Then
        Cancel = 1 ' 将 Cancel 参数设为非零值可取消卸载
    End If
End Sub
```

程序运行后单击【退出】按钮或窗体右上角的关闭按钮时，将会弹出图 5.4 所示的对话框，此时若单击【是】按钮则完成窗体卸载，单击【否】按钮则取消卸载，返回主窗体。



图 5.4 【退出】对话框

5.2.2 MsgBox 语句

MsgBox 函数也可以写成语句形式，即：

MsgBox 提示信息[, 按钮类型][, 对话框标题]
其中各参数的含义及作用与 MsgBox 函数相同。
MsgBox 语句和 MsgBox 函数实现的功能基本相同, 只是没有返回值, 因而通常是在只需输出信息而不需用户响应的情况下使用。
【建议】本部分内容相对比较简单, 对于基础比较好的班级可以让学生自学。

5.3 通用对话框

VB 的通用对话框控件 CommonDialog 提供了一组标准对话框界面, 一个控件即可显示六种对话框: 打开文件、保存文件、选择颜色、选择字体、设置打印机以及帮助对话框。这些对话框仅用于返回用户输入、选择或确认的信息, 不能真正实现文件打开和存储以及颜色设置、字体设置等操作。这些功能必须通过编写相应的代码才能实现。

CommonDialog 控件是 ActiveX 控件, 标准工具箱中没有该控件, 使用时需要将其添加到工具箱。添加的方法是: 选择【工程】菜单中的【部件】命令, 或者右击工具箱, 在快捷菜单中选择【部件】命令, 打开如第 1 章中图 1.10 所示的【部件】对话框, 在【控件】选项卡的列表中, 将 Microsoft Common Dialog Control 6.0 前面的复选框选中, 单击【确定】按钮。该控件属性非可视控件, 设计时它以图标形式显示在窗体上, 其大小不能改变, 位置任意, 程序运行时控件本身被隐藏。当需要在程序中显示通用对话框时, 推荐使用 ShowXX 方法 (XX 表示对话框类型), 也可以为该控件的 Action 属性赋值 (可读性差)。调用方法与设置属性的对应关系如表 5.3 所示。

表 5.3 ShowXX 方法与 Action 属性

方法	Action 属性值	功能
ShowOpen	1	显示“打开”对话框
ShowSave	2	显示“另存为”对话框
ShowColor	3	显示“颜色”对话框
ShowFont	4	显示“字体”对话框
ShowPrinter	5	显示“打印”对话框
ShowHelp	6	显示“帮助”对话框

除了 Action 属性外, 通用对话框还具有以下主要的共同属性:

(1) CancelError 属性

通用对话框内有一个【取消】按钮, 用于向程序表示用户想取消当前的操作。当 CancelError 属性设置为 True 时, 若用户单击【取消】按钮, 通用对话框自动将错误对象 (Err, 由 VB 提供) 的错误号 Err.Number 设置为 32755 (VB 常数为 cdlCancel) 供程序判断, 以便进行相应的处理。若 CancelError 属性设置为 False, 则单击【取消】按钮时不产生错误信息, 无法判断用户是否单击了【取消】按钮。

(2) DialogeTitle 属性

该属性可由用户自行设置对话框标题栏上显示的内容，代替默认的对话框标题。

(3) Flags 属性

该属性用于设置对话框的相关选项(各种具体对话框设置的选项略有不同)。

5.3.1 文件对话框

文件对话框用于获取文件名，有两种类型：“打开”和“另存为”对话框。在这两种对话框窗口内，可以遍历磁盘的整个目录结构，找到所需文件，并返回用户选择或输入的文件名。图 5.5 为【打开】对话框，【另存为】对话框与其相似，只是标题和按钮不同。

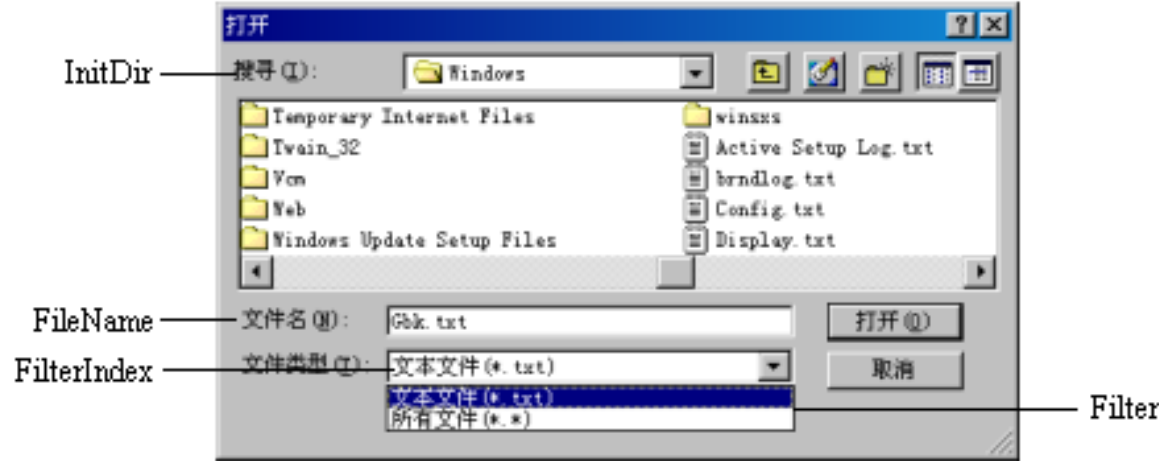


图 5.5 【打开】对话框

使用【打开】和【另存为】对话框时需要设置的属性主要有以下几种。

(1) FileName: 值为字符串，用于设置或获取用户所选的文件名(包括路径)。

(2) FileTitle: 文件标题。设计时无效，运行时只读，返回不包含路径的文件名。

(3) Filter: 过滤器。用于过滤文件类型，使文件列表框中只显示指定文件类型的文件。该属性的设置格式如下(其中竖线 | 是必须要有的语法成分)：

文件说明 1 | 文件类型 1 [| 文件说明 2 | 文件类型 2 ...]

例如，图 5.5 【文件类型】下拉列表中有两种文件类型，其 Filter 属性设置为：

文本文件(*.txt) | *.txt | 所有文件(*.*) | *.*

(4) FilterIndex: 过滤器索引。可指定【文件类型】列表框中的默认过滤器。当使用 Filte 属性指定了多个过滤器时，第一个过滤器的索引值为 1，第二个过滤器的索引值为 2，依次类推。索引值 0 与 1 等价。图 5.5 中 FilterIndex=0，默认显示的是“文本文件 (*.txt)”。

(5) InitDir: 初始化路径。用来指定文件对话框中的初始目录。若显示当前目录，则该属性无须设置。

(6) DefaultExt: 用于【另存为】对话框，它表示所存文件的默认扩展名。

在上述属性中，除 FileTitle 属性外，其他属性均可在属性窗口和代码中设置。此外，包括通用对话框控件在内的大多数 ActiveX 控件都有一种称为“属性页”的属性设置方式，可以快速设置与控件功能有关的特殊属性。右击窗体上的

通用对话框控件，选择快捷菜单中的【属性】命令，即可打开如图 5.6 所示的【属性页】，对各种对话框的特殊属性进行设置。

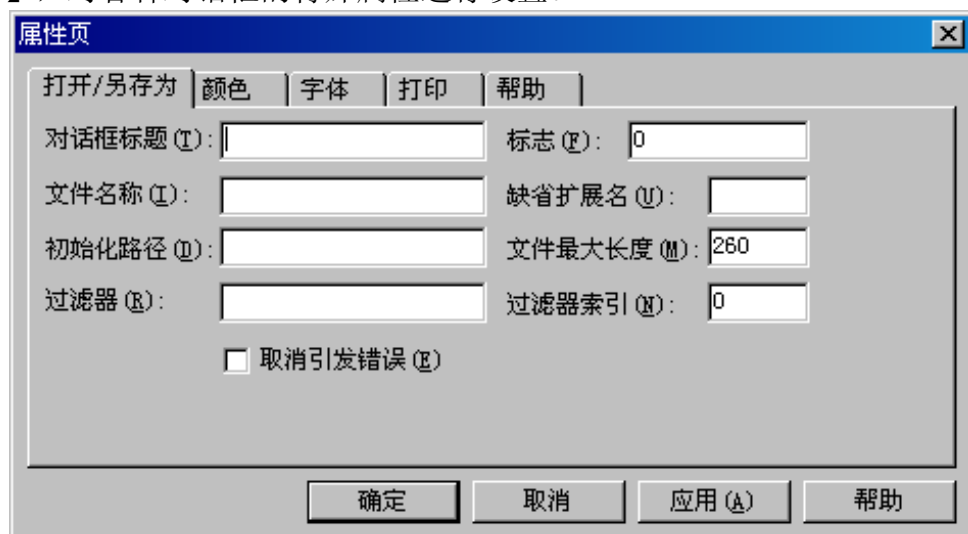


图 5.6 CommonDialog 控件【属性页】对话框

【例 5.3】用命令按钮的单击事件显示【打开】对话框，在对话框内只显示位图文件，初始目录为 C:\My Documents。当在对话框中选定一个位图文件后，单击【打开】按钮则在标签上显示所选的文件名，若单击【取消】按钮，则显示“取消操作”。代码如下：

```
Private Sub Command1_Click()
    On Error GoTo ErrCancel ' 设置出错处理语句
    With CommonDialog1
        .InitDir = "C:\My Documents" ' 设置初始目录
        .Filter = "位图文件(*.Bmp)|*.bmp" ' 过滤文件类型
        .CancelError = True ' 控制取消按钮
        .ShowOpen ' 显示【打开】对话框
        Label1.Caption = .FileName ' 显示选择的文件名
    End With
    Exit Sub ' 正常退出本过程
ErrCancel: ' 以下为错误处理程序段
    If Err.Number = cdlCancel Then ' 用户单击了【取消】按钮
        Label1.Caption = "取消操作"
    End If
End Sub
```

如果将上述代码中的 ShowOpen 改为 ShowSave 即可显示【另存为】对话框。在例 5.3 的代码中，第一次出现 On Error 语句，在此对它作简要说明。On Error 语句有多种语法格式，这里使用的是其格式之一。

On Error GoTo 标号

该语句的作用是当程序发生错误时，跳转到“标号”处继续执行。在例 5.3 中，为了防止用户单击【取消】按钮时仍在标签上显示所选的文件名，所以将对话框的 CancelError 属性设为 True，即故意引发错误，以便使程序转到标号

“ErrCancel:”处继续执行。VB 规定，标号必须以字母形状，以冒号结尾。当使用标号引导一段错误处理代码时，应在标号之前加入 Exit Sub 语句，以防止程序未出错时也执行错误处理代码。

5.3.2 【颜色】对话框

【颜色】对话框用于获取用户选择或设置的颜色。调用通用对话框的 ShowColor 方法时，显示如图 5.7 所示的【颜色】对话框。在对话框的调色板中提供了 48 种基本颜色供选择，还提供了自定义颜色供用户自己调色。

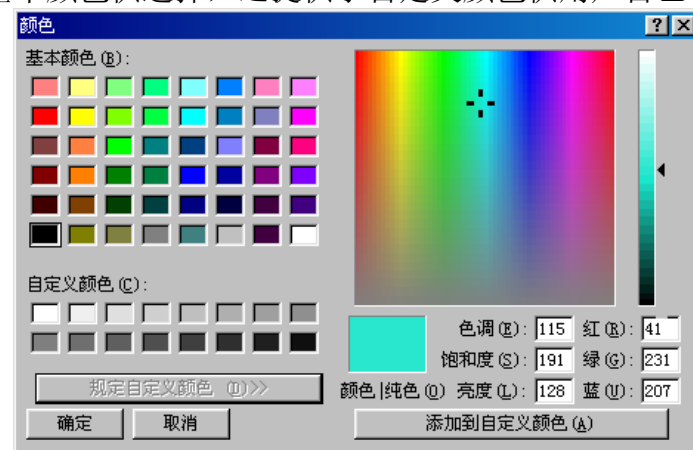


图 5.7 【颜色】对话框

Color 属性是【颜色】对话框最重要的属性，它设置或返回选定的颜色。该属性为长整型数据，有效范围为 0~&HFFFFFF (16,777,215)。当用户在调色板中选中某种颜色时，系统将该颜色值赋给 Color 属性。在代码中可利用该属性为其他对象的颜色属性赋值。例如，下面的代码可以将用户在【颜色】对话框中选定的颜色设置为文本框的背景色，并将文本框的前景色设为背景色的互补色。

```
CommonDialog1.ShowColor
```

```
Text1.BackColor = CommonDialog1.Color
```

```
Text1.ForeColor = &HFFFFFF - CommonDialog1.Color
```

注：用十六进制数&HFFFFFF 减去某个颜色值即为该颜色的互补色值。

5.3.3 【字体】对话框

【字体】对话框供用户选择字体，可获取用户所选字体的名称、样式、大小及效果。调用通用对话框的 ShowFont 方法时，显示如图 5.8 所示的【字体】对话框。

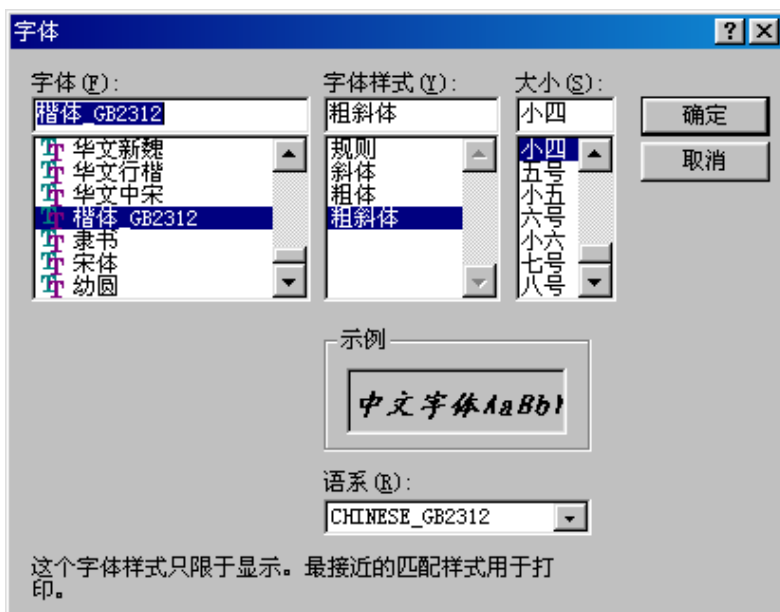


图 5.8 【字体】对话框

在使用 CommonDialog 控件选择字体之前，必须设置 Flags 属性值。该属性控制 CommonDialog 控件是否显示屏幕字体、打印机字体或者两者皆有。如果未设置 Flags 属性值而直接使用该控件打开【字体】对话框，VB 将显示图 5.9 所示的错误提示。

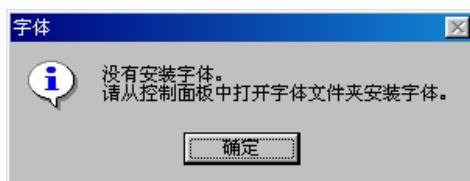


图 5.9 未设置 Flags 属性值的错误提示

通用对话框用于字体操作时涉及到的重要属性有以下几种。

(1) Flags 属性

在【字体】对话框中常用的 Flags 属性设置值如表 5.4 所示。其中，前三项必须选择其一才能防止图 5.9 所示的错误。

表 5.4 字体对话框 Flags 属性设置值

常数	值	说明
cdlCFScreenFonts	1	屏幕字体
cdlCFPrinterFonts	2	打印机字体
cdlCFBoth	3	屏幕字体和打印机字体两者皆有
cdlCFEffects	256	对话框中出现下划线、删除线、颜色元素

(2) Font 属性集

Font 属性集包括 FontName (字体名)、FontSize (字号)、FontBold (粗体)、FontItalic (斜体)、FontStrikethru (删除线) 和 FontUnderline (下划线)。

(3) Color 属性

Color 属性用于设置字体颜色。要使用该属性必须使 Flags 属性含有 cdlCFEffects 值。

【例 5.4】用【字体】对话框设置文本框的字体，要求字体对话框内出现【效果】选项（下划线、删除线和颜色）。

在窗体上放置通用对话框，文本框和命令按钮。为按钮单击事件编写以下代码：

```
Private Sub Command1_Click()  
    With CommonDialog1  
        .Flags = cdlCFBoth Or cdlCFEffects ' 设置 Flags  
        .FontName = "宋体"                ' 设置对话框默认字体  
        .ShowFont                          ' 显示字体对话框  
        Text1.FontName = .FontName         ' 设置文本框字体名  
        Text1.FontSize = .FontSize         ' 设置字体大小  
        Text1.FontBold = .FontBold         ' 设置粗体  
        Text1.FontItalic = .FontItalic     ' 设置斜体  
        Text1.FontStrikethru = .FontStrikethru ' 设置删除线  
        Text1.FontUnderline = .FontUnderline ' 设置下划线  
        Text1.ForeColor = .Color           ' 设置颜色  
    End With  
End Sub
```

当 Flags=cdlCFBoth Or cdlCFEffects 时，对话框如图 5.10 所示，与图 5.8 相比增加了【效果】选项。也可以用 Flags=259 表示该设置（256+3=259）

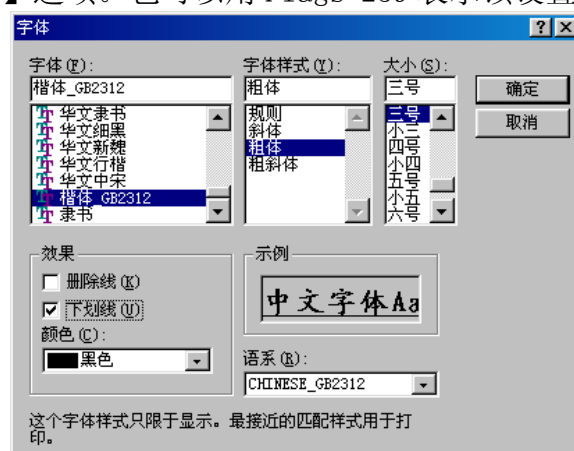


图 5.10 用 Flags 属性加入【效果】选项

5.3.4 【打印】对话框

【打印】对话框如图 5.11 所示，设计时可通过图 5.12 所示的【属性页】设置其属性。运行时该对话框供用户选择打印机，设置打印参数（如打印范围、份数等）。通过对话框中的【属性】按钮可设置打印机的属性。【打印】对话框并不能处理打印工作，只是一个供用户选择或设置打印参数的界面，所设参数存于各属性中供编程使用。若要打印必须为 Printer 对象（表示所安装的默认打印机）编写程序来实现。



图 5.11 【打印】对话框

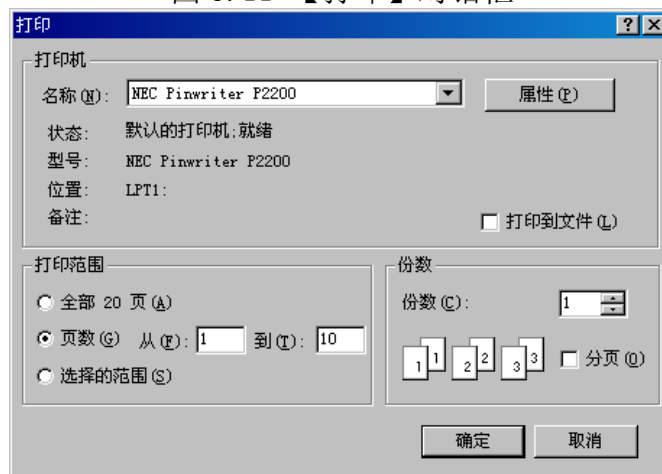


图 5.12 【打印】对话框属性

通用对话框用于打印操作时涉及到的重要属性主要有以下几种。

- (1) **【副本】** (Copies)：指定打印份数。
- (2) **【起始页】** (FromPage)、**【终止页】** (ToPage)：打印的起始页号和终止页号。
- (3) **【最小值】** (Min)、**【最大值】** (Max)：打印的最小页数和最大页数。
- (4) **【方向】** (Orientation)：打印方向。cdlPortrait 为纵向；cdlLandscape 为横向。

【例 5.5】 在例 5.4 中增加一个命令按钮，调用**【打印】**对话框，打印文本框中的内容。

调用 Printer 对象的 Print 方法将要打印的内容发送到打印机即可实现打印。调用 Printer 对象的 EndDoc 方法可结束打印操作。代码如下：

```
Private Sub Command2_Click()
    Dim i As Integer
    CommonDialog1.ShowPrinter          ' 显示“打印”对话框
    For i = 1 to CommonDialog1.Copies ' 按份数打印
        Printer.Print Text1.Text      ' 打印文本框中的内容
    Next
```

```
Printer.EndDoc  
End Sub
```

’ 结束文档打印

5.3.5 【帮助】对话框

CommonDialog 控件的 ShowHelp 方法可调用 Windows 的帮助引擎，并显示由 HelpFile 属性设定的一个帮助文件。

【帮助】对话框涉及到的重要属性有：

- (1) 【帮助文件】 (HelpFile)：用于指定帮助文件的路径及其文件名称。
- (2) 【帮助命令】 (HelpCommand)：用于返回或设置所需要的联机帮助的类型。

【例 5.6】用【帮助】对话框调用 Windows【画图】程序的帮助文件 Mspaint.hlp，显示图 5.13 所示的对话框。

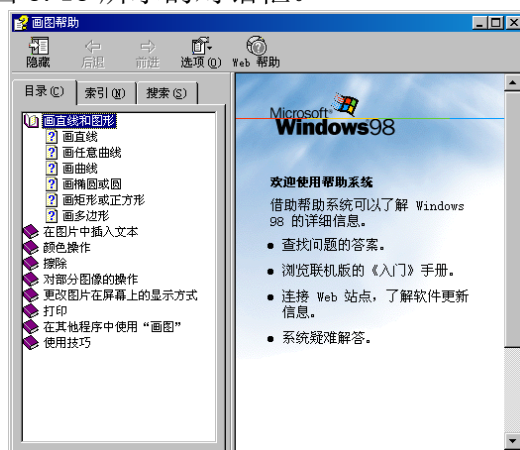


图 5.13 画图帮助窗口

在窗体上放置一个通用对话框，一个命令按钮。为按钮的单击事件编写以下代码：

```
Private Sub Command1_Click()  
    CommonDialog1.HelpCommand = cdlHelpContents  
    CommonDialog1.HelpFile = _  
        "C:\Windows\Help\Mspaint.hlp"  
    CommonDialog1.ShowHelp  
End Sub
```

【练习】编写一个程序，上面有六个按钮，按下每个按钮分别打开每一种对话框。

5.4 自定义对话框

自定义对话框是根据实际应用的需要设计的对话框。当 VB 所提供的通用对话框控件以及 InputBox 和 MsgBox 函数不能满足应用程序的需求时，就需要自制对话框。自制对话框实际上是一个含有若干控件的窗体，用以构成用户与系统对话的界面，通常将窗体的 BorderStyle 属性设为 3-Fixed Dialog 或 1-Fixed

Single, 使其无最大化和最小化按钮, 不能改变大小。在第 2 章“2.3.6 多窗体应用程序”小节内, 例 2.6 中的【关于】窗体就是一个自制对话框。

复习一下前面介绍的 MsgBox 语句。

【例 5.7】创建一个用户登录对话框, 要求用户输入用户名和密码。用户名为 admin, 不区分大小写; 密码为“12345”。若用户名和密码输入正确, 单击【确定】按钮后显示程序主窗体, 否则提示用户重新输入。若错误超过三次, 结束运行。

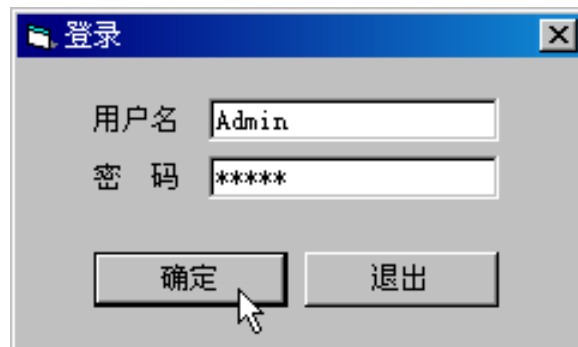


图 5.14 登录对话框

(1) 设计界面及设置属性

新建工程, 将窗体默认名称 Form1 改为 frmLogin, 设 BorderStyle 属性为 3, Caption 属性为“登录”。在窗体上添加两个文本框, 名称分别为 txtUser 和 txtPassword, Text 属性均设为空。设 txtPassword 文本框的 Password 属性为“*”。添加两个标签, Caption 属性分别为【用户名】和【用户密码】。添加两个命令按钮, 名称分别为 cmdOk 和 cmdExit, Caption 分别为【确定】、【退出】。设【确定】按钮 Default=True, 【退出】按钮 Cancel=True。

添加一个窗体, 名称为 frmMain, 设 Caption 属性为【主窗体】。在窗体上添加一个标签, 设 Caption 属性为【欢迎进入本系统】, 字体为【华文新魏】, 【二号】字。

(2) 编写代码

为 frmLogin 窗体的【确定】按钮的单击事件编写以下代码, 进行用户登录检测:

```
Private Sub cmdOk_Click()  
    Static intErr As Integer ' 静态变量累加出错次数  
    Dim sUser As String  
    Dim sPass As String  
    sUser = UCase$(Trim$(txtUser.Text)) ' 用户名不区分大小写  
    sPass = Trim$(txtPassword.Text)  
    ' 检查用户名和密码  
    If sUser = "ADMIN" And sPass = "12345" Then ' 若正确  
        frmMain.Show ' 显示主窗体  
        Unload Me ' 卸载本窗体  
    Else ' 若错误  
        intErr = intErr + 1 ' 错误次数+1  
        If intErr > 3 Then ' 若出错超过 3 次, 提示后退出  
            MsgBox "对不起, 您不是本系统的合法用户。", _
```

```

        vbInformation, "提示"
    End
Else ' 若出错不超过 3 次, 用 MsgBox 语句提示重新输入
    MsgBox "用户名或密码错误, 请重新输入!", _
        vbExclamation, "提示"
        ' 焦点返回出错的文本框并全选其内容。
        ' VB 函数 Len 可返回字符串长度。
If sUser <> "ADMIN" Then ' 若用户名错
    txtUser.SelStart = 0
    txtUser.SelLength = Len(txtUser.Text)
    txtUser.SetFocus
Else ' 若密码错
    txtPassword.SelStart = 0
    txtPassword.SelLength = Len(txtPassword.Text)
    txtPassword.SetFocus
End If
End If
End If
End Sub

```

在 frmLogin 窗体【退出】按钮的单击事件中用 End 语句结束运行。

在本例中的主窗体 frmMain 只是为了配合登录对话框而设, 因此界面设计比较简单, 没有编写编码。

程序运行结果如图 5.14、5.15 和 5.16 所示。

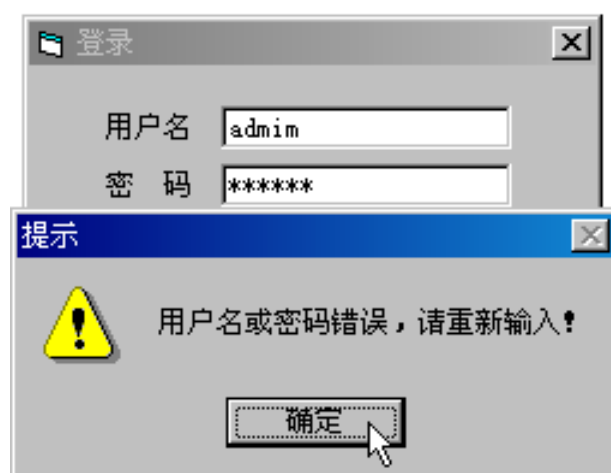


图 5.15 用户名或密码错误

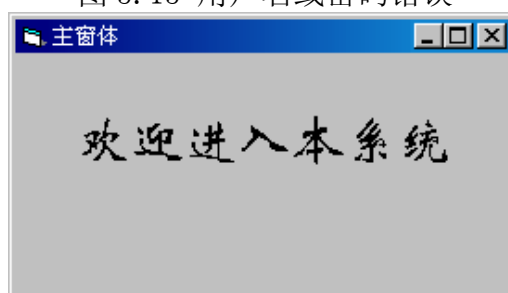


图 5.16 登录成功

以上介绍了如何设计一个简单的自定义对话框。有了这个基础，在学习其他控件以后，就可以设计出更复杂的对话框。

1. 如何建立一个输入对话框？如何确定输入对话框的位置？
2. 在消息对话框中如何设置要显示的信息？如何确定其按钮的类型？如何判断用户在消息对话框中单击了哪个按钮？
3. 如何在程序中显示通用对话框？如何自行设置通用对话框的标题？
4. 如何在【打开】对话框内过滤文件类型？怎样在【另存为】对话框中传送文件名？
5. 在使用【字体】对话框之前必须设置什么属性？要在【字体】对话框显示【效果】选项应设置什么属性，如何设置？
6. 修改例 5.7，将用户提示重新输入用户名和密码的语句改用 MsgBox 函数，在消息对话框中显示【重试】和【取消】按钮。单击【重试】按钮重新输入，否则退出。

【练习】修改例 5.7，把例 5.6 与 5.7 结合起来，如果用户名与密码正确，打开例 5.6 窗体。

【思考】如何使我们的登录界面的密码更安全？

第 6 章 常用内部控件

本章要点

- 使用选择类控件单选按钮和复选框为用户提供选项
- 使用框架为控件分组
- 使用列表类控件列表框和组合框为用户提供选项
- 图像显示控件图片框和图像框的使用
- 定时器和滚动条的使用

为了方便应用程序的开发，VB 中提供了很多的内部控件（或称标准控件），如前面介绍的命令按钮、标签、文本框等。内部控件是包含在 Visual Basic 系统内、可供用户直接使用的控件，因此具有较好的运行性能。

本章介绍单选按钮、复选框、框架、列表框、组合框、图片框、图像框、定时器、滚动条这几种内部控件及其使用方法。控件的某些属性，如 Name 属性、Caption 属性、Font 属性、Enabled 属性等，以及某些方法如 Move 等，均为多数控件所共有，其意义和用法与窗体、标签、文本框控件的对应属性相同，这里不再赘述。

6.1 选择类控件与框架

单选按钮(OptionButton)和复选框(CheckBox)都具有选择作用,称之为选择类控件。框架(Frame)是一个容器,可以在其上放置其他控件对象,能够把一些控件组织在一起形成控件组。

6.1.1 单选按钮

单选按钮(OptionButton)又称单选钮,它的作用是显示一个可以表示“打开/关闭”的选项,使用户在多个选项中只能选择其一。

例如学生性别的输入,代表性别的“男”、“女”是相互排斥的,故可以使用两个单选按钮实现,如图 6.1 所示。



图 6.1 单选按钮

1. 常用属性

(1) Value 属性

单选按钮的属性,除了 Caption、Enabled、Visible、Font、ForeColor、BackColor 等外,主要是 Value 属性。该属性表示单选按钮被选中(True)或不被选中的状态(False)。

在设计阶段,设置单选按钮的 Value 属性为 True 表示选中,为 False 表示不被选中;在程序运行时,单击单选按钮,使其单选框中出现一个黑色圆点,就表示选中了该项。也可以通过将 Value 高为 True,使单选按钮被选中。

因此,程序中可以通过单选按钮的 Click 事件过程,进行选中后的某些处理。例如,若单选按钮名称为 Option1,则格式如下:

```
Private Sub Option1_Click()  
    '此时单选按钮 Option1 已经被选中  
    .....  
End Sub
```

程序中也可以通过判断单选按钮的 Value 属性的值,来确定是否被选中,进而执行相应的操作,格式如下:

```
If Option1.Value=True Then  
    '单选按钮 Option1 已经被选中  
    .....  
Else  
    '单选按钮 Option1 未被选中  
    .....  
End If
```

Value 属性是单选按钮控件的默认属性(或称控件值)。所有控件都有一个属性,只需引用控件名而无需使用属性名即可访问这个属性,此属性被称为控件的默认属性。例如,Option1.Value = True 与 Option1 = True 等效。其他常用控件如文本框控件的默认属性为 Text,标签控件的默认属性为 Caption。使用

默认属性时，代码的可读性略受影响，所以在不引起代码阅读困难时方可考虑使用默认属性。

(2) Style 属性

单选按钮的 Style 属性用来设置控件的外观。当值为 0 时，控件显示如图 6.1 所示的标准样式；当值为 1 时，控件显示如图 6.2 所示的图形样式，其外观类似于命令按钮。



图 6.2 单选按钮的样式 (Style=1)

(3) Picture、DownPicture 和 DisabledPicture 属性

当 Style 属性为 1 时，这三个属性有效，从而使单选按钮的外观更加形象直观。其中：Picture 属性返回或设置控件中要显示的图像；DownPicture 属性返回或设置控件被选中后（即单击后）要显示的图像；DisabledPicture 属性返回或设置控件无效时显示的图像，即控件的 Enabled 属性为 False 时控件的外观图像。三个属性可以在设计阶段通过【属性】窗口直接设置为某个图像文件，也可以在运行期间由函数 LoadPicture 加载。

在图 6.3 中，单选按钮的 Style 已经设置为 1，左图表示设置了 Picture 属性的情况，而右图表示同时设置了 DownPicture 属性的情况。



图 6.3 单选按钮的 Picture 和 DownPicture 属性

【提问】在我们用过的软件中那些按钮是属于单选按钮。

2. 常用事件

单选按钮可以识别的主要事件是单击 (Click) 事件。

【例 6.1】控制文本框中文本的字体变化。字体可以使用“宋体”、“隶书”和“幼圆”三者之一。

本例通过三个单选按钮选择字体名称，属性设置见表 6.1。其中，将文本框 Text1 的 Multiline 属性设为 True 的目的是使其允许多行显示。此时，在属性窗口设置文本框的 Text 属性时，须通过组合键 Ctrl+回车来分行输入文本内容。

表 6.1 例 6.1 对象和属性设置

对象(名称)	属性	属性值	对象(名称)	属性	属性值
窗体 (Form1)	Caption	单选按钮示例	文本框 (Text1)	Multiline	True
单选按钮 (optFont1)	Caption	宋体		Text	单选按钮示例 (Ctrl+回车键) 在此输入内容，选择字体后 (Ctrl+回车键) 单击 【应用】 按钮
单选按钮 (optFont2)	Caption	隶书			
单选按钮 (optFont3)	Caption	幼圆			
命令按钮 (cmdOk)	Caption	应用			

程序的运行结果如图 6.4，代码如下：

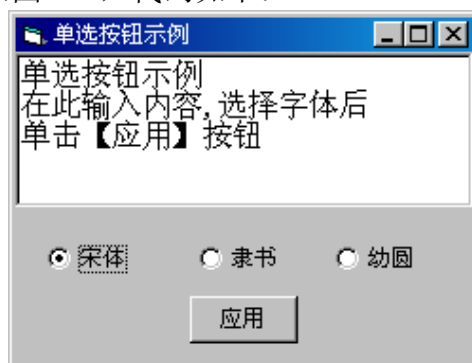


图 6.4 例 6.1 的运行结果

```
Private Sub cmdOk_Click() '单击【应用】按钮
    If optFont1 = True Then Text1.FontName = "宋体"
    If optFont2 = True Then Text1.FontName = "隶书"
    If optFont3 = True Then Text1.FontName = "幼圆"
End Sub
Private Sub Form_Load() '窗体加载
    Text1.FontName = "宋体"
    Text1.FontSize = 12
    optFont1 = True
End Sub
```

6.1.2 复选框

复选框 (CheckBox) 也称为选择框、检查框，通常用于提供 Yes/No 或 True/False 的逻辑选择。一个复选框主要有两种状态：选中状态，或称打开状态，复选框上出现“√”标志；未选中状态，或称关闭状态，不出现“√”标志。

复选框的属性和单选按钮的属性基本类似。其主要属性是 Value 属性，Value 属性指示其所处的状态：0 表示没有选中，1 表示该项选中，2 表示禁止使用。

复选框可以识别的主要事件是单击 (Click) 事件。程序运行中，当用户单击复选框时将触发其 Click 事件，每单击一次其状态就在“没有选中”和“选中”之间变换一次，相应地，其 Value 属性的值在 0 和 1 之间变换。因此，当发生了 Click 事件时，程序要判断 Value 属性的值，以便确定是否选中。

【对比】对比单选按钮和复选框的区别。

复选框与单选按钮都可表示一种状态，因此两者有相似之处，但有本质的区别：一组复选框中的多个项目是相互“兼容”的，一组单选按钮中的多个项目却是相互“排斥”的。比如，一个人的性别（男、女）。可以作为单选按钮中的项目；而一个人的年龄、身高、籍贯可以作为复选框中的项目，因此它们并不相互排斥。

【例 6.2】用复选框控制文本是否加下划线和斜体显示。

在本例程序执行过程中，如果选中了【下划线】复选框，则文本框中的内容就加上了下划线；如果清除【下划线】复选框，则文本框中的内容就没有下划线。

如果选定【斜体】复选框，则文本框中的文字字形就变成斜体；如果清除【斜体】复选框，则文本框中的文字字形就不是斜体。

在窗体上添加一个文本框，两个复选框。属性设置如表 6.2，运行界面如图 6.5 所示。

表 6.2 例题 6-2 的对象和属性设置

对象（名称）	属性	属性值	对象（名称）	属性	属性值
窗体（Form1）	Caption	复选框示例	复选框（Check2）	Caption	斜体
复选框（Check1）	Caption	下划线	文本框（Text1）	Text	下划线和斜体不相互排斥

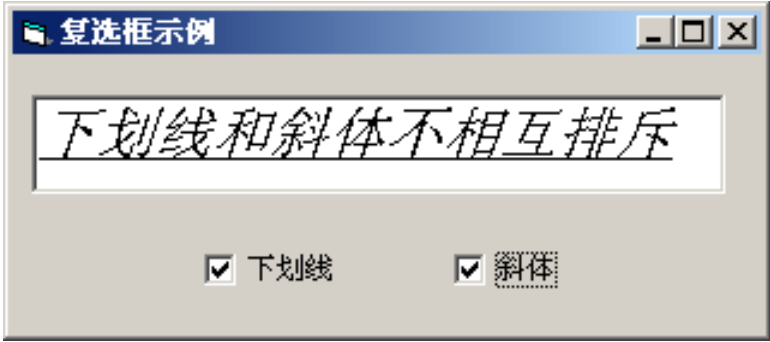


图 6.5 例 6.2 的运行结果

为两个复选框的单击和窗体加载事件编写如下事件过程。

```
Private Sub Check1_Click()          '单击【下划线】复选框
    If Check1.Value = 1 Then
        Text1.FontUnderline = True    '加下划线
    Else
        Text1.FontUnderline = False
    End If
End Sub
Private Sub Check2_Click()          '单击【斜体】复选框
    If Check2.Value = 1 Then
        Text1.FontItalic = True        '斜体
    Else
        Text1.FontItalic = False
    End If
End Sub
Private Sub Form_Load()
    Text1.FontSize = 18
End Sub
```

【练习】要求某位同学在上面的例题中加入一个“粗体”复选框。

【例 6.3】用户信息的收集是一类常见的应用程序。本例要求编写程序收集用户选择的专业类别和选修课程。其中，可选择的专业类别有“计算机专业”和“机电专业”，可选的课程有“高等数学”、“大学英语”和“程序设计”。

【分析】本例中专业的所属类别之间具有排斥性，可以用单选按钮实现；而选修课程之间具有兼容性，应该用复选框实现。

属性设置如表 6.3，其中将 Option1 的 Value 属性设为 True，表示初始时默认选中“计算机专业”。

表 6.3 例 6.3 的对象和属性设置

对象 (名称)	属性	属性值	对象 (名称)	属性	属性值
窗体 (Form1)	Caption	信息采集	复选框 (Check1)	Caption	高等数学
单选按钮 (Optfont1)	Caption	计算机专业	复选框 (Check2)	Caption	大学英语
	Value	True	复选框 (Check3)	Caption	程序设计
单选按钮 (optFont2)	Caption	机电专业	标签 (Label1)	Caption	所属类别
命令按钮 (Command1)	Caption	确定	标签 (Label2)	Caption	选修课程

代码如下，其中的 Chr(13)、Chr(10) 是产生回车换行（亦可用 VB 常数 vbCrLf）；为了简化代码，将所收集到的用户选择信息用 MsgBox 函数显示出来。

```
Private Sub Command1_Click()  
    Dim str As String, link As String  
    link = Chr(13) & Chr(10)  
    If Option1.Value = True Then  
        str = "计算机专业"  
    Else  
        str = "机电专业"  
    End If  
    str = str & "选择了：" & link  
    If Check1.Value = 1 Then str = str & link & "高等数学"  
    If Check2.Value = 1 Then str = str & link & "大学英语"  
    If Check3.Value = 1 Then str = str & link & "程序设计"  
    MsgBox str, vbYesNo, "采集信息"  
End Sub
```

单击【确定】按钮后的运行结果如图 6.6。

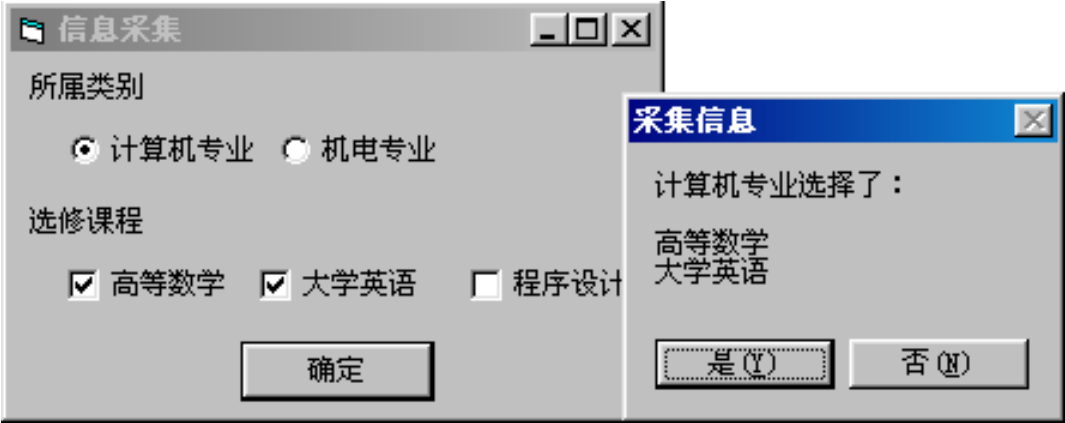


图 6.6 例 6.3 的运行结果

6.1.3 框架

框架 (Frame) 是一个容器，可以在其上放置其它控件对象，主要作用是能够把一些控件组织在一起形成控件组。分组的用途有两个：一是单纯地对其它控件分组，使功能上密切相关的控件在一个框定的区域内，以便用户分类识别；二是用于为单选按钮分组。如前所述，用户在多个单选按钮选项中只能选择其一，

如果要想实现单选按钮的多项选择，就必须使用框架。仅次于同一个框架内的多个单选按钮为一组，用户在同一组内只能选择其一；但在不同的组内却可以选择不同的项目，从而实现多项选择。

【对比】对比使用和不使用框架的区别。

当不使用框架时，窗体上的单选按钮均视为在同一组。在图 6.7 中，左图未使用框架，四个单选按钮中只能有一个被选择；右图使用了两个框架 Frame1 和 Frame2，将四个单选按钮分成了两组，一组用于选择性别，另一组用于选择职业，于是用户可以在框架 Frame1 和框架 Frame2 中各选择一项。

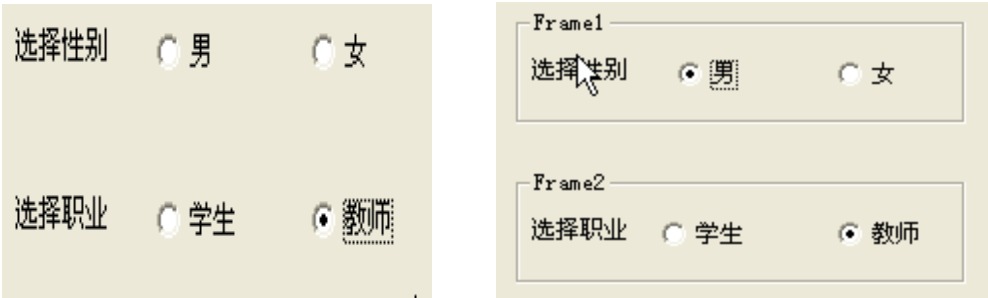


图 6.7 框架的分组效果

【注意】为了实现分组，首先画出框架，然后在框架内画出所需的控件。

如果希望将已经存在的若干控件放在某个框架中，可以先选择这些控件，将它们剪切到剪贴板上，然后选定框架控件并把它们粘贴到框架上。位于一个框架内的控件会随框架整体移动、隐藏、删除。

框架的常用属性有：Caption 属性（设置框架标题，位于框架的左上角）、Enabled 属性（设置框架是否有效）、Visible 属性（设置框架是否可见）。当框架的 Enabled 属性为 False 时，框架和框架内的控件均呈灰色，表示不可使用，相当于整体失效；当框架的 Visible 属性为 False 时，框架及其中的控件均不可见，相当于整体隐藏。

框架支持 Click 和 DbClick 事件，在大多数情况下，主要是用框架控件对控件进行分组，没有必要响应它的事件。

【例 6.4】利用框架的分组功能，同时设置文本框的字体、大小、颜色。

本例使用了三个框架，每个框架内均有三个单选按钮。在一个框架内的三个单选按钮为一组，它们是相互“排斥”的，但三个框架之间是相互“兼容”的。属性设置见表 6.4。其中未设置的属性没有列出。运行结果如图 6.8 所示。

注意：在窗体的 Form_load 进行了若干初始化工作，而命令按钮 cmdNo（标题为“恢复”）也要进行同样的初始化工作，帮在按钮 cmdNo 的单击事件中直接调用 Form_load 过程。

表 6.4 例 6.4 的对象和属性设置

对象 (名称)	属性	属性值	对象 (名称)	属性	属性值
窗体 (Form1)	Caption	框架示例	单选按钮 (optFont1)	Caption	宋体
文本框 (Text1)	Text	(空)	单选按钮 (optFont2)	Caption	隶书
	Multiline	True	单选按钮 (optFont3)	Caption	幼圆
	Scrollbars	2 (垂直滚动条)	单选按钮 (optSize1)	Caption	12
框架 (Frame1)	Caption	字体	单选按钮 (optSize2)	Caption	14
框架 (Frame2)	Caption	大小	单选按钮 (optSize3)	Caption	20
框架 (Frame3)	Caption	颜色	单选按钮 (optColor1)	Caption	蓝色
命令按钮 (cmdOk)	Caption	应用	单选按钮 (optColor2)	Caption	红色
命令按钮 (cmdNo)	Caption	恢复	单选按钮 (optColor3)	Caption	绿色



图 6.8 例 6.4 运行结果

【练习】由某位学生建立程序的界面，注意提醒学生先建立框架。
程序代码如下：

```

Private Sub cmdNo_Click() ' 单击【恢复】按钮
    Form_Load ' 执行 Form_Load 过程
End Sub
Private Sub cmdOk_Click() ' 单击【应用】按钮
    ' 确定字体名
    If optFont1 = True Then Text1.FontName = "宋体"
    If optFont2 = True Then Text1.FontName = "隶书"
    If optFont3 = True Then Text1.FontName = "幼圆"
    ' 确定字体大小
    If optSize1 = True Then Text1.FontSize = 12
    If optSize2 = True Then Text1.FontSize = 14
    If optSize3 = True Then Text1.FontSize = 20
    ' 确定颜色
    If optColor1 = True Then Text1.ForeColor = vbBlue
    If optColor2 = True Then Text1.ForeColor = vbRed
    If optColor3 = True Then Text1.ForeColor = vbGreen

```

```

End Sub
Private Sub Form_Load() ' 窗体加载
    optFont1 = True
    Text1.FontName = "宋体"
    Text1.FontSize = 12
    Text1.ForeColor = vbBlack
End Sub

```

6.2 列表类控件

当需要向用户提供的备选项目太多时（如一个国家的所有省份），若仍采用前面介绍的单选按钮和复选框，在窗体上将很难安排，界面设计和编写代码的工作量之大不堪设想。即使设计出来，也会使用户望而生畏。在这种情况下，可采用列表类控件列表框和组合框。在标准工具箱中还有三种文件系统列表类控件，将在第 13 章介绍。

6.2.1 列表框

1. 列表框的功能

列表框（ListBox）显示由若干项目组成的列表，用户可从中选择一个或多个项目。所选择的项目被突出显示。列表框的大小通常在设计阶段设定，但也可以通过 Width 属性和 Height 属性在程序运行时修改。如果列表框中的项目过多，则系统会自动增加一个垂直滚动条，如图 6.9 所示。

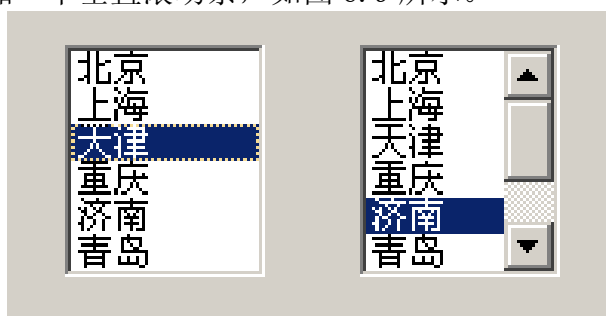


图 6.9 列表框示意图

列表框中的项目可以在设计状态下通过属性窗口设定，也可以在运行状态下由程序加入。前者使用列表框的 List 属性，一个项目为一行，且以组合键 Ctrl + 回车键进行分行，如图 6.10 所示；后者使用列表框的 AddItem 方法。

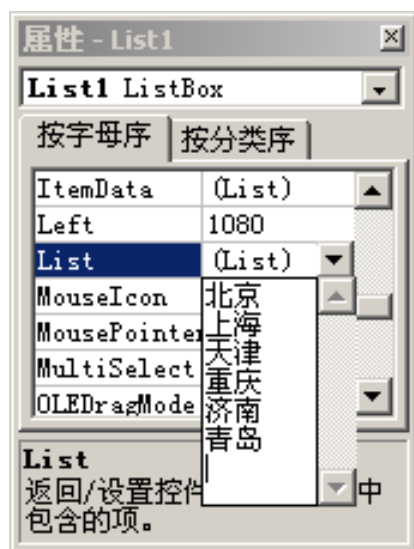


图 6.10 设置列表框的 List 属性

列表框中的项目列表是一个整体，它实际上是一个数组（若干元素的有序集合）。由于数组涉及的内容较多，故将其放在第 7 章专门讨论，在此仅对与列表框有关的内容作简要介绍。从图 6.9 中可以看出，列表框中的每个项目各占一行，所有项目构成项目列表。列表中的每一项（行）都有自己的位置，用“索引号”来表示（在数组中称为下标）。列表中第一项的索引号为 0，第二项为 1，依此类推。利用索引号可以很方便地访问列表中的任何一个项目。

列表框的功能涉及到它的许多属性和方法，可以将其中较常用者分为以下几类。

- (1) 增加和删除项目：List 属性；AddItem、RemoveItem、和 Clear 方法。
- (2) 访问所有项目：List、listCount 属性。
- (3) 获取或设置选定的项目：Text、ListIndex、selected 和 Multiselect 等属性。
- (4) 列表框外观：Columns、Style 和 Sorted 等属性。

下面详细介绍列表框的常用属性、方法和事件。

2. 常用属性

(1) Text 属性

在程序运行过程中，用于获取列表框中当前选择的项目内容。该属性在设计时不可用。

例如，将列表框 List1 中所选择的项目内容放入文本框 Text1 中：

```
Text1.Text = List1.Text
```

(2) ListCount 和 List 属性

ListCount 属性返回列表框中已有项目的总数目，它是一个设计时无效、运行时只读的属性，即在程序运行时，通过该属性可以获取项目总数，但不能直接设置该属性的值，其值的变化是由其他操作自动决定的。语法格式为：

列表框对象.ListCount

List 属性用来访问列表框中的全部项目内容。该属性实际是一个字符串数组，数组中的每个元素对应着列表框中的一个项目。语法格式为：

列表框对象.List(索引号)

其中的参数“索引号”指明数组中的元素下标，即第几个元素，它的取值从 0 开始，到项目数 ListCount-1 止。如果某个列表框含有 5 个项目，则“索引

号”参数的取值范围从 0 到 4。通过指定不同的索引值，可以访问列表的全部项目。例如，将列表框 List1 中的第 3 项复制到文本框 Text1 中：

```
Text1.Text = List1.List(2)
```

又如，将列表框 List1 中的全部项目显示在窗体上。

```
For i=0 To List1.Listcount-1
```

```
Print List1.List(i)
```

```
Next i
```

利用 List 属性还可以改变不负责任框中现有的一个项目的内容，但被改变的项目必须已经存在，否则出错。例如：

```
List1.List(0)="哈尔滨" '将列表框 List1 中的第 1 项设置为"哈尔滨"
```

```
List1.List(3)="昆明" '将列表框 List1 中的第 4 项设置为"昆明"
```

(3) ListIndex 属性

返回当前已选定项目的位置（索引）号。未选定项目时，返回的 ListIndex 值为-1。该属性只在运行时可用。当单击列表框中的一个项目后，项目的索引号（下标）便存储在 ListIndex 属性中。因此，若 ListIndex 值不是-1，则以下语句可显示当前选定的项目：

```
Print List1.List(List1.ListIndex) '与 Print List1.Text 等效
```

反之，若对该属性赋值则可选定某一项目。例如：

```
List1.ListIndex = 0 '选定列表中的第一项
```

(4) Selected 属性

该属性用来设置或返回列表框中某项目的选择状态。Selected 属性也是一个数组，每个数组元素与列表框中的一个项目相对应，用法也和 List 属性类似。不同的是，Selected 属性数组取逻辑值 True、False。若为 True 则表示相应的项目被选择，若为 False 则表示相应的项目没有被选择。

例如，对列表框 List1 中的第 3 项而言，如果单击该项目使之被选定，则 List1.Selected(2) 的值就会等于 True；如果执行语句 List1.Selected(2) = True，则相当于选择第 3 项，与 List1.ListIndex = 2 等效。

(5) Sorted 和 Style 属性

Sorted 属性确定列表框中的项目是否排序。其值设置为 False（默认）时项目不排序，若为 True 则项目按照字母升序排列（不区分大小写）。

Style 属性确定列表框的样式。取值为 0（默认值）和 1，如图 6.11 所示。这两个属性只能在设计时设置。



图 6.11 列表框的样式

(6) Columns 属性

使用 Columns 属性可以创建多列列表框。

默认情况下，列表框是一种单列列表框，我们通常使用的也是单列列表框，此时 Columns=0，并具有垂直滚动条。当希望使用多列列表框时，便可以设置 Columns 为大于 0 的值，表示具有若干列和水平滚动条。Columns=0 时和 Columns=2 时的列表框如图 6.12 所示。

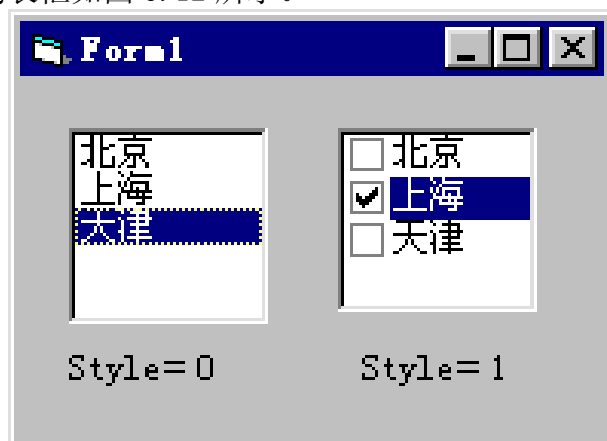


图 6.12 列表框的 Columns 属性

(7) MultiSelect 属性

MultiSelect 属性的默认取值为 0，表示列表框是单选列表框，一次只能选择一项。若将 MultiSelect 属性值设为 1 或 2，则表示列表框是复选列表框，即可以在列表框的列表中选择多个项目。值为 1 时，用鼠标单击或按空格键进行复选；为 2 时类似于【资源管理器】，可用 Shift+鼠标单击（连续多选）、Ctrl+鼠标单击（不连续多选）等来进行复选。

只能进行单选的列表框，可以通过 ListIndex 属性或 Selected 属性判断所选择的项目。允许进行复选的列表框，所选择的项目可能有多项，故不能通过 ListIndex 属性判断，一般是通过 Selected 属性判断。例如，以下代码可以显示出所有被选择的项目：

```
For i = 0 To List1.ListCount - 1
    If List1.Selected(i) = True Then Print List1.List(i)
Next i
```

(8) NewIndex 属性

返回最后加入列表框的项目的索引号。该属性在设计时无效，运行时只读。

3. 主要方法

(1) AddItem 方法

AddItem 方法用来向列表框中添加一个项目。语法格式为：

列表框对象.AddItem 项目[, 索引号]

其中“项目”为字符串表达式，表示新加项目的内容。“索引号”指定添加(插入)的项目在列表中的位置，省略参数“索引号”时，添加的项目排列在列表的最后(追加)。若指明索引号，当添加了一个项目后，其后项目的位置号自动重排。

例如，将“石家庄”追加到列表框 List1 中：

```
List1.AddItem "石家庄"
```

又如，如果在列表框 List1 中选择一个项目（可能复选），则以下代码可将被选定的项目追加到列表框 List2 中：

```

For i = 0 To List1.ListCount - 1
    If List1.Selected(i) Then
        List2.AddItem List1.List(i)
    End If
Next i

```

(2) RemoveItem 方法

RemoveItem 方法用来从列表框中删除一个项目。语法格式为：

列表框对象.RemoveItem 索引号

其中“索引号”指定要删除的项目在列表中的位置。当删除一个项目后，其后项目的位置号也自动重排。例如，删除列表框 List1 中的第一项：

```
List1.RemoveItem 0
```

(3) Clear 方法

Clear 方法清除列表框中所有项目。语法格式为：

列表框对象.Clear

4. 主要事件

列表框的主要事件是 Click 事件和 DblClick 事件。Click 事件在单击选择一个项目时被触发，DblClick 事件在双击一个项目时被触发。

要注意的是，如果在 Click 事件过程中有代码，则不会触发 DblClick 事件。在通常的操作中，单击一个项目后再配合一个确认按钮来表示选中；而双击一个项目则往往表示直接选中。为达到此效果，需要为 DblClick 事件设置代码，但不为 Click 事件设置代码，同时使用一个具有“确认”功能的命令按钮，在命令按钮的代码中检查列表框的 ListIndex 属性或 Selected 属性，以判断是否有项目被选中以及哪一个项目被选中。

【例 6.5】使用列表框显示城市名称，供用户选择，当单击**【确定】**按钮时，显示所选择的的城市名称。当双击列表框中的项目时，则直接显示所选择的的城市名称。

属性设置如表 6.5，运行结果如图 6.13 所示。

表 6.5 例 6.5 的对象的属性设置

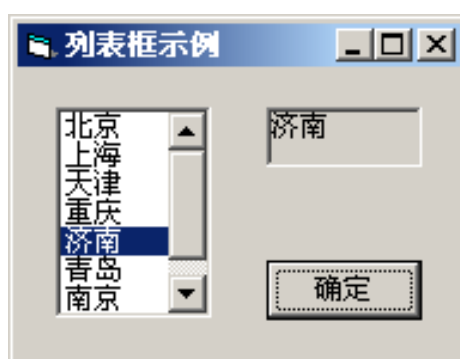


图 6.13 例 6.5 的运行结果

程序代码如下：

```

Private Sub cmdOk_Click() '单击“确定”按钮显示所选城市名称
    If List1.ListIndex <> -1 Then
        Label1.Caption = List1.List(List1.ListIndex)
    Else
        Label1.Caption = ""
    End If
End Sub

```

```

End If
End Sub
' 双击列表框中的项目，直接显示所选城市名称
Private Sub List1_DblClick()
    Label1.Caption = List1.List(List1.ListIndex)
End Sub

```

【例 6.6】设计一个选择购买图书种类的程序，程序的运行结果如图 6.14 所示。

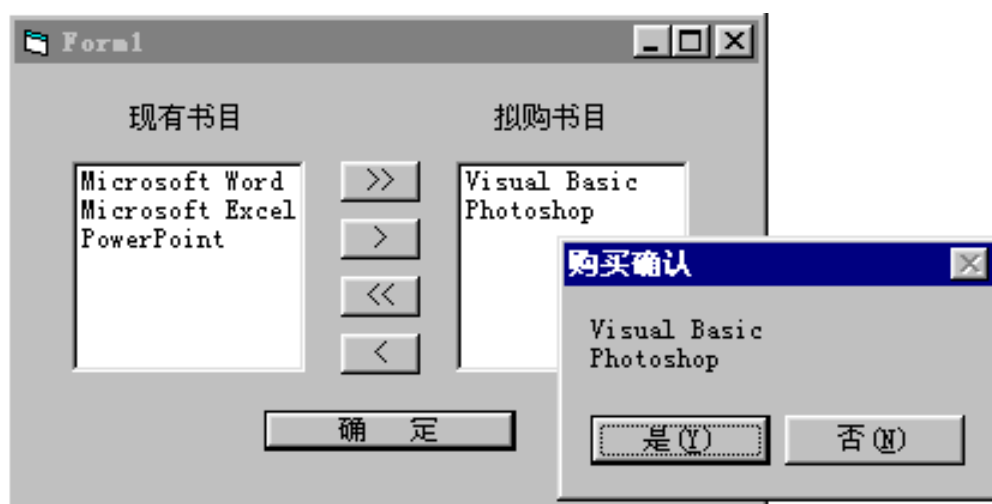


图 6.14 例 6.6 的运行结果

设计思路：使用两个列表框，左边的列表框 List1 显示现有书目，右边的列表框 List2 显示用户选择的拟购书目；四个命令按钮 cmdAll、cmdOne、cmdBackAll 和 cmdBackOne 的 Caption 属性分别设为“>>”、“>”、“<<”、“<”，形象地表示购买所有书目、购买选择的一个书目、退回所有书目、退回选择的一个书目；另设两个标签 Label1 和 Label2，其 Caption 属性分别设为“现有书目”和“拟购书目”；当用户选择好计划购买的书目时，单击标题为“确定”的命令按钮 cmdOk，弹出“购买确认”消息框，单击“是”，执行购买操作。

```

Dim i As Integer

Private Sub cmdBackAll_Click()
    For i = 0 To List2.ListCount - 1
        List1.AddItem List2.List(i)
    Next
    List2.Clear
End Sub

Private Sub cmdBackOne_Click()
    If List2.ListIndex <> -1 Then
        List1.AddItem List2.List(List2.ListIndex)
        List2.RemoveItem List2.ListIndex
    End If
End Sub

```

```

Private Sub cmdOk_Click()
    Dim str As String, ans
    If List2.ListCount > 0 Then
        For i = 0 To List2.ListCount - 1
            str = str & List2.List(i) & Chr(13) & Chr(10)
        Next i
        ans = MsgBox(str, vbYesNo, "购买确认")
        If ans = vbYes Then
            Print "Yes"
        End If
    End If
End Sub

Private Sub cmdOne_Click()
    If List1.ListIndex <> -1 Then
        List2.AddItem List1.List(List1.ListIndex)
        List1.RemoveItem List1.ListIndex
    End If
End Sub

Private Sub cmkAll_Click()
    For i = 0 To List1.ListCount - 1
        List2.AddItem List1.List(i)
    Next
    List1.Clear
End Sub

Private Sub Form_Load()
    List1.AddItem "Microsoft Word"
    List1.AddItem "Photoshop"
    List1.AddItem "Visual Basic"
    List1.AddItem "Microsoft Excel"
    List1.AddItem "Microsoft Powerpoint"
End Sub

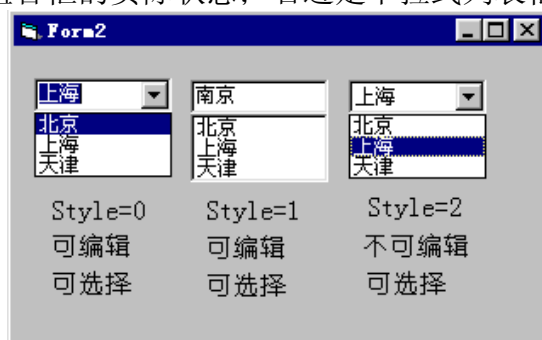
```

6.2.2 组合框

组合框(ComboBox)控件将文本框和列表框的功能结合在一起,既具有文本框的输入功能,又具有列表框的选择功能。通过组合框,用户既可输入文本内容,也可从列表中选择项目。

组合框的样式特点由 Style 属性确定。共可设置三种样式: Style=0 时,称为下拉式组合框(可编辑输入,可选择项目); Style=1 时,称为简单组合框(可编辑输入,可选择项目); Style=2 时,称为下拉式列表框(不可输入,只可选择项目)。

图 6.15 是组合框三种样式的示意图。其中，左边是下拉式组合框展开后的状态；中间是简单组合框的实际状态；右边是下拉式列表框展开后的状态。



在使用方式上，组合框具有和列表框相似的特征。组合框的主要属性有 Text、List、ListIndex、ListCount 和 Sorted 等，主要方法有 AddItem、RemoveItem 和 Clear。组合框的主要事件是 Click 事件。当 Style=1 时，还支持 DblClick 事件。

【例 6.7】用组合框提供商品类别，用列表框提供产品名称，设计一个产品信息查询程序。程序的运行结果如图 6.16 所示。



图 6.16 例 6.7 的运行结果

【分析】程序执行时，先隐藏标签 Label1、命令按钮 cmdReturn，并向组合框 Combo1 追加项目。当单击命令按钮 cmdShow 时，隐藏组合框 Combo1、列表框 List1 以及按钮 cmdReturn 和 cmdEnd，并显示标签 Label1 和按钮 cmdReturn。当单击命令按钮 cmdReturn 时，再次隐藏 Label1、cmdReturn，并显示 Combo1、List1、cmdShow 和 cmdEnd。这相当于在窗口的同一区域内交替显示不同的控件，是一种实用的设计技术。为了获得较好的视觉效果，设计时应该对控件的大小、位置等精心安排和调整。如果需要交替显示的控件数目较多，应该考虑使用框架。

当用户在组合框中选择了商品类别时，根据所选类别填充产品名称列表框。当用户选择了列表框中的项目（产品名称）并单击【显示商品信息】按钮时，根据用户在组合框中选择的类别（家电类、图书类、体育类），通过标签显示商品的类别、名称和价格。

交替显示的控件可分为两组。因此可以将它们分别放在两个框架（Frame）中，将两个框架设置为同样大小，在设计时（或在窗体的 Load 事件中）将两个框架的位置重合。这样，用于处理控件显示状态的代码只需两行语句：

```
Frame1.Visible = True      ' 或 False
Frame2.Visible = False    ' 或 True
```

【布置任务】：上机完成本例题。

【思考】：每年的考试结束后，老师们都要对试卷进行核分。设计一个程序，在输入每个题目的分数后，自动进行求和。同时要注意：1、各门课程的试卷的题目数是不同的。有时，每门课程的卷面分数所占比例不同。要使用列表框或组合框选择出题目的个数与分数的折扣，根据题目的个数，显示出相应个数的文本框。2、尽量减少鼠标使用，在文本框输入分数后，按下回车键，自动进入下一个文本框开始输入。当最后一个题目输入完成后，计算出总分与折扣后的分数并显示，同时，激活第一个文本框，开始下一个同学的成绩的计算。

6.3 图像显示控件

6.3.1 图片框

【目标】：通过本节内容的学习，能够编写出一个简单的图片浏览程序。

图片框 (PictureBox) 既可以用来显示图像，也可以作为其它控件的容器，起到类似于框架及窗体的作用。它可以使用多种类型的图形文件：位图 (bitmap, 扩展名为 .bmp)、图标 (icon, 扩展名为 .ico)、Windows 元文件 (metafile, 扩展名为 .wmf) 以及 JPEG 和 GIF 文件。图片框还和以前介绍的窗体一样，支持各种绘图方法。

1. 主要属性

(1) Picture 属性

设置要显示的图形，支持 bmp、jpg、ico、wmf 等文件类型。图形文件可以在设计阶段装入，也可以在运行期间装入。

在设计状态下，可以通过属性窗口中的 Picture 属性指定图形文件。在运行时，Picture 属性和 LoadPicture 函数配合，将图形加载到控件上。LoadPicture 函数格式如下：

LoadPicture([文件名])

若省略文件名，则清除控件中的图形。

例如，若图片框控件的名称为 PName，则在程序中装载和清除图形的方法如下：

装入图形到控件：

PName.Picture = LoadPicture("c:\image\aa.bmp")

删除控件中图形：PName.Picture = Nothing

或：Set PName.Picture = Nothing

或：PName.Picture = LoadPicture()

向图片框控件装入图形，还可以通过剪贴板进行。首先通过 Window 常规操作向剪贴板放入图像；然后在 VB 的设计状态下选中图片框控件，执行【编辑|粘贴】命令。

(2) Autosize 属性

该属性设置图片框是否会根据装入图形的大小作自动调整。默认值为 False，表示图片框的大小不会自动改变，对于较大的图片，显示不下的部分被隐藏；当值为 True 时，表示自动改变图片框的尺寸，以适应图形的大小，如图 6.17 所示。

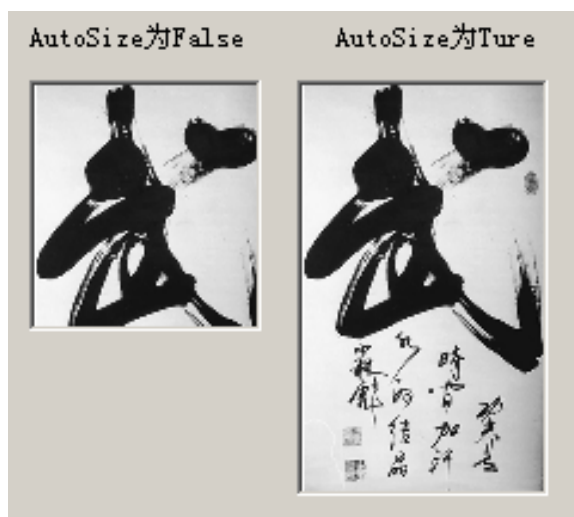


图 6.17 图片框的 Autosize 属性

2. 主要方法和事件

(1) Print 和 Cls 方法

这两个方法的使用与窗体相同，不再赘述。

(2) 绘图方法

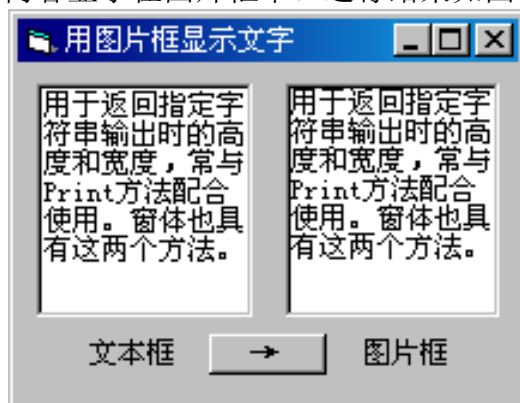
绘图方法包括 Line、Circle、Pset 和 Point 方法。这些方法可用于图片框和窗体，详细内容将在第 12 章介绍。

(3) TextHeight 和 TextWidth 方法

用于返回指定字符串输出时的高度和宽度，常与 Print 方法配合使用。窗体也具有这两个方法。

图片框支持常用的鼠标事件、键盘事件和焦点事件等。

【例 6.8】用图片框显示文字。程序运行时先在文本框中输入内容，单击“→”按钮后将文本框中的内容显示在图片框中。运行结果如图 6.18 所示。



在窗体上添加一个文本框 Text1，设 MultiLine 属性为 True，Text 属性为空。添加一个图片框 Picture1，设 AutoRedraw 属性为 True，背景色为白色。添加两个标签，Caption 属性分别为“文本框”和“图片框”。添加一个命令按钮 Command1，Caption 属性为“→”。

编程思路：用 Print 方法在图片框中显示文字时，若内容较多，超出图片框宽度的部分将被截掉。为了能够像多行文本框那样自动换行，可以利用图片框的 CurrentX 属性和 TextWidth 方法。具体做法是利用循环结构，一次只输出一个字符，每次输出前先作检查，如果下一字符的输出位置将超过图片框的宽度时则换行。代码如下：

```

Private Sub Command1_Click() '单击“→”按钮
    Dim strS As String, tmp As String
    Dim intW As Integer, i As Integer
    strS = Text1.Text '取文本框中的内容存入变量
    Picture1.Cls
    For i = 1 To Len(strS) '通过循环每次输出一个字符
        tmp = Mid(strS, i, 1) '取第 i 个字符
        intW = Picture1.TextWidth(tmp) '取第 i 字符的宽度
        '如果第 i 字符的宽度+当前输出位置 CurrentX 超过图片框的宽度则换
        行
        If intW + Picture1.CurrentX > Picture1.Width Then
            Picture1.Print
        End If
        Picture1.Print tmp; '输出第 i 个字符。注意分号
    Next
End Sub

```

窗体对象也具有 AutoRedraw、CurrentX 和 CurrentY 属性以及 TextHeight 和 TextWidth 方法。利用这些属性和方法可以在窗体（或图片框）中显示特殊效果文字。例如，为窗体的单击事件编写以下代码，可以在窗体上显示如图 6.19 所示的立体字和阴影字。



图 6.19 特殊效果文字

```

Private Sub Form_Click()
    Const Pos = 200
    Const Mov = 80
    Dim i As Integer, ss As String
    ss = "立体字"
    FontSize = 48
    FontName = "黑体"
    FontBold = True
    ForeColor = vbBlack
    CurrentX = Pos
    CurrentY = Pos
    For i = 1 To Mov
        Print ss
        CurrentX = Pos - i
        CurrentY = Pos - i
    Next
    ForeColor = vbWhite
    Print ss;

```

```

ss = "阴影字"
FontName = "华文新魏"
FontBold = False
ForeColor = vbBlack
CurrentX = CurrentX + Pos
CurrentY = Pos
Print ss;
CurrentX = CurrentX - TextWidth(ss) - Mov
CurrentY = CurrentY - Mov
ForeColor = vbWhite
Print ss
End Sub

```

6.3.2 图像框

图像框控件（Image）的使用方法与图片框（PictureBox）类似，但它只能用来显示图像，不能完成复杂的图像操作。

Image 控件的属性主要是 Picture 属性和 Stretch 属性。其 Picture 属性的意义和用法与 PictureBox 控件相同。Stretch 属性可以决定所加载的图片是否缩放，默认值为 False，表示图片不缩放，控件的大小由图片决定，即控件自动适应图片的大小；当 Stretch 属性为 True 时控件的大小不变，图片自动伸缩（放大或缩小）以便适合控件。图 6.20 展示了 Stretch 属性不同取值的效果。

其中左图为设计时界面，右图为运行时界面。从图中可以看出，当 Stretch 属性为 False 时，尽管在设计时将几个 Image 控件设为不同大小，但运行时，控件均自动调整为图片的大小。

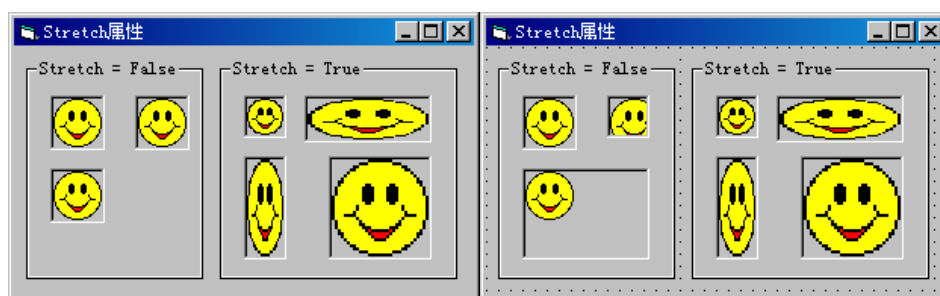


图 6.20 Stretch 属性

Image 控件与 PictureBox 控件的比较如下：

1. 两者都可加载图片，都支持相同的图片格式，加载图片的方法也一样。但 PictureBox 控件的图形功能更强，而 Image 控件由于属性少，使用的系统资源比 PictureBox 控件少，装载图形的速度快。
2. Image 控件中，通过设置 Stretch 属性为 True 可以实现图片缩放以适合控件的大小，但图片可能变形失真；在 PictureBox 控件中，仅可通过 Autosize 属性调整控件的大小以适合图形，图形本身并不缩放。
3. PictureBox 控件可以作为其它控件的容器，其内允许包括其它控件，起到类似于框架的作用，还支持各种绘图方法和 Print 方法；而 Image 控件则不能。

【例 6.9】设计一个程序，根据用户指定的图像文件显示图像。为了实现文件选择功能，本例使用了上一章介绍的通用对话框中的“打开”对话框，在代码中将对话框的 Filter 属性（文件过滤器）设置为*.bmp、*.jpg、*.ico 和*.*。



另外，两个命令按钮 cmdCls、cmdLoad 分别清除和装载给定的图像，标签控件显示所选择的图像文件名称。程序的运行结果如图 6.21 所示。

程序代码如下：

```
Private Sub cmdCls_Click()           ' 单击“清除”按钮
    Image1.Picture = LoadPicture("")
    Label1.Caption = ""
End Sub
Private Sub cmdLoad_Click()          ' 单击“装载”按钮
    On Error GoTo ErrCancel
    CommonDialog1.CancelError = True
    CommonDialog1.Filter = "BMP(*.bmp)|*.bmp|" & _
        "JPG(*.jpg)|*.jpg|ICO(*.ico)|*.ico|All(*.*)|*.*"
    CommonDialog1.ShowOpen
    Image1.Picture = LoadPicture(CommonDialog1.FileName)
    Label1.Caption = CommonDialog1.FileName
Exit Sub
ErrCancel:
    Label1.Caption = "No File"
End Sub
```

6.4 定时器

定时器（Timer）或称计时器，是一个响应时间的控件。它独立于用户，运行时不可见，可用来在一定的时间间隔中周期性地执行某项操作。

1. 主要属性

（1）Enabled 属性

设置定时器是否生效。当该属性为 True（默认值）时，定时器处于工作状态（生效）；而当 Enabled 被设置为 False 时，它会暂停操作而处于待命状态（无效）。

（2）Interval 属性

设置定时器的时间间隔，单位为毫秒（1000 毫秒=1 秒），取值范围为 0~65535，因此最大时间间隔约为 65.5 秒。尽管设置值可取 1 毫秒，但在 Windows

9x 下，实际最短间隔仅能达到 1/18 秒（约 56ms），在 Windows 2000/XP 下，实际最短间隔可达 10ms。要注意的是，Interval 属性的默认值是 0，此时，即使 Enabled 属性为 True，定时器仍无效。

2. 事件

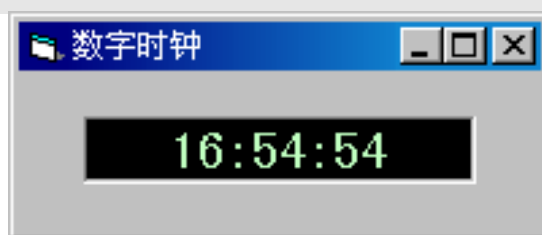
定时器只能识别 Timer 事件。当到达由 Interval 属性所设定的时间间隔时，系统会自动触发其 Timer 事件，转去执行 Timer 事件中的代码，从而完成指定的操作，接着又开始新一轮的计时。这样，Timer 事件中的代码可以每隔一个时间段就被执行一次。

【例 6.10】设计数字时钟，动态显示系统当前时间。

通过标准函数 Time 可以取得系统的当前时间，要使其动态显示出来则可以使用定时器控件实现。在窗体上添加一个定时器 Timer1，设 Enabled 属性为 True，Interval 属性为 1000（即 1 秒）。添加一个标签，设 Caption 为空，Alignment 为 2，背景色为黑色，前景色为浅绿色。

为定时器 Timer 事件编写以下代码：

```
Private Sub Timer1_Timer()  
    ' 调用 Time 函数在标签中显示时间  
    Label1.Caption = Time  
End Sub
```



【例 6.11】用定时器控件制作秒表。

设计和运行界面如图 6.23 所示，其中左图为设计界面，中图和右图为运行界面。



图 6.23 例 6.11 设计和运行界面

（1）设计界面及设置属性

在窗体上添加一个定时器 Timer1，设 Enabled 属性为 False，Interval 属性为 10。添加一个标签 Label1 用于显示计时时间，设其 Caption 为

“0:00:00.00”，Alignment 为 2，背景色为黑色，前景色为浅绿色。再添加两个命令按钮，名称分别为 cmdTime 和 cmdReset，设 Caption 分别为“开始”和“清零”。

（2）编写代码

为了简化界面，便于用户操作，本例中通过代码让 cmdTime 按钮“身兼三职”，完成开始、暂停和继续功能。程序启动时该按钮的标题为“开始”。单击“开始”，开始计时，按钮变为“暂停”。单击“暂停”，定时器停止工作，按钮变为“继

续”。单击“继续”，继续计时，按钮又变为“暂停”。单击“清零”按钮，定时器停止工作，标签中的计时读数置 0，cmdTime 按钮的标题恢复为“开始”。

制作秒表的几个关键环节：①记录开始计时的时间，可以通过调用 VB 内部函数 Timer 为变量赋值来实现。该函数返回从午夜零点开始至当前时刻的总秒数（Single 型数据，精度为 7 位）。②计算开始计时至当前时刻的时间差，用 Timer 函数的返回值减去开始计时的时刻即可获得该时间差。③在系统允许的最短时间间隔内将时间差以“时:分:秒.xx”的形式显示。适当设置定时器控件的 Interval 属性，在定时器的 Timer 事件中将时间差总秒数转换为时、分、秒，并调用 Format 函数以特定的时间格式显示。为完成上述功能，需要设置若干变量，用于存储和计算有关的时间数据。程序代码如下：

```
Dim strH As String, strM As String
Dim strS As String, strSS As String
Dim sngT As Single
Dim intT As Integer
Dim sngStart As Single
Private Sub cmdTime_Click()
    If cmdTime.Caption = "暂停" Then
        cmdTime.Caption = "继续"
        Timer1.Enabled = False
    Else
        If cmdTime.Caption = "开始" Then sngStart = Timer
        Timer1.Enabled = True
        cmdTime.Caption = "暂停"
    End If
End Sub
Private Sub Form_Load()
    Timer1.Enabled = False
    Label1.Caption = "0:00:00.00"
    cmdTime.Caption = "开始"
End Sub

Private Sub mdReset_Click()
    Form_Load
End Sub
Private Sub Timer1_Timer()
    sngT = Timer - sngStart
    strSS = Format(sngT * 100 Mod 100, "00")
    intT = Int(sngT)
    strS = Format(intT Mod 60, "00.")
    strM = Format(intT \ 60 Mod 60, "00:")
    strH = Format(intT \ 3600, "00:")
    Label1.Caption = strH & strM & strS & strSS
End Sub
```

【例 6.12】设计一个具有简单动画效果的程序。

编程思路：将一个装有图片的图像框在定时器的每个 Timer 事件中按一定方向和距离移动，即可实现简单的动画效果。设计动画程序首先要选择一个合适的图像文件，本例选择的是 ARW11NE. ICO 文件。在窗体上放置一个计时器控件 Timer1，Interval 属性设置为 50。添加一个图像框控件 Image1，设 Picture 属性为上述图像文件。设计界面如图 6.24 所示。运行时可看到图中的箭头周而复始地从左下向右上移动。

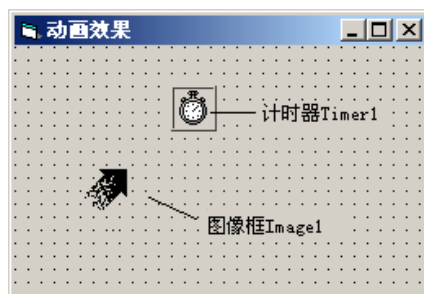


图 6.24 例 6.12 的界面设计

代码如下：

```
Private Sub Form_Load()      ' 窗体加载
    Me.Height = 4000
    Me.Width = 6000
    Image1.Top = Me.Height    ' 图像框定位
    Image1.Left = 0
End Sub
Private Sub Timer1_Timer()    ' 定时器事件
    Dim x As Integer, y As Integer
    x = Image1.Left + 50      ' 设置图像框的新位置参数(向右上)
    y = Image1.Top - 50
    ' 若新位置移出窗体则回到初始位置
    If x > Me.Width Or y < 0 Then
        x = 0
        y = Me.Height
    End If
    Image1.Move x, y          ' 移动图像框到新位置
End Sub
Private Sub Form_Unload(Cancel As Integer) ' 窗体卸载
    ' 关闭定时器。亦可用 End 语句结束运行。
    Timer1.Enabled = False
End Sub
```

【思考】：给篮球比赛设计一个计时器。具体功能要查找篮球比赛规则。尽量使用键盘操作，不要过多使用鼠标。时间要精确到 0.1 秒，必须采用倒计时。为了防止操作时候的一些失误，尽量有记忆与恢复功能。

6.5 滚动条

VB 提供了两种滚动条控件：水平滚动条（HScrollBar）和垂直滚动条（VScrollBar）。两者除滚动的方向不同外，其功能和操作是一样的。滚动条的两端各有一个滚动箭头，在滚动箭头之间有一个滚动块。滚动块从一端移至另一端时，其值在不断变化。垂直滚动条的值由上往下递增，水平滚动条的值由左往右递增。滚动条的值均以整数表示，取值范围为-32768~32767，最小值和最大值分别在两个端点。

1. 常用属性

（1）Min 属性

设置滚动条所能代表的最小值，默认值为 0。

（2）Max 属性

设置滚动条所能代表的最大值，默认值为 32767。

（3）Value 属性

设置或返回滚动条当前表示的值，也即当前滑块的位置。

（4）SmallChange 属性

最小变化值，当鼠标单击滚动条上的箭头时，一次产生的变化值。

（5）LargeChange 属性

最大变化值，当鼠标单击滚动条滑块与箭头之间的空白区域时，一次产生的变化值。

2. 主要事件

（1）Change 事件

当滚动条的 Value 值发生改变时，触发 Change 事件。能引起滚动条 Value 值改变的操作包括：单击滚动条两端的箭头、单击箭头与滑块之间的区域、直接对 Value 属性赋值等。

（2）Scroll 事件

当拖动滚动条的滑块时产生 Scroll 事件。

注意：Change 事件和 Scroll 事件是有差异的，当开始拖动滑块后，只要拖动动作在继续，就会不断地产生 Scroll 事件；当拖动停止时，则产生 Change 事件。

【例 6.13】设计程序，利用滚动条控件控制颜色改变。

利用 VB 提供的标准函数 RGB 可以方便地设置颜色，RGB 函数的语法格式如下：

RGB(红, 绿, 蓝)

红、绿、蓝三个参数的取值范围均为 0~255。只要控制函数 RGB 的三个参数即可设置不同颜色，这可以由三个滚动条完成。程序中使用了三个水平滚动条：hsbR、hsbG、hsbB，分别代表红绿蓝三色。为了适应 RGB 参数，其 Min 和 Max 分别设为 0 和 255；SmallChange 和 LargeChange 分别设为 1 和 5。当任何一个滚动条的状态发生变化时，在其 Change 事件中将各滚动条的当前值（Value 属性）作为 RGB 函数的参数，为标签 Label4 的 BackColor 属性赋值，即改变标签的背景色，以便适时、直观地反映变化后的颜色。

程序的运行结果如图 6.25 所示，滚动条的状态控制着 Label4 的背景色。

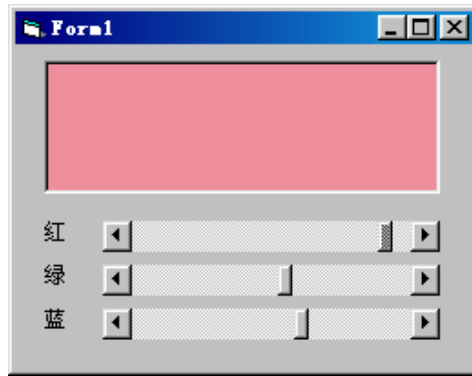


图 6.25 例 6.13 的运行结果

程序代码如下：

```
Private Sub Form_Load() ' 窗体加载，控件初始化
    Label4.BackColor = RGB(100, 100, 100)
    hsbR.Value = 100
    hsbG.Value = 100
    hsbB.Value = 100
End Sub

Private Sub hsbR_Change() ' “红” 滚动条的值改变
    Label4.BackColor = RGB(hsbR.Value, _
        hsbG.Value, hsbB.Value)
End Sub

Private Sub hsbG_Change() ' “绿” 滚动条的值改变
    Label4.BackColor = RGB(hsbR.Value, _
        hsbG.Value, hsbB.Value)
End Sub

Private Sub hsbB_Change() ' “蓝” 滚动条的值改变
    Label4.BackColor = RGB(hsbR.Value, _
        hsbG.Value, hsbB.Value)
End Sub
```

上述三个滚动条的 Change 事件中的语句完全相同，而且语句较长。对这种在程序中重复使用的程序段，可以将其编制成一个通用过程或函数供调用，以简化代码，提高代码的复用性。在本例中可设置一个通用过程 ShowColor()，在每一个滚动条的 Change 事件中调用。过程代码如下：

```
Private Sub ShowColor()
    Label4.BackColor = RGB(hsbR.Value, _
        hsbG.Value, hsbB.Value)
End Sub
```

在各滚动条的 Change 事件中通过以下语句调用：

ShowColor ' 或 Call ShowColor

运行此程序时，当拖动滚动条的滑块到达某个位置并释放后，标签 Label4 的背景色发生了变化，但在拖动滚动条滑块期间，颜色并没有变化。这是因为没有设置 Scroll 事件的缘故。只要为每个滚动条的 Scroll 事件添加代码，加入对 ShowColor 过程的调用，即可解决此问题。

【例 6.14】设计一个浏览图像的程序。

浏览图像是指利用较小的屏幕区域观察较大尺寸的图像。本例的设计思路是：使用 PictureBox 控件作为容器，其内包含有一个 Image 控件，要浏览的图像完整地装载到 Image 控件，利用滚动条改变 Image 控件在 PictureBox 控件中的位置，实现浏览的目的。通用对话框控件用于打开图片文件。三个命令按钮 cmdCls、cmdLoad、cmdEnd 分别用于装载、清除图像以及退出。除命令按钮的名称和 Caption 属性外，其他控件的属性均取默认值。

设计和运行界面如图 6.26 所示，其中左图为设计界面，中图和右图为运行界面。



图 6.26 例 6.14 设计和运行界面

1. 简述列表框控件和组合框控件的主要方法。
2. 使用复选框或单选按钮时，程序中如何判断它们的状态是否被选中？
3. 对滚动条进行什么操作时，会触发其 Change 事件、Scroll 事件？
4. 比较 Image 控件与 PictureBox 控件各自的特点。如何选择使用哪种控件？
5. Timer 控件的 Enable 属性为 True 时，将其 Interval 属性分别设置为 10000、60000 意味着什么？
6. 设计一个程序，用滚动条控制改变标签的字体大小。
7. 根据图 6.27 所示界面，设计“个人信息数据”采集程序。姓名在文本框内输入；出生日期从 3 个组合框中选择，年份从 1910 到 2003，月份从 1 到 12，日期从 1~31（不计闰月）；选择后的出生日期在标签中实时显示。【添加到列表】按钮将姓名和出生日期组成一条记录，添加到列表框中；【从列表删除】按钮将列表框内选中的记录删除。
8. 按图 6.28 所示界面设计一个程序，求汽车在不同速度、不同时间下的行车距离。其中，汽车的速度最大为 100 公里/小时，行车时间最长为 50 小时。
9. 任选两个图像文件，设计一个具有图像交替显示效果的动画程序。
10. Time 函数和 Data 函数可以返回系统的时间、日期。设计程序用来显示系统的当前时间、日期。并思考：如何修改系统的时间、日期。

第 7 章 数组

【本章要点】

数组的有关概念、声明或建立的方法
一维数组、二维数组和动态数组的应用
控件数组的有关概念及应用

7.1 一维数组

7.1.1 引例

【例 7.1】求一个班 40 名学生的平均成绩，然后统计高于平均分的人数。

若用简单变量结合 For...Next 语句，求平均成绩的程序段如下：

```
P = 0
For I = 1 to 40
    S = InputBox("请输入第" & I & "位学生的成绩：")
    P = P + S
Next I
P = P / 40
```

但是，若要统计高于平均分的人数，则无法实现。可用数组解决求 40 人的平均分和高于平均分人数的问题，完整程序编写如下：

```
Private Sub Command1_Click()
    Dim S(1 To 40) As Integer
    Dim P!, N%, i%
    P = 0
    For i=1 To 40
        S(i)= InputBox("请输入第" & i & "位学生的成绩：")
        P = P + S(i)
    Next I
    P = P / 40
    N = 0
    For i=1 To 40
        If S(i) > P Then N = N + 1
    Next i
    Print "平均分="; P, "高于平均分的人数="; N
End Sub
```

7.1.2 一维数组的概念

在上例中，使用的 S(i) 是一个数组，因为只有一个下标，所以又称一维数组。一维数组就是只有一个下标的数组。实际上，数组就是一组具有相同名字、不同下标的变量的集合。需要注意数组并不是一种数据类型，它是用来存放或表示一组相关的数据。

VB 中的数组有一维数组、二维数组、…，最多 60 维；二维及二维以上的数组也称多维数组。按声明时数组的大小确定与否分为静态（定长）数组和动态（可调）数组两类。

数组必须先声明后使用，主要声明数组名、类型、维数、数组大小。按声明时下标的个数确定数组的维数。例 7.1 中的语句：

```
Dim S(1 To 40) As Integer
```

声明了一个一维定长数组，该数组的名字为 C，类型为整型；共有 40 个元素，下标范围为 1 到 40；S 数组的各元素是 S(1), S(2), S(3), ..., S(40)；S(i) 表示由下标 i 值决定是哪一个元素。

7.1.3 一维数组的声明和引用

1. 一维数组的声明

声明一维数组的格式如下：

Dim 数组名(下标) [As 类型]

其中：

下标：必须为常数，不可以为表达式或变量。下标的形式为：[下界 To] 上界，下标的上下界不得超过长整型数据类型的范围，且受内存大小限制。若省略下界，其默认值为 0。一维数组的元素（分量）个数为：上界—下界+1。

As 类型：指定数组的数据类型（数组中各元素的数据类型）。如果省略，即不明确给出数组的类型，则数组与以前所述简单变量的声明一样，默认为变体型数组。

用 Dim 语句声明数组，实际上就是为系统提供数组名、数组类型、数组的维数和各维大小等相关信息。

例如：

Dim W(100) As Integer

声明了 W 为数组名，整型，一维数组，有 101 个元素；下标的范围 0~100。若在程序中使用 W(101)，则系统会显示错误信息“下标越界”。

又如：

Dim T(-5 To 8) As String * 6

声明了 T 为数组名，字符串类型，一维数组，有 14 个元素；下标的范围-5~8，每个元素最多存放 6 个字符。

注意，以下数组声明是错误的：

M = 50

Dim X(M) As Single

因为数组声明中的下标不能是变量，只能是常量。

2. 一维数组的引用

在对数组操作时，引用一维数组元素的形式是：

数组名(下标)

注意：下标不能超出数组声明时的上、下界范围。下标可以是整型的常数、变量、表达式，甚至又是一个数组元素。

如 C(8)、C(3+13)、C(i) 都是正确的引用形式。

一维数组元素的使用规则与同类型的简单变量相同。

在通常情况下，数组中的各元素类型必须相同，但若数组类型为 Variant（变体）时，可存放不同类型的数据，但

【比较】比较 VB 与 C 语言以及学过的其他语言中的数组。

7.1.4 一维数组的使用

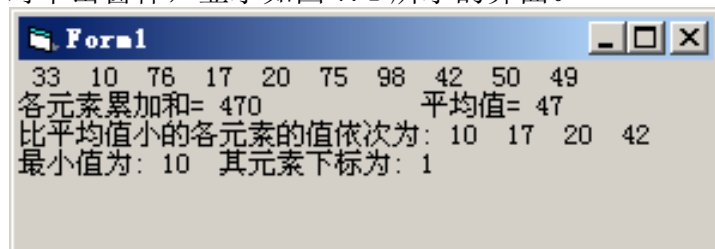
编写程序时，一维数组通常与 For 循环结合使用，For 语句中的循环变量作为数组元素的下标，通过循环变量的不断改变，达到对每个数组元素依次进行处理的目的。

【例 7.2】随机产生 10 个两位数的随机整数，赋给 a 数组，然后求各元素之和、平均值，将比平均值小的各元素的值打印出来，最后找出数组中的最小值及其元素下标并打印。

编写代码如下：

```
Private Sub Form_Click()  
    Dim S, P, T, B, I, a(9) As Integer  
    Randomize  
    ' 随机产生 20 个两位数的随机整数赋给 a 数组  
    For I = 0 To 9  
        a(I) = Int(Rnd * 90 + 10)  
        Print a(I);  
    Next I  
    Print  
    S = 0  
    For I = 0 To 9  
        S = S + a(I)    ' 求累加和  
    Next I  
    P = S / 10    ' 求平均值  
    Print "各元素累加和="; S, "平均值="; P  
    Print "比平均值小的各元素的值依次为:";  
    For I = 1 To 9  
        If a(I) < P Then Print a(I);  
    Next I  
    Print  
    ' 求数组中的最大值及其元素下标并打印  
    T = a(0): B = 0  
    For I = 1 To 9  
        If T > a(I) Then T = a(I): B = I  
    Next I  
    Print "最小值为:"; T, "其元素下标为:"; B  
End Sub
```

程序运行时单击窗体，显示如图 7.1 所示的界面。



【例 7.3】设某数组中有 10 个数，用选择法按递增顺序排序。

算法分析：选择法排序是最为简单且易理解的算法。假定有 n 个数的序列，要求按递增的次序排序，算法步骤是：

（1）通过循环从 n 个数中选出最小数的下标，出循环后最小数与第 1 个数交换位置。

（2）除第 1 个数外，其余 $n-1$ 个数再按步骤（1）的方法选出次小的数，与第 2 个数交换位置。

（3）重复步骤（1） $n-1$ 遍，最后构成递增序列。

设计方法：在窗体上添加一个命令按钮，设 Caption 为“选择法排序”，用来触发排序过程。为简化操作，数组中的 10 个数据为 0~99 的整数，用随机函数来产生。运行结果如图 7.2，代码详见教材。另外，数组排序还可使用冒泡法、插入法等方法。

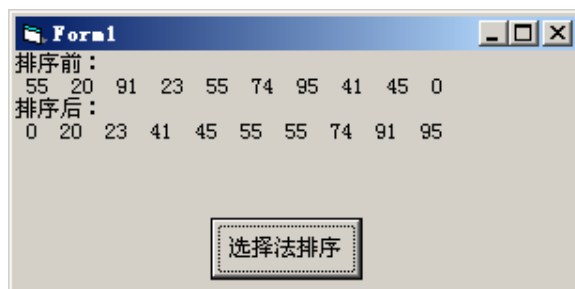


图 7.2 例 7.3 运行结果

【思考】：把例 7.3、组合框与时间控件相结合，在排序之前，先从组合框中选择秒数，然后在排序的时候，每隔若干秒，排一次。这样一来，我们就能看清楚排序的整个过程。如果能选择排序的算法，则更好。

7.2 多维数组

7.2.1 多维数组的声明

多维数组声明形式如下：

`Dim 数组名(下标 1, 下标 2[, 下标 3 ...]) [As 类型]`

其中，下标有两个以上的数字，它决定了数组的维数为多维。

每一维的大小为：上界—下界+1；数组的大小为各维大小的乘积。

例如：

`Dim K(0 To 2, 0 To 3) As Long`

或

`Dim K(2, 3) As Long`

都是声明了长整型的二维数组 K，第一维下标范围为 0~2；第二维下标范围为 0~3，占据 3×4 个长整型变量的存储空间。

说明：

（1）在默认情况下，声明的静态数组其下标下界从 0 开始，为了便于使用，在 VB 中的窗体级或标准模块级中可用 `Option Base n` 语句重新设定数组的默认下界。例如：

`Option Base 1` ' 设定数组下标默认下界为 1

(2) 在数组声明中的下标关系到每一维的大小，是数组说明符，说明了数组的整体；而在程序其他地方出现的下标是为了确定数组中的一个元素，也就是用来表示数组中的一个元素。两者写法相同，但意义不同。例如：

```
Dim R(10) As Long ' 声明了 R 数组，有 11 个元素
R(10)=100         ' 对 R(10) 这个数组元素赋值
```

(3) 在数组声明时的下标只能是常数，而在其他地方出现的数组元素的下标可以是变量，注意加以区分。例如：

```
Dim x(n) As Integer ' 出错
n 是变量，运行时会出现“要求常数表达式”的提示信息
x(n) = 10 ' 数组元素的下标可以是变量
```

在对多维数组操作时，引用多维数组元素的形式是：

数组名（下标 1，下标 2 [，下标 3 …]）

有关规则和方法与一维数组相同。

(4) 尽管 VB 允许声明多达 60 维的数组，但是随着数组维数的增加，数组所占内存会大幅度增加，甚至造成内存溢出，因此应慎用多维数组。

7.2.2 多维数组的使用

在利用多维数组编写程序时，多维数组通常与多重 For 循环结合使用，每重 For 语句中的循环变量分别作为数组元素的不同下标，通过循环变量的不断改变，达到对多维数组中每个数组元素依次进行处理的目的。

在多维数组中，使用最多的是二维数组。编写程序时，二维数组通常与二重 For 循环结合使用，每重 For 语句中的循环变量分别作为数组元素的两个下标。

例如，声明一个二维数组，用于存放 20 名学生的 4 门课程成绩：

```
Dim S(1 To 20, 1 To 4) As Integer
```

若数组中已有成绩，下面的程序段可显示 20 名学生的 4 门课成绩：

```
For i = 1 To 20 ' 共显示 20 行
    For j = 1 To 4 ' 每行显示 4 个成绩
        Print S(i, j);
    Next j
Print
Next i
```

二维数组常用于矩阵操作，下面的示例说明了它的应用。

【例 7.4】利用随机函数随机产生两个两位数的 5×5 矩阵并作运算。

要求如下：

- (1) 将两个矩阵相加，结果放入矩阵 C 中；
- (2) 统计矩阵 C 中最大值和下标；
- (3) 求矩阵 A 两条对角线元素之和；
- (4) 将矩阵 A 按列的次序把各元素放入一维数组 D 中。

设计方法：在窗体上添加 4 个命令按钮，用来触发有关运算的事件过程；3 个图片框，分别用来显示 A 矩阵、B 矩阵及各种运算结果。界面设计见图 7.3。

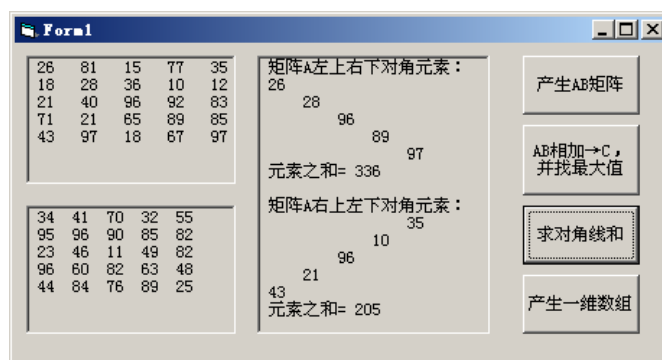


图 7.3 例 7.4 运行结果

【思考】将本例中，随即产生的矩阵修改为由用户输入，并且由用户确定是几行、几列的矩阵，然后进行计算。需要增加文本框与组合框。

7.3 动态数组

动态数组也叫可调数组或可变长数组，指在声明数组时未给出数组的大小（省略括号中的下标），当要使用它时，随时用 ReDim 语句重新声明数组大小。使用动态数组的优点是根据用户需要，有效地利用存储空间，它是在程序执行到 ReDim 语句时分配存储空间，而静态数组是在程序编译时分配存储空间的。

建立动态数组的方法是，使用 Dim、Private 或 Public 语句声明括号内为空的数组，然后在后续的代码中用 ReDim 语句指明该数组的大小。

ReDim 语句形式如下：

ReDim 数组名（下标[，下标...]）[As 类型]

其中：下标可以是常量，也可以是有了确定值的变量（这一点与静态数组不同）。类型可以省略，若不省略，必须与 Dim 声明语句保持一致。

说明：

（1）在静态（定长）数组声明中的下标只能是常量，在动态数组 ReDim 语句中的下标可以是常量，也可以是有了确定值的变量。

（2）在过程中可多次使用 ReDim 来改变数组的大小，也可改变数组的维数。

（3）每次使用 ReDim 语句都会使原来数组中的值丢失，可以在 ReDim 关键字后加 Preserve 参数用来保留数组中的数据，但使用 Preserve 只能改变最后一维的大小，前面几维大小不能改变。

（4）使用 UBound 和 LBound 函数可以获取数组的上下界，并据以确定数组的大小。格式如下：

UBound(数组名[, 维]) ' 取上界

LBound(数组名[, 维]) ' 取下界

若省略“维”则默认为第一维。

【例 7.5】对例 7.1 稍作改进。学生的人数用输入对话框输入，学生的成绩用随机函数产生，计算的平均分和高于平均分的人数放在该数组的最后。

程序运行后当输入 n 的值为 10 时，显示结果如图 7.4，代码详见教材。

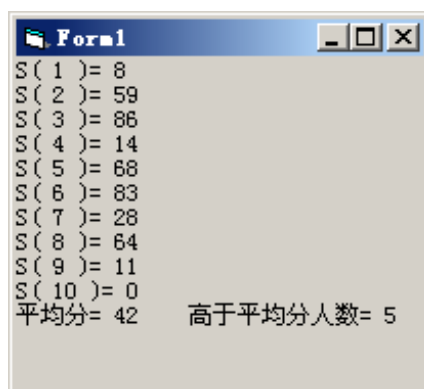


图 7.4 例 7.5 运行结果

7.4 控件数组

7.4.1 控件数组的概念

控件数组是由一组相同类型的控件组成。它们共用一个控件名，绝大部分属性也相同，但至少有一个属性不同，即 Index 属性的值不同。当建立控件数组时，系统给每个元素赋一个唯一的索引号(Index)，通过属性窗口的 Index 属性，可以知道该控件的下标是多少，第 1 个元素下标是 0。例如，CmdNum(8)表示名为 CmdNum 的控件数组的第 9 个元素。

控件数组最大的特点是：控件数组各元素共享同样的事件过程，所以适用于若干个控件执行的操作相似的场合。例如，控件数组 CmdNum 有 10 个命令按钮，则不管单击哪个命令按钮，就会调用同一个单击事件过程。为了区分是控件数组中的哪个元素触发了事件，在程序运行时，通过系统传送给过程的索引值（即下标值）来确定。

一个控件数组至少包含一个元素，最多可达 32768 个。

控件数组事件过程的结构如下（以控件数组 CmdNum 单击事件为例）：

```
' 比普通控件多了 Index 参数
Private Sub CmdNum_Click(Index As Integer)
    [语句块]
End Sub
```

7.4.2 控件数组的建立和使用

1. 在设计时建立

(1) 窗体上画出某控件，可进行控件名的属性设置，这是建立的第一个元素。

(2) 选中该控件，进行“复制”和“粘贴”操作，系统会提示如图 7.5 所示的对话框。单击【是】按钮后，即可建立控件数组元素。

(3) 进行事件过程的编程。

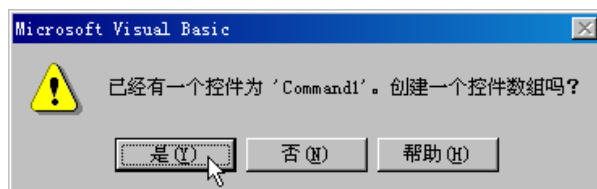


图 7.5 创建控件数组

【例 7.6】建立含有 6 个命令按钮的控件数组，当单击其中的命令按钮时，分别显示不同的图形或结束操作。运行界面见图 7.6。



图 7.6 例 7.6 运行结果

设计方法：先在窗体上建立一个命令按钮控件数组（有六个数组元素），然后在窗体上添加一个图片框。属性设置及代码见教材。

2. 运行时添加控件数组元素

（1）先在窗体上画出某控件，设置该控件的 Index 值为 0，表示该控件为数组；也可进行控件名的属性设置，这是建立的第一个元素。

（2）在程序代码中通过 Load 语句添加其余的若干个元素，也可以通过 Unload 语句删除某个添加的元素。

（3）对每个新添加的控件数组元素通过设置 Left 和 Top 属性，确定其在窗体上的位置，并将 Visible 属性设置为 True。

【思考】在例题中，增加几个文本框，根据文本框中的参数进行画图形。使该程序更加灵活。

【例 7.7】在窗体上建立一个标签控件数组，每个标签显示不同颜色。设计界面见 126 页图 7.7，运行界面见图 7.8。

要求：

① 设计时在窗体上放一个标签 Label1，设其 Index 属性为 0；添加两个命令按钮 Command1 和 Command2，Caption 分别为“一次性添加 8 个”和“逐个添加”。

② 程序运行时，单击命令按钮 Command1 时，自动产生 8 个 Label1 控件数组元素，BackColor 为不同颜色。

③ 若单击 Command2，则逐个产生标签控件数组元素，BackColor 为不同颜色。

设计方法：为了编程方便，在设计时建立的第 0 号元素起一个“模板”的作用，设其 Visible 属性为 False，程序运行时产生标签控件数组的其余元素。每个元素依次排列在同一行上，标签 BackColor 属性的赋值可通过 QBColor(n) 函数设置，其中 n 的值为 0~15。代码详见教材。

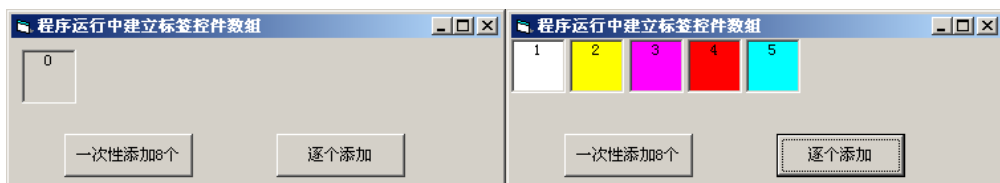


图 7.7 例 7.7 设计界面 7.8 例 7.7 运行结果

【例 7.8】编写一个能完成 $+$ 、 $-$ 、 $\times$ 、 $\div$ 四则运算的简易计算器程序。

思路：在窗体上建立一个命令按钮控件数组，有 10 个数组元素，其 Caption 属性值分别为 0~9，与小数点“.”按钮一起用来输入运算数据；一个标签作为显示屏，用来显示输入的数据及运算结果；另有五个命令按钮，其 Caption 属性值分别为 $+$ 、 $-$ 、 $*$ 、 $/$ 、 $=$ ，用来输入运算的符号；再添加两个按钮用于设置正负号和清屏。

运行界面如图 7.9 所示。

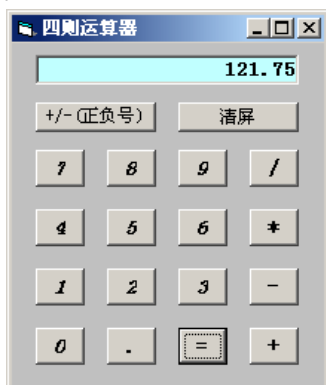


图 7.9 例 7.8 运行结果

【思考】编写一个计算器，能够完成类似 Windows 下的计算器。

习题：

1. 要分配 50 个元素的整型数组，下列数组声明（下界若无，按默认规定）哪些符合要求？

- (1) `k=50`
`Dim a(1 to k) As Integer`
- (2) `Dim a%`
`k=50`
`ReDim a(k)`
- (3) `Dim a%(9, 4)`
- (4) `Dim a(1, 4, 4) As Integer`
- (5) `Dim a%(100)`
`ReDim a(1 to 50)`
- (6) `Dim a!`
`ReDim a(4, 9) As Integer`
- (7) `Dim a%(5, 10)`
- (8)

2. 已知下面的数组声明，写出它的数组名、数组类型、维数、上下界、数组的大小，并行的顺序列出各元素。

`Dim M(3 To 5, 3) As Single`

3. 利用随机数生成两个 4×4 矩阵 A、B（矩阵 A 的每个数在 1~9 之间，矩阵 B 的每个数在 10~20 之间）。

要求：

- (1) 将两个矩阵相乘，结果放入矩阵 C 中（注意矩阵的乘法规则）；
 - (2) 统计矩阵 C 中最大值和下标及最小值和下标；
 - (3) 求矩阵 A 两条对角线元素的平均值；
 - (4) 将矩阵 A 按列的次序把各元素放入一维数组 D 中，显示结果。
4. 设某数组有 10 个元素，元素的值由键盘输入，要求将前 5 个元素的值与后 5 个元素的值互换，即第 1 个与第 10 个互换，第 2 个与第 9 个互换，依次类推。最后输出数组各元素原来的值和交换后各元素的值。
5. 设某数组有 10 个元素，用冒泡法按递增顺序排序。
6. 输入整数 n，分别按直角三角形和等腰三角形显示 n 行的杨辉三角形。含有 10 行元素的杨辉直角三角形和杨辉等腰三角形。

第 8 章 过程

本章要点：

- 子过程的概念和应用
- 函数过程的概念和应用
- 过程的参数与传递
- 标准模块与 Sub Main 过程的应用
- 常用的键盘和鼠标事件过程

8.1 子过程

8.1.1 通用过程的定义

1. 通用过程的语法格式

通用过程的语法格式如下：

```
[Public | Private] [Static] Sub 过程名([形参表])  
    [局部变量或常数声明]  
    [语句块]  
    [Exit Sub]  
    [语句块]  
End Sub
```

说明：

- (1) [Public | Private]：可选。指定过程的作用范围。若省略，则默认为 Public（全局）。
- (2) Static：可选。指定本过程内的所有局部变量均为静态变量。

(3) 过程名：命名规则与变量命名规则相同。无参数时，过程名后的括号不能省略。

(4) 形参表：形参表类似于变量声明，指明本过程被调用时传送给本过程的变量个数和类型。若有多个变量，各变量之间用逗号间隔。形参表中出现的参数称为形式参数，简称形参。每个形参的格式为：

[ByVal | ByRef] 形参名[()][As 类型]

ByVal 表示该参数按值传递，ByRef 表示该参数按地址传递（默认）。形参名必须是合法的变量名或数组名（后面加括号）。类型代表该参数的数据类型，默认为 Variant。不能用定长字符串变量或定长字符串数组作为形式参数，但是可以在调用过程时用简单定长字符串变量作为“实际参数”，VB 将其转换为变长字符串变量传递给过程。

(5) Exit Sub 语句表示立即退出子过程，通常将其置于选择结构中。

(6) 在过程内不能再定义过程，但可以调用其他过程。

2. 通用过程的创建

创建通用过程有两种方法：使用“添加过程”对话框；直接在代码编辑器窗口中输入过程代码。

(1) 使用“添加过程”对话框

步骤如下：

① 打开要添加过程的代码编辑器窗口。

② 执行【工具】菜单【添加过程】命令，打开如图 8.1 所示的【添加过程】对话框。

③ 在【名称】文本框中输入过程名。在【类型】框架中选择过程类型，其中【子程序】表示建立 Sub 过程，【函数】表示建立 Function 过程。在【范围】框架中选择范围，相当于使用 Public 或 Private 关键字。

④ 单击【确定】按钮后，在代码编辑器窗口中将出现如图 8.2 所示的过程模板。

(2) 直接在代码编辑器窗口中输入

在代码编辑器窗口中，将插入点放在已有过程的外面，按照规定的语法格式输入过程名和参数，系统会自动产生最后一行语句 End Sub。

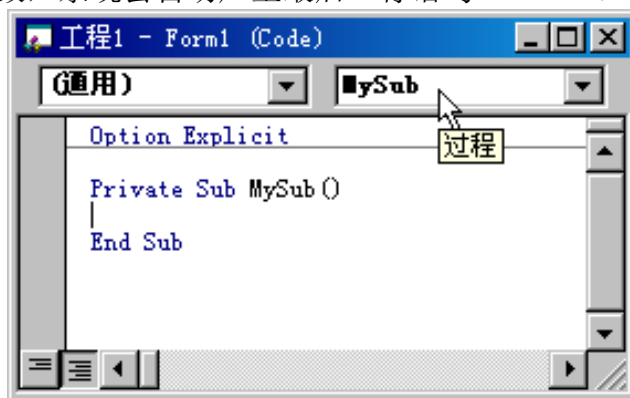


图 8.1 【添加过程】对话框

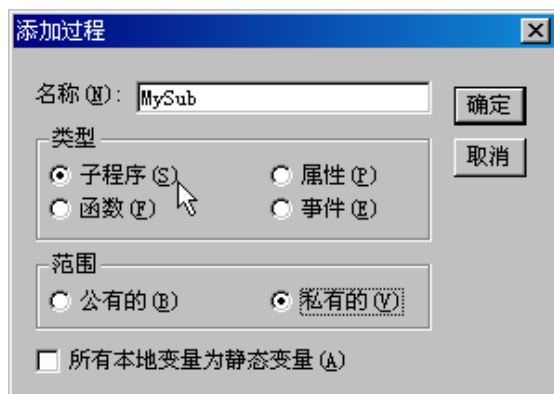


图 8.2 过程模板

8.1.2 子过程的调用

子过程的 Sub 与 End Sub 之间的语句序列称为过程体，每次调用子过程都会执行过程体中的语句。在程序中，既可以调用通用过程，也可以调用事件过程。

1. 调用通用过程

调用子过程有两种方法：使用 Call 语句；直接使用过程名。语法格式如下：

Call 过程名 [(实参表)]

或者：

过程名 [实参表]

说明：

(1) 实参表是实际参数（简称实参）列表，若有多个参数，参数之间要用逗号间隔。各实参与形参在参数列表中的位置相互对应，实参与对应位置的形参必须是同一类型，可以是常数、变量、数组元素或表达式。

(2) 当用 Call 语句调用子过程时，若有参数，则参数必须放在圆括号内；若无参数，则省略过程名后的圆括号。

(3) 若不使用 Call 关键字，则过程名后不能加括号。若有参数，则参数直接跟在过程名之后，参数与过程名之间用空格间隔，参数与参数之间用逗号间隔。

【例 8.1】编写一个计算圆面积和周长的通用过程。在窗体的单击事件中通过 InputBox 函数输入圆的半径，然后调用该过程计算圆面积和周长，计算结果通过消息对话框输出。程序运行结果如图 8.3 所示，代码如下。

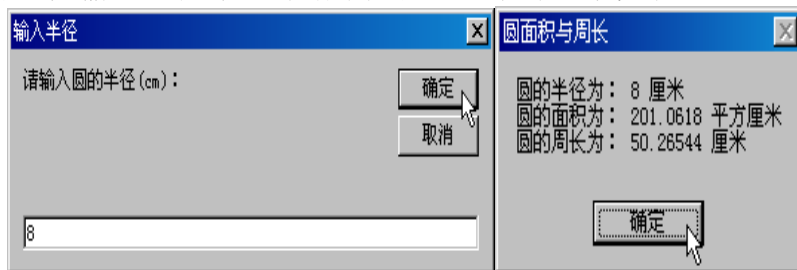


图 8.3 计算圆面积和周长

```
Private Sub Form_Click()  
    Dim r As Single, s As Single, p As Single  
    r = Val(InputBox("请输入圆的半径 cm(): ", "输入半径"))  
    Call CirAreaPeri(r, s, p)  
    MsgBox "圆的半径为: " & r & " 厘米" & vbCrLf _
```

```

        & "圆的面积为：" & s & " 平方厘米" & vbCr _
        & "圆的周长为：" & p & " 厘米", , "圆的面积与周长"
End Sub
Public Sub CirAreaPeri(rr As Single, ss As Single, pp As Single)
    Const PI = 3.14159
    ss = PI * rr * rr
    pp = 2 * PI * rr
End Sub

```

【例 8.2】创建一个简单的文本编辑程序，利用通用过程控制用于编辑操作的命令按钮的有效状态（Enabled 属性）。运行界面如图 8.4 所示。左图为选定文本后的状态，右图为单击“复制”按钮后的状态。

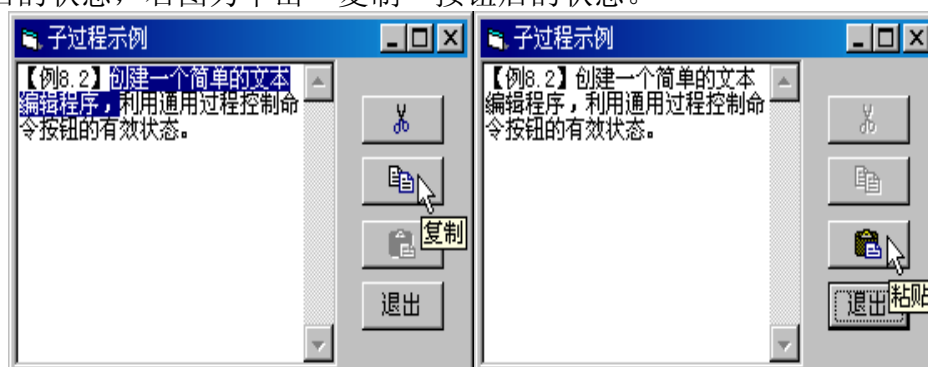


图 8.4 子过程示例

在窗体上放置一个文本框，名称为 txtEdit，设其 Text 属性为空，MultiLine 属性为 True，ScrollBars 属性为 2。添加三个命令按钮，名称分别为 cmdCut、cmdCopy 和 cmdPaste，设 Caption 属性均为空，Enabled 属性均为 False，Style 属性均为 1（图形），通过 Picture 属性为它们各设置一幅图片，设 ToolTipText（工具提示文本）属性分别为“剪切”、“复制”和“粘贴”。再添加一个命令按钮 cmdExit，设其 Caption 属性为“退出”。

编程思路：编制一个通用过程 CtlEnabled(blnEn As Boolean)，形参 blnEn 为逻辑型。由于控件的 Enabled 属性也是逻辑型，因此在程序运行时，根据用户的不同操作，将实参设为 True 或 False，传递给通用过程，即可控制各编辑按钮的 Enabled 属性。

程序代码如下：

```

Option Explicit
Dim str As String
Private Sub Cmdcopy_Click()
    str = TextEdit.SelText
    Call ctlenabled(False)
End Sub
Private Sub Cmdcut_Click()
    str = TextEdit.SelText
    TextEdit.SelText = ""
    Call ctlenabled(False)
End Sub
Private Sub cmdexit_Click()
End

```

```

End Sub
Private Sub Cmdpaste_Click()
    TextEdit.SelText = strs
End Sub
Private Sub ctlenabled(blnen As Boolean)
    Cmdcut.Enabled = blnen
    Cmdcopy.Enabled = blnen
    Cmdpaste.Enabled = Not blnen
End Sub
Private Sub TextEdit_MouseMove(Button As Integer, Shift As
Integer, X As Single, Y As Single)
    If TextEdit.SelText <> "" Then
        Call ctlenabled(True)
    Else
        Call ctlenabled(False)
    End If
    Cmdpaste.Enabled = CBool(Len(strs))
End Sub

```

2. 调用事件过程

在程序中不仅可以调用通用过程，也可以调用事件过程，二者的语法格式相同。

例如：

```

' 执行窗体加载事件过程中的语句
Form_Load      ' 或： Call Form_Load
' 调用命令按钮 cmdAdd 的单击事件过程
cmdAdd_Click   ' 或： Call cmdAdd_Click

```

调用事件过程实际上就是执行事件过程中的语句序列，如同通用过程一样，亦可起到复用和简化代码的作用。

8.2 函数过程

Function (函数) 过程也是独立的过程，可读取参数、执行一系列语句并改变其参数的值，这一点与前面介绍的 Sub 过程相同。Function 过程与子过程不同的是：子过程没有返回值，只能作为独立的基本语句被调用，不能出现在表达式中；而 Function 过程有返回值，既可以出现在表达式中，也可以作为独立的语句被调用。

8.2.1 函数过程的定义

函数过程的定义与子过程的定义相似。不同的是，由于函数过程可以返回一个值，因此要在定义中加入返回值类型说明。语法格式为：

```

[Private|Public] [Static] Function 函数名([形参表]) [As 类型]
    [语句块]

```

```
[函数名=表达式]
[Exit Function]
[语句块]
[函数名=表达式]
End Function
```

说明：

- (1) “函数名”即函数过程的名称，命名规则与变量相同。
- (2) [As 类型] 指定函数过程返回值的类型，可以是 Integer、Long、Single、Double、Currency、String 或 Boolean。若省略，则默认的数据类型为 Variant。
- (3) “表达式”的值是函数返回的结果，通过赋值语句将其赋给函数名。若在函数过程中省略“函数名=表达式”，则该过程返回一个默认值。数值函数过程返回 0，字符串函数返回空字符串。因此，为了能使一个函数过程完成所指定的操作并获取返回值，通常要在过程中为函数名赋值。
- (4) 在过程体中，可以使用一个或多个 Exit Function 语句退出函数。
- (5) 函数过程语法中其他部分的含义与子过程相同。

与子过程一样，可以在“代码编辑器”窗口中直接输入代码来创建函数过程，也可以使用“添加过程”对话框来创建函数过程，只是在选择类型时要选择“函数”。

例如，以下函数 RectArea 可计算并返回矩形的面积：

```
Private Function RectArea(a As Single, _
    b As Single) As Single
    RectArea = a * b
End Function
```

8.2.2 函数过程的调用

函数过程通常在表达式中调用，也可以作为独立的语句被调用。

1. 在表达式中调用

格式：

函数名([实参表])

这种调用方式与大部分 VB 内部函数的调用相同，即将函数名及其实参写在表达式中。例如，以下语句均可调用前面 8.2.1 小节中计算矩形面积的函数过程：

```
S = RectArea(3, 5)
Print "矩形面积为："; RectArea(6, 8)
MsgBox "矩形面积为：" & RectArea(3, 4)
```

2. 以独立语句形式调用

调用的格式与 Sub 过程相同。例如：

```
Call RectArea(2, 7)
RectArea 2, 7
```

当用这种方式调用函数过程时，VB 放弃函数的返回值。

实际上，有些 VB 内部函数也可采用这种忽略返回值的调用方式，此时常将某某函数称为某某语句。其中较典型的内部函数有 MsgBox、Shell 等。

例如，以下语句均可打开 Windows 记事本程序（Shell 函数用于运行一个可执行文件）：

’函数形式

x = Shell("C:\Windows\notepad.exe", 1)

’语句形式

Call Shell("C:\Windows\notepad.exe", 1)

Shell "C:\Windows\notepad.exe", 1

【例 8.3】计算 1~10 阶乘之和。运行结果如图 8.5 所示。

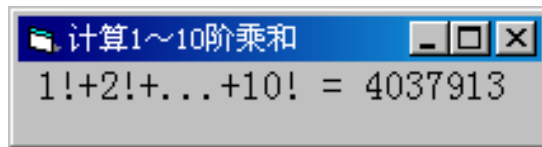


图 8.5 计算 1~10 阶乘之和

思路：首先编制一个求阶乘（N!）的函数过程，在窗体的单击事件中调用此过程，依次求出 1!、2!...10! 的值，并将其累加。

代码如下：

’求阶乘（N!）的函数过程

```
Private Function Factorial(N As Integer) As Long
```

```
    Dim i As Integer, p As Long
```

```
    p = 1
```

```
    For i = 1 To N
```

```
        p = p * i      ’累乘
```

```
    Next
```

```
    Factorial = p      ’对函数名赋值，返回函数值
```

```
End Function
```

```
Private Sub Form_Click() ’单击窗体
```

```
    Dim Sum As Long, i As Integer
```

```
    ’在循环中调用函数过程求 1~10 的阶乘并累加
```

```
    For i = 1 To 10
```

```
        Sum = Sum + Factorial(i)
```

```
    Next
```

```
    Print " 1!+2!+...+10! ="; Sum ’显示结果
```

```
End Sub
```

【例 8.4】求多个数的最大公约数。程序运行结果如图 8.6 所示。

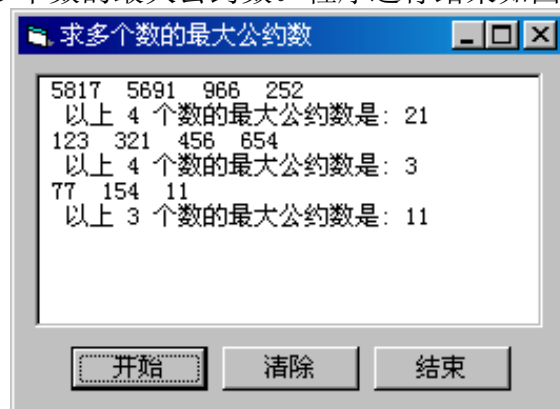


图 8.6 求多个数的最大公约数

思路：首先编写一个用辗转相除法求两个数的最大公约数的函数过程，通过多次调用该函数过程，求出多个数的最大公约数。

具体做法是先求出前两个数的最大公约数，将所得最大公约数与第三个数求最大公约数，以此类推，直至求出所有数的最大公约数。在计算过程中，只要出现最大公约数为 1，即不必再对后续的其他数求公约数。

(1) 设计界面及设置属性

在窗体上添加一个图片框 Picture1，设背景色为白色，AutoRedraw 属性为 True。添加三个命令按钮，Caption 属性分别为“开始”、“清除”和“退出”。

(2) 编写代码

创建一个函数过程 GCD，该函数过程含有两个形参 m 和 n，均为 Long 型，用于传送拟求最大公约数的两个数。在函数过程中用辗转相除法求出最大公约数作为返回值。

在“开始”按钮的单击事件中，定义一个动态数组，用 InputBox 函数获取用户欲求最大公约数的数字个数，以该数值定义动态数组的上界，再通过 For 循环，用 InputBox 函数将每个数存入数组，并在图片框中显示各数。计算多个数的最大公约数时仍然采用 For 循环(循环次数=数字个数-1)，在循环中依次取出动态数组中的数字，调用 GCD 函数过程计算。计算结束后将计算结果显示在图片框中。程序代码如下：

```
Option Explicit
Private Function GCD(m As Long, n As Long) As Long
    Dim r As Long
    r = m Mod n
    Do While r <> 0
        m = n
        n = r
        r = m Mod n
    Loop
    GCD = n
End Function
Private Sub Command1_Click()
    Dim AR() As Long
    Dim n%, i%, n1&, m1&
    n = Val(InputBox("求几个数的最大公约数? "))
    If n < 2 Or n > 20 Then Exit Sub
    ReDim AR(n)
    For i = 1 To n
        AR(i) = Val(InputBox("输入第" & i & "个数: "))
        If AR(i) <= 0 Then
            Picture1.Cls
            Exit Sub
        End If
        Picture1.Print AR(i);
        If Picture1.CurrentX > Picture1.Width * 0.8 Then
            Picture1.Print
```

```

Next
Picture1.Print
n1 = AR(1)
For i = 2 To n
    m1 = AR(i)
    n1 = GCD(m1, n1)
    If n1 = 1 Then Exit For
Next
Picture1.Print "    以上"; n; "个数的最大公约数是: "; n1
End Sub

Private Sub Command2_Click()
    Picture1.Cls
End Sub

Private Sub Command3_Click()
    End
End Sub

```

8.3 参数传递

8.3.1 传值与传址

1. 传值

在定义过程时,如果形参使用关键字 ByVal 声明,则规定了在调用此过程时,该参数将按值传递(传值)。调用过程时,传递给形参的只是调用语句中实参的值,即把调用语句中实参的值复制给子过程或函数过程中的形参。若在被调过程中改变了形参的值,不会影响到实参的值。当被调过程结束并返回调用它的过程后,实参的值还是调用前的值。传值方式可以用一个比喻来说明:假设我有一篇好文章,朋友要看,我把文章复印一份给他,他可以任意在副本上画重点、做标记甚至修改,对我手头的文章不会有任何影响。

2. 传址

在 VB6.0 中,传址是默认的参数传递方式,即形参前不使用任何关键字,相当于用 ByRef 声明形参。传址方式也可以用前面的比喻来说明:传址就像直接把文章借给朋友,如果他在上面画重点、做标记甚至修改,当他把文章还回来时,已经是面目全非了。在调用一个过程时,如果用传址方式进行参数传递,则会将实参的内存地址传递给形参,即让形参和实参使用相同的内存单元。因此,在被调过程中对形参的任何操作都变成了对相应实参的操作,实参的值就会随形参改变。

【例 8.5】分别使用传址和传值两种方式编写实现两数交换的子过程,要求分别显示两种方式下实参与形参的变化。

在窗体上放置两个单选按钮,设 Caption 属性分别为“传址”和“传值”。程序代码如下:

```

Private Sub SwapByRef(x%, y%)
    Dim t As Integer
    Print "交换前形参值: x= "; x; "y="; y
    t = x: x = y: y = t
    Print "交换后形参值: x= "; x; "y="; y
End Sub

Private Sub SwapByval(ByVal x%, ByVal y%)
    Dim t As Integer
    Print "交换前形参值: x= "; x; "y="; y
    t = x: x = y: y = t
    Print "交换后形参值: x= "; x; "y="; y

End Sub

Private Sub Form_click()
    Dim a%, b%
    a = 20: b = 10
    Cls
    Print vbCr; "调用前实参值: a="; a; "b="; b; vbCr
    If Option1.Value Then
        Call SwapByRef(a, b)
    Else
        Call SwapByval(a, b)
    End If
    Print vbCr; "调用前实参值: a="; a; "b="; b;
End Sub

```

运行结果如图 8.7 和图 8.8 所示。比较两图可以看出，采用传址方式时，主调过程（窗体单击事件过程）中的变量 a 和 b 实现了数据交换，而采用传值方式时，a 和 b 无变化。

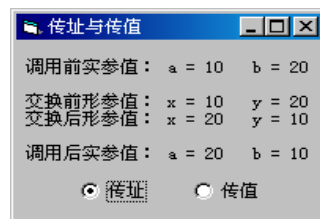


图 8.7 传址方式

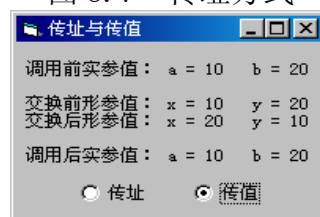


图 8.8 传值方式

8.3.2 对象参数

在 VB 中, 可以将窗体、控件等对象传递给过程。用对象作为参数与用其他数据类型作为参数的过程在语法格式上相同, 只需将参数声明为特定对象类型即可。这里的“对象类型”是指对象所属的类。窗体和控件所属的类可以在属性窗口的对象下拉列表框中看到。在教材图 8.9 所示的属性窗口中, 鼠标指针所指处是名称为 txtChn 的文本框对象所属的类 TextBox。

调用含有对象参数的过程时, 需要将实参设置为对象名称, 该对象必须与形参的类型相同, 并且采用传址方式。

【例 8.6】编制一个计算平均成绩的程序, 当用户输入的分数超出规定范围(0~100)时, 焦点返回出错的文本框并全选其内容, 以便让用户修改或重新输入。运行结果如图 8.10 和图 8.11 所示。



图 8.10 输入错误

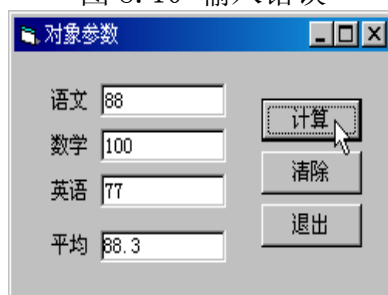


图 8.11 输入正确

(1) 设计界面和设置属性

在窗体上添加三个文本框, 名称分别为 txtChn、txtMath 和 txtEng, 设 Text 属性均为空, HideSelection 属性均为 False。添加四个标签, 均采用默认名称, 设 Caption 属性分别为“语文”、“数学”、“英语”和“平均”。再添加一个标签, 名称为 lblAver, 设 BorderStyle 属性为 1, Caption 属性为空, 背景色为白色, 用于显示平均分。添加三个命令按钮, 名称分别为 cmdCalc、cmdClear 和 cmdExit, Caption 分别为“计算”、“清除”和“退出”。

(2) 编写代码

建立一个通用过程 TxtSetFocus, 形参类型为 TextBox。如:

```
Private Sub TxtSetFocus(txtX As TextBox)
```

在过程中, 形参代表的是被传入的文本框对象。通过该形参可引用文本框对象的所有属性和方法。在“计算”按钮的单击事件中检查用户的输入, 若输入有误, 将出错的文本框作为实参传给通用过程, 使焦点返回该文本框并全选其内容。程序代码如下:

```
Option Explicit
```

```

' 通用过程，焦点返回指定文本框并全选其内容。形参为文本框对象
Private Sub TxtSetFocus(txtx As TextBox)
    txtx.SelStart = 0
    txtx.SelLength = Len(txtx.Text)
    txtx.SetFocus
    MsgBox "输入错误，请重新输入。", vbInformation, "提示"
End Sub
Private Sub CmdCac1_Click()      ' 单击“计算”按钮
    Dim sngChn As Single, sngMath As Single, sngEng As Single
    sngChn = Val(txtChn.Text)
    sngMath = Val(txtMath.Text)
    sngEng = Val(txtEng.Text)
    If sngChn < 0 Or sngChn > 100 Then
        Call TxtSetFocus(txtChn)      ' 以文本框对象为实参调用通用过程
    Exit Sub
    End If
    If sngMath < 0 Or sngMath > 100 Then
        Call TxtSetFocus(txtMath)
    Exit Sub
    End If
    If sngEng < 0 Or sngEng > 100 Then
        Call TxtSetFocus(txtEng)
    Exit Sub
    End If
    lblAver = Format((sngChn + sngMath + sngEng) / 3, "0.##")
End Sub
Private Sub cmdClear_Click()      ' 清除
    txtChn = ""
    txtMath = ""
    txtEng = ""
    lblAver = ""
    txtChn.SetFocus
End Sub
Private Sub cmdExit_Click()      ' 退出
    End
End Sub

```

8.4 过程的应用

8.4.1 过程的作用范围

1. 模块级过程

模块级过程是在某个模块内，用关键字 Private 定义的子过程或函数过程，这种过程只能被本模块内的过程调用，即其作用域为本模块。

2. 全局过程

全局过程是在某个模块内用关键字 Public（或省略范围）定义的子过程或函数过程，这种过程可被整个工程内的所有过程调用，即其作用域为整个工程。

定义全局过程有如下两种方法：

（1）在当前工程中添加标准模块（.bas），在标准模块中用关键字 Public 定义通用过程或函数过程，该过程可以被其他窗体中的过程直接调用。

（2）在某窗体中添加 Public 类型的通用过程，当其他窗体调用此过程时，需要在过程名前添加窗体名，即：

Call 窗体名.过程名

此外，如果定义过程时在过程名前面使用了 Static 关键字，则表示在本过程内声明的局部变量均为静态变量。

8.4.2 标准模块与 Sub Main 过程

1. 标准模块

标准模块保存在扩展名为 .bas 的文件中。在标准模块中用 Public 关键字声明的变量、常数、类型、过程等可以供应用程序中的其他模块和本模块访问。因此，标准模块常被称为过程和声明的容器。使用标准模块能够提高代码的可复用程度。例如，在前面的示例中，通用过程 TxtSetFocus 中未使用具体的对象名称，完全可以从窗体模块中独立出来，放到标准模块中（用 Public 声明）。这样，当程序中有多个窗体时，各窗体模块均可直接调用该过程，而不必重复编写代码。

在工程中添加标准模块的方法为：执行【工程】菜单中的【添加模块】命令，在【添加模块】对话框的【新建】选项卡中选择【模块】，单击【打开】按钮，即可在工程中添加一个默认名称为 Module1 的标准模块。标准模块没有窗体，只有代码，在标准模块中输入代码的方法与窗体模块相同。

2. Sub Main 过程

在较复杂的应用程序中，有时需要在启动时暂不加载任何窗体，先进行一些初始化工作，这就需要在程序启动时执行一个特定的过程。这一过程称为 Sub Main 过程，位于标准模块中。一个工程只能有一个 Sub Main 过程。

将 Sub Main 过程设置为启动对象的方法为：执行【工程】菜单【工程属性】命令，在打开的【工程属性】对话框中选择【通用】选项卡，然后在【启动对象】组合框中选择【Sub Main】并单击【确定】按钮。

例如，以下代码可根据用户输入的登录码决定显示哪一个窗体：

```
Public Sub Main()      ' 本过程为启动对象
    Dim Inp As String
    Inp = InputBox("请输入登录码：")
    If Inp = "123" Then
        Form1.Show
    ElseIf Inp = "456" Then
        Form2.Show
    End If
End Sub
```

8.5 键盘和鼠标事件过程

8.5.1 键盘事件过程

1. KeyPress 事件

当一个对象具有焦点时，用户按下再松开一个可返回 ASCII 码的按键时触发该事件。KeyPress 键盘事件过程语法格式如下：

```
Private Sub 对象_KeyPress([Index As Integer,] KeyAscii As Integer)
```

说明：

- (1) 对象：响应事件的对象。窗体用 Form，其他控件用控件名。
- (2) Index：当对象为控件数组时，此参数是控件数组元素下标。
- (3) KeyAscii：返回按键对应的 ASCII 码（整数）。若改变它的值可以给对象发送一个不同的字符。将它改变为 0 时，将取消按键，对象接收不到字符。
- (4) 该事件可以引用任何可打印的键盘字符，包括大小写字母、数字、标点、运算符以及 Enter、Backspace、Tab 和 Esc 键等。对方向键等不产生 ASCII 码的按键无响应。

(5) KeyPress 键盘事件过程在截取 TextBox 或 ComboBox 控件中的按键时非常有用。它可以立即测试按键的有效性或在字符输入时对其进行格式处理。

【例 8.7】利用 KeyPress 事件限制用户输入合法数据。

本例对 8.3 节中的例 8.6 略作改进，限制用户输入成绩时只能输入数字和小数点，而且小数点只允许输入一次。由于三个文框处理按键的代码相似，所以将其编制为自定义函数过程 NumKey，形参为文本框对象和按键的 ASCII 码。在文本框的 KeyPress 事件中调用该函数，调用形式为：

```
NumKey(文本框对象, KeyAscii)
' 函数过程：限制用户只能输入数字和小数点。
' 形参为文本框对象和按键的 ASCII 码。
Private Function NumKey(txtX As TextBox, _
    KeyX As Integer)
    ' 48、57、46 和 8 分别为 "0"、"9"、"."
    ' 和回删键(Backspace)的 ASCII 码。
    If KeyX >= 48 And KeyX <= 57 Or _
        KeyX = 46 Or KeyX = 8 Then
        NumKey = KeyX
    Else
        NumKey = 0
    End If
    ' 只允许有一个小数点
    If KeyX = 46 And InStr(txtX.Text, ".") _
        > 0 Then NumKey = 0
End Function
```

2. KeyDown 和 KeyUp 事件

当一个对象具有焦点时，按下一个键时触发 KeyDown 事件，松开一个键时触发 KeyUp 事件。KeyDown 和 KeyUp 键盘事件过程语法格式为：

```
Private Sub 对象_KeyDown|KeyUp([Index As Integer,] KeyCode As Integer, Shift As Integer)
```

说明：

(1) KeyCode: 键代码，如 vbKeyF5 (F5 键)，vbKeyHome (Home 键)。KeyCode 用来表示物理键，不区分字母的大小写。要辨别 ASCII 字符，应使用 KeyPress 事件。

(2) Shift: 是一个位域参数，用 3 位二进制数表示键盘事件发生时，是否按下了键盘上的 Shift、Ctrl 和 Alt 键，如教材表 8.1 所示。请注意区分 Shift 参数与 Shift 键。

不同 Shift 码值相加可表示按键组合，如 6 (2+4) 表示同时按下了 Ctrl 和 Alt 键。

(3) KeyDown 和 KeyUp 事件经常用于下列情况：①扩展的字符键，如功能键等；②定位键；③键盘修饰和按键的组合；④区别数字小键盘和常规数字键。

(4) 下列情况不能引用 KeyDown 和 KeyUp 事件：①窗体上有一个 Default 属性为 True 的命令按钮时按 Enter 键；②窗体上有一个 Cancel 属性 True 的命令按钮时按 Esc 键。

例如，下面的语句可判断是否同时按下了 Ctrl+Shift+L 组合键：

```
If KeyCode = vbKeyL And Shift = _  
vbCtrlMask + vbShiftMask Then ...
```

8.5.2 鼠标事件过程

MouseDown、MouseMove 和 MouseUp 事件与 Click 和 DblClick 不同，它们可以区分按下的鼠标按钮以及是否同时按下 Shift、Ctrl 和 Alt 键。

1. 鼠标事件过程语法

鼠标事件过程语法格式如下：

```
Private Sub 对象_MouseDown | MouseMove | MouseUp ([Index As Integer,] Button As Integer, Shift As Integer, X As Single, Y As Single)
```

说明：

(1) 对象：响应事件的对象。窗体用 Form，其他控件用控件名。

(2) Index: 当对象为控件数组时，此参数是控件数组元素下标。

(3) Button: 位域参数，用 3 位二进制数表示哪一个鼠标按钮被按下，如 147 页表 8.2 所示。

(4) Shift: 位域参数，用 3 位二进制数表示鼠标事件发生时，是否同时按下了键盘上的 Shift、Ctrl 和 Alt 键。取值与键盘事件相同。

(5) X、Y: 鼠标在当前对象上的相对坐标，即鼠标指针的位置。

2. MouseDown 和 MouseUp 事件

当按下或释放鼠标按钮时分别发生这两个事件。由于鼠标事件可以区分左右按钮并返回指针坐标，因此这两个事件在鼠标右击操作中和判断鼠标指针位置时特别有用。例如，在许多 Windows 应用程序中，右击某对象会弹出一个快捷菜单，

这就是运用MouseDown或MouseUp事件过程的典型实例。以下语句可以在MouseDown/MouseUp事件过程中判断是否按下/释放了鼠标右键：

```
If Button = vbRightButton Then ...
```

3. MouseMove 事件

此事件在移动鼠标时发生。当鼠标指针处于某个对象的边界内时，该对象能够识别MouseMove事件。应用程序能连续识别大量的MouseMove事件，因此，MouseMove事件过程中的代码不能太复杂，不应去做那些耗时较多的工作。

上述三个鼠标事件常用于绘图操作，详细内容将在第12章介绍。

【例 8.8】用鼠标事件结合Move方法移动控件。要求在程序运行时，按鼠标右键拖动一个命令按钮，使其随鼠标指针移动。

在窗体左上角放置一个命令按钮，设Caption属性为“按住鼠标右键拖动”。代码如下：

```
’窗体级变量存放按下鼠标按钮时的指针坐标
Dim iW As Integer, iH As Integer
’在命令按钮上按下鼠标按钮时触发此事件
Private Sub Command1_MouseDown(Button As _
    Integer, Shift As Integer, _
    X As Single, Y As Single)
    iW = X: iH = Y          ’获取鼠标指针初始坐标
End Sub
’在命令按钮上移动鼠标时触发此事件
Private Sub Command1_MouseMove(Button As _
    Integer, Shift As Integer, _
    X As Single, Y As Single)
    ’若移动鼠标的同时按住鼠标右键
    If Button = vbRightButton Then
        ’用Move方法移动命令按钮
        Command1.Move X + Command1.Left - iW, _
            Y + Command1.Top - iH
    End If
End Sub
```

提示：用同样的方法可以移动没有标题栏(BorderStyle=0)的窗体。

4. 鼠标指针

大部分可视对象具有MousePointer和MouseIcon属性，利用它们可以在鼠标移动到对象的一个特定部分时改变鼠标指针的形状。

MousePointer属性用于设置鼠标指针的类型。设置值0~15是由系统提供的常用类型，如标准指针(0，默认值)、沙漏(11)、十字线(2)、带问号的箭头(14)等；设置值为99时，可以通过MouseIcon属性为鼠标指针指定一个图标文件(.ico或.cur)。

例如，在图8.14中，图片框的MousePointer属性值为14，窗体该属性为默认值。

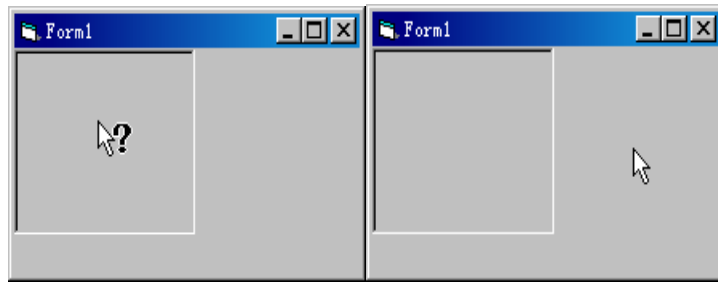


图 8.15 鼠标指针

1. 子过程与函数过程有何区别？
2. 什么是形参？什么是实参？如何使用对象参数？
3. 调用子过程的函数过程时，实参和形参的对应关系如何？应注意些什么问题？

4. 传址和传值的特点是什么？如何选择？
5. 指出下列过程语句中错误，说明原因。

```
Sub MySub(X%) As Integer
Function F1%( F1%)
Sub MySub(ByVal n() As Integer)
```

6. 已知有以下求两个立方数的和 CubeSum 子过程。

```
Private Sub CubeSum( sum%, ByVal x%, ByVal y%)
Sum=x*x*x+y*y*y
End Sub
```

在窗体单击事件过程中有如下变量声明和赋值。

```
Private Sub Form_Click()
Dim a%, b%, c#
A=20: b=40
```

请指出以下各过程调用语句的错误。

```
CubeSum 7, 8, 9
CubeSum c, b, a
CubeSum a+b, a, b
Call CubeSum c, a, b
```

7. 编制随机整数函数，产生 20 个 1~100 之间的随机数。
8. 简述 KeyPress 与 KeyDown 事件、Click 与 MouseDown 事件的区别。

第 9 章 程序调试

本章要点：

- 应用程序可能发生的几类错误
- 程序调试的手段和方法

● 捕获和处理错误的手段和方法

在编写程序中难免会出现错误，从而导致程序不能运行，或能够运行却得不到正确的结果。如何跟踪、避免和解决错误，是程序开发人员面临的不可避免的问题。本章介绍 VB 程序的调试和错误处理。

给出一个简单的程序：输入 6 个整数，求出其中的最大值。解决此问题的正确代码如下，可以将它们放在窗体的 Click 事件中。为了便于说明问题，我们称之为“示例程序”，并在每行前增加了表示行号的数字。在后面的叙述中将多次引用该示例。

```
Private Sub Form_Click()  
1   Dim a(5) As Integer  
2   Dim i As Integer  
3   Dim max As Integer  
4   For i = 0 To 5  
5       a(i) = Val(InputBox("输入一个整数"))  
6   Next i  
7   max = a(0)  
8   For i = 1 To 5  
9       If max < a(i) Then max = a(i)  
10  Next i  
11  For i = 0 To 5  
12      Print a(i)  
13  Next i  
14  Print "MAX="; max  
End Sub
```

9.1 程序可能发生哪几类错误

程序中出现错误可以说是多种多样的，归纳起来可以分为三类：编译错误、逻辑错误和运行异常错误。

9.1.1 编译错误

违背 VB 语法规则，不正确地书写代码，会造成编译错误，这是最常见的错误类型。例如输入了拼写错误的关键字，遗漏了某些必要的标点符号，使用了 For 语句但没有 Next 语句与之对应，调用函数没有提供必要的参数，等等。

VB 提供的“自动语法检测”功能能够自动检测到编译错误，并终止程序的运行。在“示例程序”中，如果在输入第 5 行时，将 InputBox 函数的参数中第二个双引号误输成中文双引号，即第 5 行变为：

```
a(i) = Val(InputBox("输入一个整数"))
```

则当运行程序时就会出现编译错误，VB 自动检测到该错误并弹出错误信息，如图 9.1 所示。



图 9.1 编译错误

【自动语法检测】是 VB 默认选项设置。如果需要修改该选项，可以【选择【工具】菜单中的【选项】，在弹出的对话框的【编辑器】选项卡中改变【自动语法检测】设定。

在程序中不恰当地使用变量，也会引发编译错误。按照 VB 的规定，变量可以不经声明而直接使用，此时变量具有默认的 Variant 类型，对较小的程序而言这通常不会带来麻烦，但当程序规模较大时，变量的使用混乱可能造成错误，且不易被发现。

为此，可以强制进行变量的显式声明，即在程序代码的“通用-声明”段中加入语句：Option Explicit。要使以后新建的窗体均自动加入该语句，可选择【工具】菜单中的【选项】，在弹出的对话框的【编辑器】选项卡中选中【要求变量声明】。强制变量显式声明后，VB 将自动检查是否有未定义的变量，发现后将显示错误信息。

在“示例程序”中的第 2 行定义了循环变量 i。如果已经有了 Option Explicit 语句，再删除第 2 行，则运行程序时会出现图 9.2 所示的错误信息。



图 9.2 变量未定义

9.1.2 逻辑错误

程序运行时没有按照预期的方式去执行，或者没有得到预期的结果，则程序发生了逻辑错误。从语法的角度来看，代码是正确的，运行过程也顺利，但是却产生了不正确的结果，其原因是程序中的处理逻辑出现了错误。

例如，“示例程序”应该求出一个最大值，若将第九行中的“<”误写为“>”，则求出的是最小值，显然这不是我们要求的结果。

要检验程序是否含有逻辑错误，可以人工检查代码，亦可进行程序测试，设定一组特定的甚至是苛刻的操作或数据，测试程序的执行情况和运行结果。

9.1.3 运行异常错误

程序运行时，当一个语句试图执行一个不能执行的操作时，就会发生运行异常错误(实时错误)。例如，某些系统硬件问题，意料之外的数组下标越界，除法

运算中除数为 0，试图读取未准备好的磁盘文件等等，均会引起运行异常错误。

例如，将“示例程序”中第 4 行循环语句改为：

```
For i = 0 To 6
```

则 VB 编译时不会发现其中的错误，而且还能够生成可执行程序（.EXE），但在运行中会出现图 9.3 所示的错误信息。

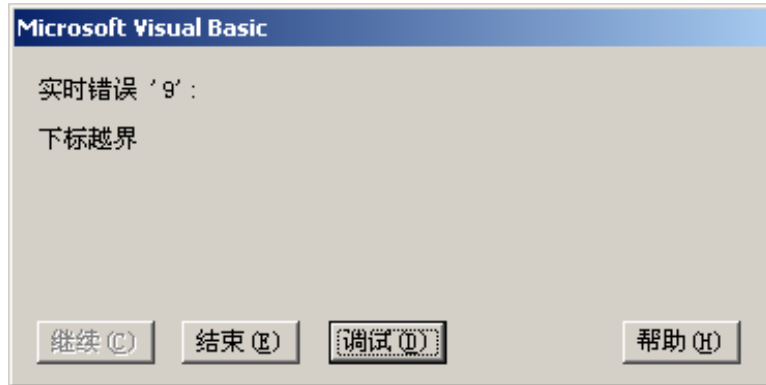


图 9.3 运行异常错误

运行异常错误会导致程序突然异常终止而无法恢复运行，为了避免这种情形的出现，在代码中可以用 VB 的错误处理语句捕获并中断错误，转而执行正确的操作。

在上述三类错误中，编译错误最为简单，也最容易发现和处理，只要根据编译时提供的错误信息进行修改就可以了。只要存在编译错误，应用程序也就不可能运行起来。而对其他两类错误的处理就要复杂的多，需要花一番功夫。本章后面将介绍如何处理逻辑错误和运行异常错误。

9.2 如何调试程序

刚刚设计好的程序可能会含有各种错误，因此调试程序是一个较复杂的工作。能否快速发现问题，判断错误，最后纠正错误，既是程序设计能力的反映，也是实际经验的积累。VB 提供了一系列用于程序调试的有效手段，能够帮助我们深入到应用程序内部去观察。确定到底发生了什么以及为什么会发生，从而逐步缩小问题的范围，确定问题所在。常用的调试手段包括设置运行断点、使用调试窗口、单步调试和跳跃调试等。

9.2.1 设置运行断点

在设计状态，可以改变应用程序的设计和代码，但却不能立即看到这些变更对程序运行所产生的影响；在运行程序时，可以观察到程序的运行状态，但却不能直接改变代码。通过设置运行断点，VB 系统可以中止程序的运行，使得程序进入到中断模式。在中断模式下，系统保留着发生中断时的运行状态，包括各个变量和属性的设置值，供用户观察、分析，同时，允许直接修改应用程序的代码，影响程序的运行。

设置运行断点通常有两种方法：①在代码窗口中单击最左边的灰色区域，使之出现一个棕色●标志，对应的代码行被同时加亮，则此处便设置了一个断点。

②将光标移动到要设置断点的代码行，打开【调试】菜单，选择【切换断点】，亦可设置一个断点。图 9.4 表示设置了两个断点的情况。

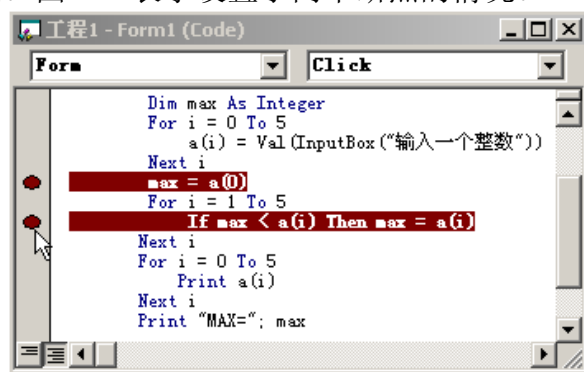


图 9.4 设置断点

要清除已经设置的断点，只需将上述操作重复一次，断点便被撤销。也可以打开【调试】菜单，选择【清除所有断点】。

VB 允许在一行上有多条语句，其间用冒号 (:) 分隔。在这种具有多条语句的行上，断点只被设置在第一条语句上。

另外，在代码中使用 Stop 语句，也可以设置一个断点。

程序运行到所设置的断点时，自动停止运行，并且不执行包含断点的代码行，进入中断模式。此时，将光标移动到某个变量或表达式上，系统会立即显示出该变量或表达式的当前值，如图 9.5 所示。

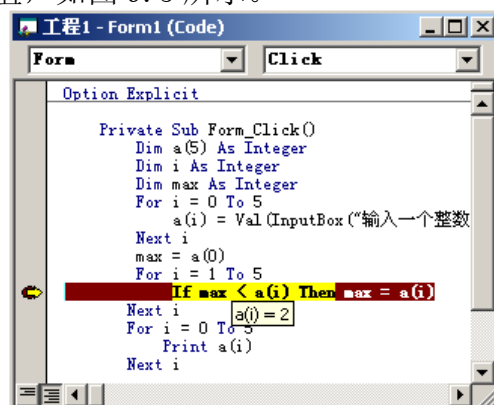


图 9.5 在断点处中断运行

选择【运行】菜单中的【继续】，程序可以继续执行，直到程序结束或再次遇到断点。

通过观察变量或表达式的当前值，了解其在代码前后的变化情况，就可以发现许多存在的问题和错误。某些错误是由于输入代码时的疏忽造成的，例如写错了变量名称、使用了对象不支持的属性或方法等等，因为在中断模式下可以直接修改代码，所以这类错误很容易得到更正。当更正或改变了代码后，继续运行程序便可以验证问题或错误是否得到了解决或纠正。

9.2.2 使用调试窗口

有些问题和错误往往需要通过对数据的变化进行分析才能发现。当程序处于中断模式下时，可以使用三个调试窗口来监视变量或表达式的值，它们是：【立即】窗口、【监视】窗口和【本地】窗口。打开它们的菜单命令均位于【视图】

菜单下。

1. 【立即】窗口

【立即】窗口显示正在调试的代码产生的信息。可以直接在该窗口中键入命令请求这些信息，如图 9.6 所示，也可以在程序中使用 Debug.Print 语句输出某些变量和表达式的值到【立即】窗口。

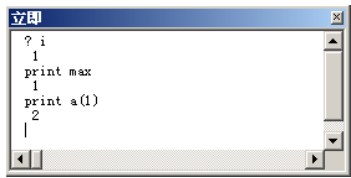


图 9.6 【立即】窗口

【立即】窗口是调试程序时使用最多的窗口。它最容易使用，功能也最强。使用该窗口可以实现以下功能：

(1) 检查某个属性或变量的值，或对表达式求值。例如：

```
? i; a(i); max; a(i) * a(i - 1)
? Text1.Text
```

(2) 为变量或属性设置新值。例如：

```
i = 5: a(i) = 10: Text1.Text = "张三"
```

(3) 测试过程。可以指定参数来调用过程。例如，假设有一个函数过程 Sum(n%)用于求 1~n 之和，可用以下方法测试：

```
a = 10: b = 20
? Sum(a); Sum(b)
```

2. 【监视】窗口

【监视】窗口显示当前的监视表达式的信息，如图 9.7 所示。在【监视】窗口中，【上下文】列出监视表达式所在的过程或模块，只有当前语句在指定的上下文中时，【监视】窗口才能显示出监视表达式的值，否则，其【值】列只显示一条消息，指出语句不在上下文中。

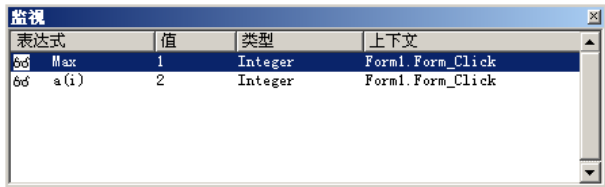


图 9.7 【监视】窗口

要添加一个监视表达式到【监视】窗口，执行【调试】菜单中的【添加监视】命令，打开如图 9.8 所示的【添加监视】窗口。在【表达式】框中输入需要监视的变量或表达式，通过设置【上下文】指定监视的范围，还可以设定监视类型。添加了监视之后，随着程序的执行，【监视】窗口中的变量或表达式会同步更新。

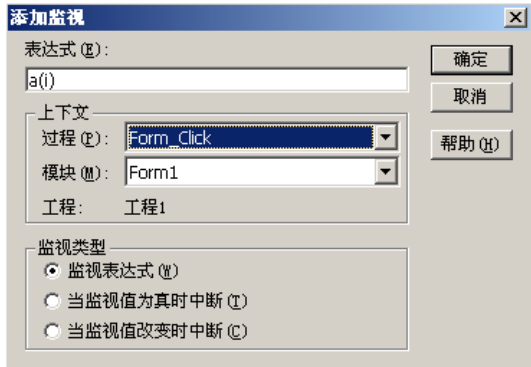


图 9.8 添加监视表达式

3. 【本地】窗口

【本地】窗口显示当前过程中所有变量的值，如图 9.9 所示。当程序从一个过程切换到另一个过程时，【本地】窗口的显示内容会相应改变，它只反映当前过程中可用的变量。

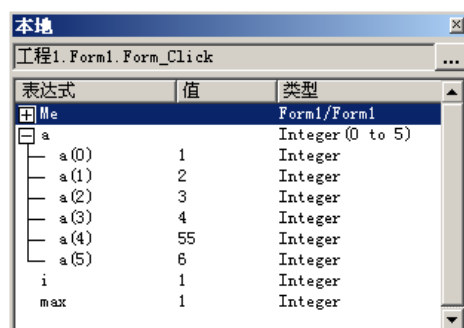


图 9.9 【本地】窗口

9.2.3 单步调试和跳跃调试

程序遇到断点便进入中断模式，因此，中断时含有断点的代码行还没有被执行。如果要观察断点所在的行以及其后语句行的执行情况，可以使用“逐语句”功能进行单步运行。

使用 F8 快捷键，或者使用【调试】菜单中的【逐语句】，都可以进行单步运行。单步运行时，程序每次执行一行代码，然后继续等待，我们便可以观察到该步执行后的运行状态。反复使用“单步运行”功能，就可以对程序执行的每一步进行全面跟踪。

如果当前代码行调用了过程，则利用“逐语句”功能可以跟踪到过程的内部，了解该过程是如何被执行的。

使用 Shift+F8 快捷键，或者使用【调试】菜单中的【逐过程】，也可以进行单步运行。与“逐语句”功能的区别是，当被执行的代码行调用一个过程时，“逐过程”功能不向过程的内部跟踪，它把过程调用视为一个整体单元来执行，然后到达过程执行后的下一条语句。例外的情况是，被调用的过程中含有断点，则即便是“逐过程”方式，程序仍然会在过程内的断点处中断。

使用【调试】菜单中的【运行到光标处】（快捷键 Ctrl+F8），可以使程序直接运行到光标所在的代码行，然后进入中断模式。这相当于连续执行了多次的单步运行，可以快速通过某些不需要调试的代码行。

使用【调试】菜单中的【设置下一条语句】（快捷键 Ctrl+F9），可以不执行部分代码行，直接绕过它们，到达下一个要调试的代码行，该行既可以位于当前断点之后，也可以位于当前断点之前。

9.3 如何捕获和处理错误

对于运行期间程序可能会遇到的错误，在设计代码时就应该考虑到并采取预防措施，以避免程序的异常终止。预防的办法就是添加错误处理程序，捕获并处理产生的错误。由于 Visual Basic 不支持集中处理技术，所以，每一个过程或

事件都应该有一个错误处理程序来解决它自己的错误。

设计错误处理程序包括三个方面：设置错误捕获、编写错误捕获、编写错误处理程序、退出错误处理程序。

9.3.1 设置错误捕获

1. On Error GoTo line 语句

当错误发生时，该语句捕获错误，并使得程序转移到由标号 line 指示的错误处理程序处去执行。相应地，错误处理程序也以相同的标号开始，从而激活错误处理。

标号是一种标识符，其命名方式和变量的命名一样，以字母开始，以冒号结束。On Error 语句中的标号可以带也可以不带冒号，但标号所指示的位置必须是与 On Error GoTo 语句同一过程中的一个语句。

例如，若“ErrorHandler:”是一个标号，则 On Error 语句为如下形式：

```
On Error GoTo ErrorHandler
```

或：

```
On Error GoTo ErrorHandler:
```

2. On Error Resume Next 语句

当错误发生时，使程序转移到发生错误的语句之后的语句，并在此继续运行。这相当于不中断代码的执行，也不转移到别的代码上去执行，而是忽略了所发生的错误。这种情况下不需要编写错误处理程序。

3. On Error GoTo 0 语句

取消错误捕获。对当前过程中的错误捕获由 On Error 语句启动，当退出本过程时，自动取消错误捕获。可以使用 On Error GoTo 0 语句取消对当前过程中的错误捕获。

9.3.2 编写错误处理程序

错误处理程序含有实际处理错误的代码，与 On Error 语句在同一个过程中。它不是一个过程，而是一个程序段，因此也常称之为错误处理例程、错误处理代码。

编写错误处理程序的第一步是添加标号，标号标志着错误处理程序的开始，标号后面必须带有冒号。这个标号也就是在 On Error GoTo 语句中使用到的标号。

通常，将错误处理程序放置在过程的末端，并在标号所在行的上一行中增加语句 Exit Sub 或 Exit Function 等，以便在没有错误发生时，避免执行错误处理代码。

例如，一个含有错误处理程序的通用过程 TestError 形式如下：

```
Sub TestError()  
    On Error GoTo ErrorHandler  
    .....  
    Exit Sub  
ErrorHandler:  
    ' 错误处理代码位于此处
```

.....
End Sub

9.3.3 退出错误处理程序

在错误处理程序内，可使用下列语句退出错误处理程序。

- (1) Resume 语句：使程序返回到导致错误的那条语句上重新执行。
- (2) Resume Next 语句：使程序返回到导致错误的语句之后的那条语句上开始执行。
- (3) Resume line 语句：使程序转移到由标号 line 指示的位置上执行。

通常，若错误在错误处理程序内得以修正，可以使用 Resume 语句；否则，若错误处理程序不能修正所出现的错误，则可以使用 Resume Next 语句。

【例 9.1】设计一个进行除法运算的简单程序，运行界面如图 9.10 所示。在第一个文本框 Text1 中输入被除数，在第二个文本框 Text2 中输入除数，单击命令按钮 CmdAdd (“运算”) 时，在第三个文本框 Text3 中显示结果。

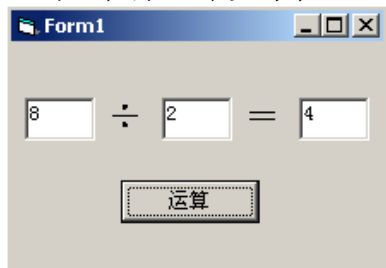


图 9.10 例 9.1 运行结果

代码如下：

```
Private Sub CmdAdd_Click()  
    Text3.Text = Val(Text1.Text) / Val(Text2.Text)  
End Sub
```

正常的运算中一般不会出现任何问题，但当在 Text2 中输入的除数为 0 时，将导致错误，使程序异常终止，显示如图 9.11 所示的错误信息。

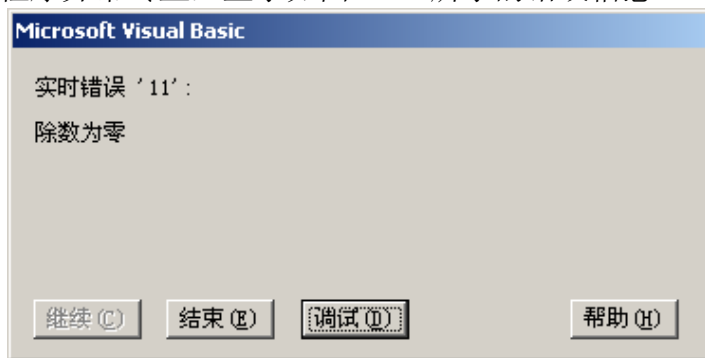


图 9.11 除数为 0 的错误信息

现在考虑错误处理，为上述代码增加错误捕获和处理的功能，改变成如下形式：

```
Private Sub CmdAdd_Click()  
    On Error GoTo Error1 ' 开始捕获错误  
    ' 此语句可能引发错误  
    Text3.Text = Val(Text1.Text) / Val(Text2.Text)
```

```

Exit Sub
Error1: ' 进行错误处理
Text3.Text = "无效"
' 返回到语句 Exit Sub 上执行
Resume Next
End Sub

```

经过上述处理后,当 Text2 中的除数为 0 时,程序不会异常终止,如图 9.12。

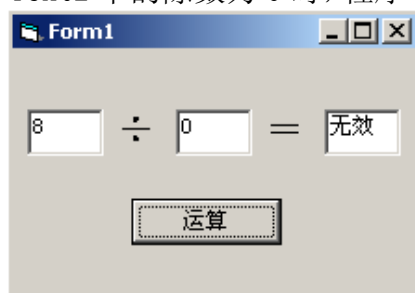


图 9.12 增加了错误处理后的运行情况

【例 9.2】读写软盘中的文件时,如果软驱中没有插入软盘,将引起错误。此时也需要捕获错误并进行处理。

```

Private Sub cmdOpenFile()
    On Error GoTo Error2 ' 开始捕获错误
    ' 此语句可能引发错误
    Open "a:\test.txt" For Output As #1 .....
    Exit Sub
Error2: ' 进行错误处理
    If MsgBox("可能没有插入软盘", vbRetryCancel) _
        = vbRetry Then
        Resume ' 若单击“重试”按钮,返回到 Open 语句重新执行
    End If
End Sub

```

程序运行时,如果确实是因为没有插入软盘而引起的错误,则上述处理可以提示用户,使之有机会插入软盘,而 Resume 语句使和谐返回到 Open 语句上重新执行。

9.3.4 关于 Err 对象

在例 9.2 中,引起错误的原因也许不是软驱中没有插入软盘,其他很多原因都有可能引起软盘读写错误,因此尽管处理了错误,但并不十分有效。要编写有效的错误处理代码,则应该了解 VB 中的 Err 对象。

Err 对象是一个 VB 运行期对象,它包含了关于最新的错误信息,可以帮助确定发生的错误类型、原因和错误发生的地方。当程序运行时遇到一个错误,或者当我们使用 Err 对象的 Raise 方法故意引发一个错误时,系统便设置 Err 对象的属性。

以下简要介绍 Err 对象的主要属性和方法。

1. Number 属性

用于标识错误的错误编号。例如,“6”表示数据溢出,“7”表示内存溢出,

“11”表示除数为0，“9”表示下标越界，“71”表示磁盘未准备好，等等。

2. Source 属性

当前 VB 应用程序的名字。

3. Description 属性

表义性的错误信息。如果某个错误没有这个字符串型的错误信息，该属性就会指明“应用程序定义或对象定义错误”。

4. Clear 方法

在错误处理后清除 Err 对象的所有属性的值。当退出过程时，系统会自动清除 Err 对象的属性值，若要显式地清除，则可调用 Clear 方法。

5. Raise 方法

这个方法用来产生一个错误。它是在测试和评估的时候使用的，这样可以主动地产生错误，以便使程序中的错误处理程序对它进行处理。其简化的语法格式为：

```
Err.Raise number [, source , description]
```

例如，语句 Err.Raise 9 将产生“下标越界”的错误。

使用 Err 对象，例 9.2 的代码可改写如下：

```
Private Sub cmdOpenFile()  
    On Error GoTo Error2  
    Open "a:\test.txt" For Output As #1  
    .....  
    Exit Sub  
Error2:  
    If Err.Number = 71 Then  
        res = MsgBox("没有插入软盘", vbRetryCancel)  
        If res = vbRetry Then  
            Resume  
        Else  
            MsgBox "发生其它错误"  
        End If  
    End If  
End Sub
```

9.3.5 如何避免错误

以下是避免在应用程序中发生错误的几点建议：

1. 写出相关事件以及代码响应每个事件的方法，精心设计应用程序。为每个事件过程和每个通用过程都指定一个特定、明确的目标。

2. 多加注释。用注释说明每个过程的目的、关键语句和代码段的作用、重要变量的用途等，可以避免对它们的错误引用，而且在以后分析代码时，能加深对它们的理解。

3. 尽可能显式地声明和引用对象，即具体指明对象的类型，而不用 Variant 或一般的 Object 数据类型。

4. 造成错误的一个最常见的原因就是输入了不正确的变量名，或把一个控件与另一个控件搞混了。可用 Option Explicit 语句避免变量名或对象名的拼写

错误。

5. 为对象或变量命名时最好采用大小写混合的拼写方式。输入后续代码时则一概用小写字母，当输入引用了某变量或对象的语句并按回车键后，如果所显示的变量或对象名不是定义时的大小写混合格式，则肯定拼写有误。

6. 使用缩排格式可以使代码结构清晰，避免漏掉 End If、Next、End With 等语句。

最后需要说明的一点是，使用 If 语句也可以防止某些可预见的错误的产生。不过，If 语句是防止产生错误，而不是产生错误后如何捕获和处理错误，因此，它对已经产生的错误和无法预见的错误均无能为力。

1. VB 程序中会出现哪三类错误？试举例说明。
2. 编写一个具有错误捕获和处理的函数，将两个数字参数做除法运算。
3. 语句 Debug.Print 将结果显示在哪个窗口上？需要监视一个变量的值应该使用哪个窗口？
4. 使用选择法对 10 个整数从小到大排序，编写程序并对其调试，了解 VB 的程序调试方法。
5. 参考例 9.1 设计除法程序，当错误发生时，利用 MsgBox 函数和 Err 对象，显示出错误类型、原因和错误发生的位置。

[http://msdn2.microsoft.com/zh-cn/library/2x7h1hfk\(VS.80\).aspx](http://msdn2.microsoft.com/zh-cn/library/2x7h1hfk(VS.80).aspx)

第 10 章 设计多功能用户界面

本章要点：

- 菜单设计及应用
- 工具栏设计及应用
- 多文档界面（MDI）的设计
- 应用程序向导的使用

10.1 设计菜单

现在大型应用程序的用户界面绝大多数是菜单界面。菜单栏中包括了各种操作命令。通过不同的菜单标题将命令进行分组，以使用户能够更直观、更容易地访问这些命令。

图 10.1 说明了菜单的组成元素。主菜单栏包含若干主菜单名，每个菜单名下可包含若干菜单项和子菜单名。每个菜单项就是一个命令（对应一个应用程序），菜单项可以有热键（访问键）与快捷键，而菜单名只能有热键。子菜单名又可包含自己的若干菜单项。

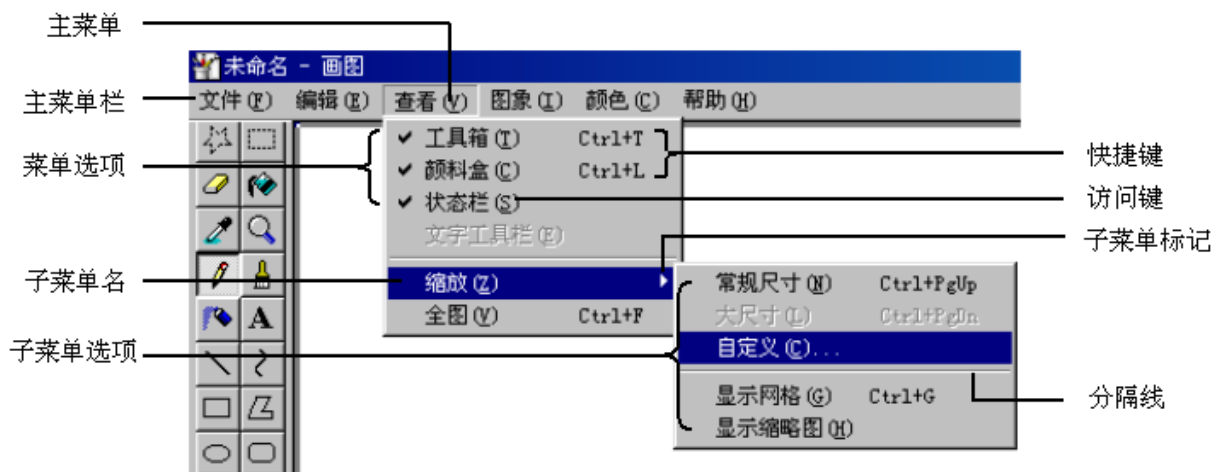


图 10.1 菜单的组成元素

10.1.1 菜单编辑器

VB 提供的【菜单编辑器】是一种用来建立菜单栏的工具，利用它可以非常方便、快捷地在应用程序的窗体上建立菜单。打开【菜单编辑器】对话框的方法有以下几种：

- (1) 选择【工具】菜单中的【菜单编辑器】命令；
- (2) 单击标准工具栏中的【菜单编辑器】按钮；
- (3) 让窗体显示在开发环境中，按 Ctrl+E 键；
- (4) 右击窗体空白处，在快捷菜单中选择【菜单编辑器】命令。

打开【菜单编辑器】对话框后，其界面如图 10.2 所示。



图 10.2 【菜单编辑器】对话框

【菜单编辑器】对话框窗口分为上下两部分。上部用来设置属性，下部则用来显示用户设置的菜单和菜单项。菜单编辑器中各项内容和作用见教材表 10.1。

表 10.1 【菜单编辑器】中各项内容和作用

对话框选项	作用
标题	使用该选项可以输入菜单名或命令名。如果想在菜单中建立分隔条，则应在【标题】文本框中键入一个连字符“—”。建立热键（访问键）可在标题

	后加“&字母”。这是菜单项最重要的两个属性之一
名称	为菜单项输入名字，用于在代码中访问菜单项；它不会出现在菜单中。这是菜单项另一个最重要的属性
索引	可指定一个数字值来确定菜单项在菜单项控件数组中的位置
快捷键	允许为每个命令选定快捷键
帮助上下文 ID	允许为帮助上下文 ID 指定唯一数值。在 HelpFile 属性指定的帮助文件中用该数值查找适当的帮助主题
协调位置	允许选择菜单的协调位置属性。该属性决定是否及如何在容器窗体中显示菜单
复选	允许在菜单项的左边设置复选标记。通常用它来指出切换选项的开关状态
有效	由此选项可决定是否让菜单项对事件做出响应
可见	将菜单项显示在菜单上
显示窗口列表	在 MDI 应用程序中，确定菜单控件是否包含一个打开的 MDI 子窗体列表
左箭头 ←	每次单击都把选定的菜单向左移一个等级。一共可以创建四个子菜单等级
右箭头 →	每次单击都把选定的菜单向右移一个等级。一共可以创建四个子菜单等级
上箭头 ↑	每次单击都把选定的菜单项在同级菜单内向上移动一个位置
下箭头 ↓	每次单击都把选定的菜单项在同级菜单内向下移动一个位置
下一个	选定下一行
插入	在列表框的当前选定行上方插入一行
删除	删除当前选定行
菜单列表框	该列表框显示菜单项的分级列表。将子菜单项缩进以指出它们分级位置或等级
确定	关闭菜单编辑器，并对建立或修改的菜单进行确定
取消	关闭菜单编辑器，取消所有修改

菜单列表框中的每一行都是一个菜单控件，分属不同的等级：菜单标题、菜单项、子菜单标题和子菜单项。菜单控件在列表框中的位置决定了该控件的等级。

(1) 位于列表框中左侧平齐的菜单控件作为菜单标题显示在菜单栏中。

(2) 列表框中被缩进去的菜单项为下拉式菜单选项。

(3) 一个缩进过的菜单控件，如果后面还紧跟着再次缩进的一些菜单控件，它就成为一个子菜单的标题。

10.1.2 利用菜单编辑器创建菜单栏

利用菜单编辑器创建菜单栏实际上就是根据设计的菜单栏结构逐个创建每一菜单项。

1. 创建菜单项

(1) 在标题栏输入该菜单项的文本。

(2) 在名称栏输入程序中要引用该菜单项的名称（类似于控件的 Name）。

(3) 单击【下一个】按钮或【插入】按钮，建立下一个菜单项。

(4) 重复(1)(2)(3)步骤，将菜单项全部建立完毕后，单击【确定】按钮，关闭【菜单编辑器】。

菜单项的属性设置与通常控件属性设置类似。复选（Checked）、有效

(Enabled)、可见(Visible)、上下箭头按钮、左右箭头按钮等的作用见表 10.1。在菜单列表框中，子菜单项标题前比上一级菜单项多“...”标志。

2. 创建分隔线

当一个菜单标题上放置的菜单项较多时，为了直观，可以使用水平线将菜单项分组。建立菜单分隔线的步骤与建立菜单项的步骤相似，惟一的区别就是在菜单编辑器的【标题】框中输入一个连字符“-”。

3. 创建热键与快捷键

建立热键（访问键）的方法与命令按钮相同，即在菜单标题的某个字符前加上一个&符号，在菜单中这一字符会自动加上下划线，表示该字符是一个热键字符。

建立快捷键的方法是打开菜单编辑器中快捷键（Shortcut）下拉式式列表框并选择一个组合键选项，则菜单项标题的右边会显示快捷键名称。

说明：热键指使用 Alt 键+字符键来打开菜单。

【例 10.1】创建一个简易文本编辑器。要求含有表 10.2 所示的菜单栏。

表 10.2 文本编辑器菜单结构

标题	名称	快捷键	标题	名称	快捷键
文件	FileMenu		编辑	EditMenu	
...新建	FileNew	Ctrl+N	...复制	EditCopy	Ctrl+C
...打开	FileOpen	Ctrl+O	...剪切	EditCut	Ctrl+X
...保存	FileSave	Ctrl+S	...粘贴	EditPaste	Ctrl+V
...另存为	FileSaveAs				
...退出	FileExit				

设计方法：在窗体上放置一个通用对话框和一个文本框，然后按表 10.2 设计菜单。

菜单设计完成后，需要为菜单项编写事件过程。本例中我们对【打开】、【保存】、【退出】菜单项编程。程序中通过对话框打开所选定的文本文件，然后将文件内容传送到文本框。保存时，先在文本框中输入内容，然后单击【保存】菜单项，弹出保存对话框，逐步操作即可。关于文件的打开和读写操作可参阅第 13 章。

```
Private Sub FileOpen_Click()  
    On Error GoTo ABC  
    CommonDialog1.Filter = "文本文件|*.TXT"  
    CommonDialog1.CancelError = True  
    CommonDialog1.ShowOpen  
    Text1.Text = ""  
    Open CommonDialog1.FileName For Input As #1  
    Do While Not EOF(1)  
        Line Input #1, inputdata  
        Text1.Text = Text1.Text + inputdata + vbCrLf  
    Loop  
    Close #1  
    i = i + 1  
ABC:  
End Sub
```

```

If i < 6 Then
    bar3.Visible = True
    Load MenuAdd(i)
    MenuAdd(i).Caption = CommonDialog1.FileName
    MenuAdd(i).Visible = True
Else
    t = i Mod 5
    If t = 0 Then t = 5
    MenuAdd(t).Caption = CommonDialog1.FileName
End If
Exit Sub
ABC:
    If Err.Number = 32755 Then Exit Sub
End Sub

Private Sub FileSave_Click()
    ' On Error GoTo qwe:
    ' CommonDialog1.Filter = "文本文件|*.TXT"
    ' CommonDialog1.CancelError = True
    CommonDialog1.ShowSave
    Open CommonDialog1.FileName For Output As #1
    Print #1, Text1.Text
    Close #1
    ' Exit Sub
' qwe:
    ' Exit Sub
End Sub
Private Sub FileExit_Click() ' 退出
    End
End Sub

```

选择【打开】菜单项时的运行界面如图 10.3 所示。

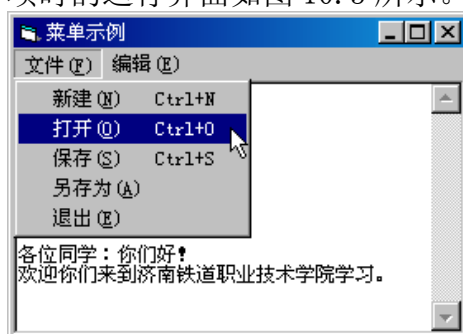


图 10.3 例 10.1 运行界面

10.1.3 运行时修改菜单项

设计时创建的菜单在程序运行时也能动态地改变其设置。例如，如果菜单项

的动作成为不适当时，通过使其失效可防止对该菜单项的选取。如大家所熟悉的剪贴板操作，当剪贴板中没有任何文本时，则【编辑】菜单中的【粘贴】菜单项变暗，因而就不能被选中。

1. 使菜单命令无效

所有的菜单项（也叫菜单控件）都具有 Enabled 属性。当 Enabled 设为 False 时，菜单命令无效使它不响应操作。此时，快捷键的访问也无效。一个无效的菜单控件会变暗。

例如，若要使例 10.1 中【编辑】菜单下的【粘贴】菜单项无效，可用下列语句完成：

```
EditPaste.Enabled = False
```

菜单标题无效将使得整个菜单无效，也就不能访问该菜单标题中的任何菜单项。

例如，语句 EditMenu.Enabled = False 可使例 10.1 中的【编辑】菜单无效。

2. 使菜单控件不可见

在菜单编辑器中，通过选中或不选【可见】(Visible)复选框，可以设置菜单控件的 Visible 属性的初值。

在运行时，要使一个菜单控件可见或不可见，可以从代码中设置其 Visible 属性。

例如：

```
' 使菜单控件数组 0 号元素可见  
mnuFileArray(0).Visible = True  
' 使菜单控件数组 0 号元素不可见  
mnuFileArray(0).Visible = False
```

当一个菜单控件不可见时，菜单中的其余控件会上移以填补空出的空间。如果控件位于菜单栏上，则菜单栏上其余的控件会左移以填补该空间。

使菜单控件不可见也产生使之无效的作用，因为该控件通过菜单、访问键或者快捷键都再无法访问。如果菜单标题不可见，则菜单上所有控件均无效。

3. 在菜单上使用复选标记

可以用 Checked 属性来创建复选标志（√）。设计时通过选取菜单编辑器中的【复选】(Checked)复选框来设置菜单控件 Checked 属性的初始值。在运行时要在一个菜单控件上增加或删除复选标志，可以从代码中设置它的 Checked 属性。例如：

```
' 将菜单项当前的复选状态取反：若有复选标志则删除，若无则添加  
mnuStatus.Checked = Not mnuStatus.Checked
```

10.1.4 动态菜单

在应用程序运行过程当中，可以根据需要动态地增加或减少一些菜单项。这些可以动态增减的菜单项组合就是动态菜单。建立动态菜单必须使用菜单控件数组。

建立菜单控件数组的方法是：在【菜单编辑器】对话框中加入一个菜单项，将其索引 (Index) 项属性设置为 0。然后可以加入名称相同，Index 值有序相连的菜单项。也可以只有一个 Index 为 0 的菜单项，在运行时通过菜单项控件数组

名和索引值,使用 Load 语句加入新的菜单项;使用 Unload 语句删除菜单项。Load 和 Unload 语句格式如下:

Load 菜单控件数组名(Index)

Unload 菜单控件数组名(Index)

【例 10.2】使例 10.1 中的文件菜单能保留最近打开过的文件清单。运行界面如图 10.4 所示。设计方法如下:

以例 10.1 为基础,在文件菜单的【退出】菜单项前面插入一个菜单项 MenuAdd,设【索引】属性为 0,使 MenuAdd 成为菜单数组,设 Visible 属性为 False,再插入一个名为 bar3 的分隔线,Visible 属性亦为 False。在菜单的最后加入名称为 MenuDel,标题为“删除菜单项”的菜单。在【打开】和【另存为】后各插入一个分隔线 bar1 和 bar2。

假定要保留的文件清单限定为 5 个文件名,用 Load 方法向 MenuAdd() 数组加入动态菜单成员,代码如下:

```
I = I + 1          ' 记录文件打开的数量
If I < 6 Then      ' 如果已打开的文件数量<6
    bar3.Visible = True ' 显示分隔线
    Load MenuAdd (I)   ' 装入新菜单项并显示对应文件名
    MenuAdd (I).Caption = CommonDialog1.FileName
    MenuAdd (I).Visible = True
Else
    T = I Mod 5 ' 第 6 个及以后的文件刷新数组控件第 I 项的标题
    If T = 0 Then T = 5
    MenuAdd (T).Caption = CommonDialog1.FileName
End If
```

将上述代码段插入到例 10.1 FileOpen_Click 事件过程内 Close #1 与 Exit Sub 两行语句之间,并设置 I 为模块级变量即可。

要删除所建立的动态菜单项,使用 Unload 方法。本例在菜单项 MenuDel_Click 事件中删除某一菜单项,代码如下:

```
Private Sub MenuDel_Click()
    If I > 0 Then
        If I > 5 Then I = 5      ' 若文件数>5, 设为 5
        Unload MenuAdd(I)       ' 删除菜单项
        I = I - 1
    End If
    ' 若菜单项已全部删除,隐藏分隔线
    If I = 0 Then bar3.Visible = False
End Sub
```

运行界面如图 10.4 所示。



图 10.4 例 10.2 运行界面

10.1.5 弹出菜单

对于使用 Windows 操作系统的新老用户，都很熟悉的一种操作就是右击。无论鼠标指针指向哪一个对象，右击后总能弹出下一个快捷菜单。这种快捷菜单就是这里所说的弹出菜单，也叫浮动菜单。

弹出菜单的设计方法是：先用菜单编辑器设计一个普通菜单，然后用 VB 提供的 PopupMenu 方法来显示弹出菜单。该方法的使用形式是：

[对象.]PopupMenu 菜单名, 标志, X, Y

其中：菜单名是必需的，其他参数是可选的。X、Y 参数指定弹出菜单显示的位置。标志参数用于进一步定义弹出菜单的位置和性能，其取值参见表 10.3。

表 10.3 弹出菜单的位置和性能

标志类型	常数	值	说明
位置	vbPopupMenuLeftAlign	0	X 位置确定弹出菜单的左边界（默认）
	vbPopupMenuCenterAlign	4	弹出菜单以 X 为中心
	vbPopupMenuRightButton	8	X 位置确定弹出菜单的右边界
性能	vbPopupMenuLeftButton	0	只能用鼠标左键触发弹出菜单（默认）
	vbPopupMenuRightButton	2	能用鼠标左键和右键角发弹出菜单

说明：选择位置值和性能值时，将其用“或”(Or)运算符进行组合。PopupMenu 方法应结合 MouseDown 或 MouseUp 事件过程来使用。该方法也可用于选定一个子菜单名。

如果不希望弹出菜单的菜单项出现在一般菜单栏里，只需将菜单的 Visible 属性设置为 False，即在菜单编辑器内不选中【可见】复选框。当使用 PopupMenu 方法时，它可以忽略 Visible 的设置。

例如，在例 10.2 中要加入有关“编辑”这部分菜单的弹出菜单功能，用鼠标右击 Text1 时能弹出 EditMenu 菜单中的菜单项，并以鼠标指针坐标 X 为弹出菜单的中心，可使用如下代码：

```
Private Sub Text1_MouseDown(Button As Integer, _  
    Shift As Integer, X As Single, Y As Single)  
    If Button = 2 then PopupMenu EditMenu, 4, X, Y  
End Sub
```

这里，Button=2 表示按下鼠标右键，EditMenu 为编辑菜单名，4 指定弹出菜单的位置。运行界面如图 10.5 所示。

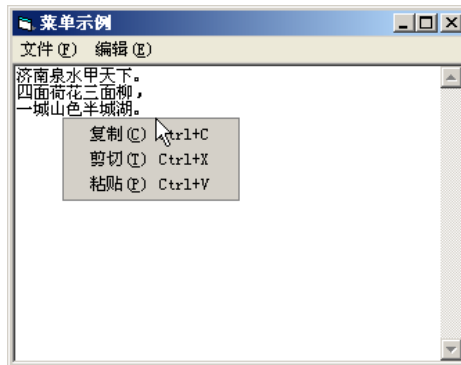


图 10.5 弹出菜单的运行界面

10.2 设计工具栏

在基于 Windows 操作系统的应用程序中，一般都是将常用的命令以按钮的形式集合在一起，以方便用户的操作，这就是工具栏。工具栏为用户提供了对于应用程序中最常用的菜单命令的快速访问，进一步增强了应用等量齐观的菜单界面。制作工具栏有两种方法：一是手工制作，即利用图片框和命令按钮，比较麻烦；另一种方法是通过组合使用 Toolbar 和 ImageList 控件来建立，这种方法简单、快捷，容易学习。

Toolbar 和 ImageList 控件都是 ActiveX 控件，使用这些控件前必须先将其添加到工具箱。添加的方法有两种：

(1) 选择【工程】菜单→【部件】命令→弹出对话框，在对话框的【控件】选项卡中选中 Microsoft Windows Common Control 6.0 选项，单击【确定】。

(2) 用鼠标右击工具箱，弹出快捷菜单，选【部件】命令。后续操作同上。

执行上述操作后，工具箱中将添加如图 10.6 所示的 9 个图标，Toolbar 和 ImageList 控件即在其中。



图 10.6 添加到工具箱中的 ActiveX 控件

创建工具栏的步骤如下：

(1) 将 ImageList 控件添加到窗体上，然后在 ImageList 控件中添加所需的图像。

(2) 将 Toolbar 控件添加到窗体上，在 Toolbar 控件中创建 Button (按钮)对象。

(3) 在 ButtonClick 事件中用 Select Case 语句对各按钮进行相应的编程。

在多文档界面 (MDI) 应用程序的开发中，工具栏应放在 MDI 父窗体中。

10.2.1 在 ImageList 控件中添加图像

ImageList 控件包含了一个图像的集合，它专门用来为其他控件提供图像库。特别是 ListView, TreeView, TabStrip 和 Toolbar 等控件都是从其中获取图像。在利用 Toolabar 控件制作工具栏时，其中按钮的图像就是从 ImageList 的图像库中获得。

在窗体上添加 ImageList 控件后，其默认名为 ImageList1，右击该控件，从弹出菜单中选择【属性】，然后在【属性页】对话框选择【图像】选项卡，见图 10.7。

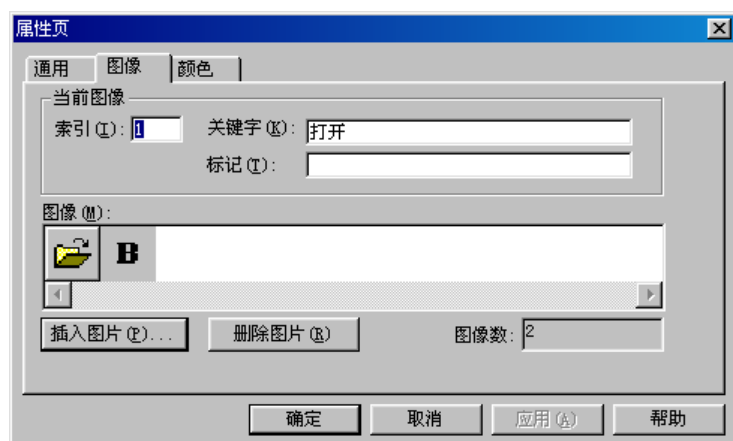


图 10.7 ImageList 属性页【图像】选项卡

其中：

【索引】：表示每个图像的编号，在 Toolbar 的按钮中引用。

【关键字】：表示每个图像的标识名，在 Toolbar 的按钮的引用。

【图像数】：表示已插入的图像数目。

【插入图片】按钮：插入新图像，图像文件的扩展名为 .ico、.bmp、.gif、.jpg 等。

【删除图片】按钮：删除选中的图像。

向 ImageList 中添加图像的具体操作是：单击【插入图片】按钮，这时会弹出【选定图片】对话框，通过对话框选定需要的一个图像文件，再单击【选定图片】对话框中的【打开】按钮，然后赋予该图像一个编号和一个关键字。接着再单击【插入图片】按钮，重复上述过程，直到添加完毕，最后单击 ImageList 属性页中的【确定】按钮。

10.2.2 在 Toolbar 控件中添加按钮

在 Toolbar 工具栏中可以建立多个按钮。每个按钮的图像均来自 ImageList 控件中插入的图像。

1. 为工具栏连接图像

在窗体上添加 Toolbar 控件后，右击该控件，在快捷菜单中选择【属性】，打开【属性页】对话框，选择【通用】选项卡，见图 10.8。

在【图像列表】下拉列表框中选择 ImageList 控件（如 ImageList1）。

说明：若要对 ImageList 控件增、删图像，必须先在【图像列表】下拉列表框内设置<无>，即与 ImageList 切断联系。否则无法对 ImageList 控件进行设置。

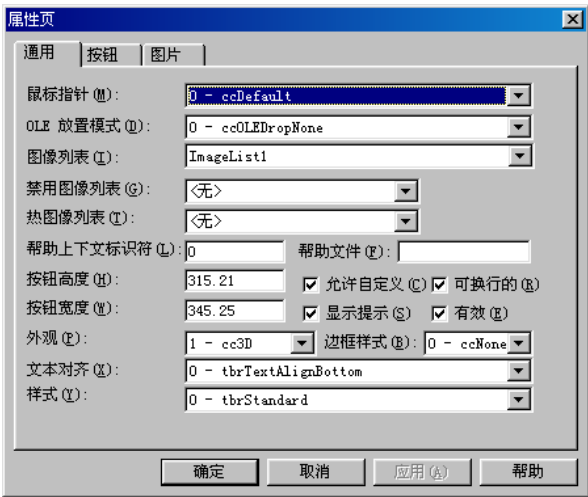


图 10.8 Toolbar 属性页【通用】选项卡

2. 为工具栏增加按钮

在 Toolbar 属性页选择【按钮】选项卡，打开如图 10.9 所示的该选项卡界面。单击【插入按钮】，可以在工具栏中增加按钮。

Toolbar 属性页【按钮】选项卡中的主要属性有：

【索引】(Index) 文本框：表示每个按钮的索引号。在 ButtonClick 事件中引用。

【关键字】(Key) 文本框：表示每个按钮的标识名。在 ButtonClick 事件中引用。

【样式】(Style) 下拉式列表框：提供 6 种按钮样式。见表 10.4。

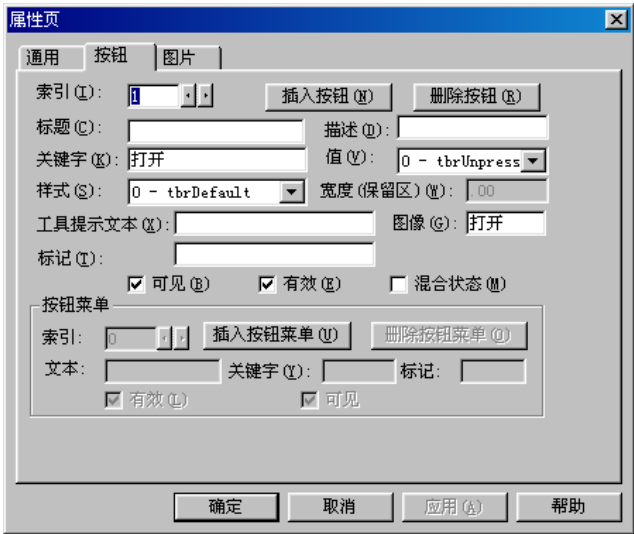


图 10.9 Toolbar 属性页【按钮】选项卡

表 10.4 工具栏按钮的样式 (Style)

常数	值	说明
tbrDefault	0	(默认的) 按钮。按钮是一个规则的下压按钮。
tbrCheck	1	复选。按钮是一个复选按钮，它可以被选定或者不被选定。
tbrButtonGroup	2	按钮组。在任何时刻都只能按下组内一个按钮。
tbrSeparator	3	分隔符。按钮的功能是作为固定宽度 (8 个像素) 的分隔符。
tbrPlaceholder	4	占位符。按钮在外观和功能上象分隔符，但宽度可改变。
tbrDropDown	5	菜单式下拉按钮。用于查看 MenuButton 对象。

【图像】(Image) 文本框：表示 ImageList 对象中的图像，它的值可以是图像的关键字 (Key) 或索引 (Index)。

【值】(Value) 下拉式列表框：表示按钮的状态。有按下 (tbrPressed) 和没按下 (tbrUnpressed)，对样式 1 和样式 2 有用。

向 Toolbar 中添加按钮及为按钮添加图像的具体操作是：单击【插入按钮】按钮，然后为该按钮赋予一个关键字，再设置【样式】和【工具提示文本】，并在【图像】文本框中为按钮设置一个图像值 (它的值可以是 Key 或 Index 的值)；接着再单击【插入按钮】按钮，重复上述过程，直到添加完毕，最后单击 Toolbar 属性页中的【确定】按钮。若需在按钮上显示文字可设置【标题】。

10.2.3 为工具栏按钮编写事件过程

工具栏创建完成后，还要编写相应的代码，这样按钮才能起作用。

Toolbar 控件常用的事件有两个：ButtonClick 和 ButtonMenuClick。前者对应按钮样式属性为 0~2，后者对应样式为 5 的菜单按钮。

实际上，工具栏上的按钮是对象数组。单击工具栏上的按钮会发生 ButtonClick 或 ButtonMenuClick 事件。可以利用数组的索引 (Index 属性) 或关键字 (Key 属性) 来识别被单击的按钮，再使用 Select Case 语句完成代码编写。现以 ButtonClick 事件为例写出其事件过程的编写结构。

假设工具栏上有“新建”、“打开”等命令按钮，则用不同方法编程如下：

(1) 用索引 Index 确定按钮

```
Private Sub Toolbar1_ButtonClick(ByVal Button _
    As MSComctlLib.Button)
    Select Case Button.Index
        Case 1
            ' “新建” 语句块
        Case2
            ' “打开” 语句块
        .....
    End Select
End Sub
```

(2) 用关键字 Key 确定按钮

以下程序段与上面基本相同，仅用 Button.Key 代替 Button.Index。

```
Private Sub Toolbar1_ButtonClick(ByVal Button _
```

```

        As MSComctlLib.Button)
Select Case Button.Key
    Case "ToolNew"
        ' “新建” 语句块
    Case "ToolOpen"
        ' “打开” 语句块
    .....
End Select
End Sub

```

10.2.4 菜单与工具栏综合应用举例

【例 10.3】编制文字处理程序。要求有菜单栏和工具栏，运行界面如图 10.10 所示。

设计方法：先用菜单编辑器在窗体上建立菜单栏。然后通过组合使用 Toolbar 和 ImageList 控件创建工具栏。制作工具栏的步骤如下：①在窗体上放置一个 ImageList 控件；②在 ImageList 控件中添加所需的图像；③在窗体上放置一个 Toolbar 控件；④将 Toolbar 与 ImageList 关联，在 Toolbar 中创建 Button 对象，然后将 ImageList 中的图像添加给 Button 对象。

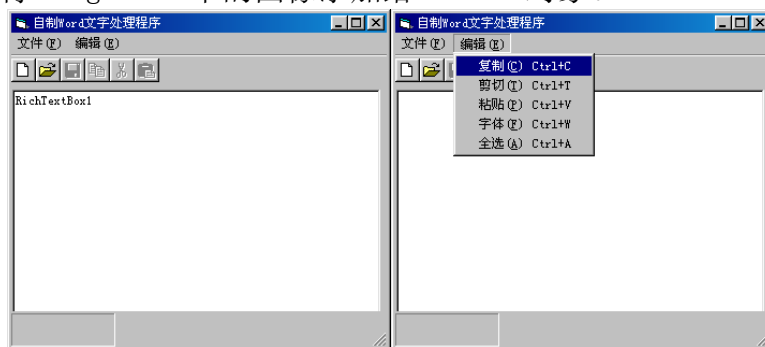


图 10.10 例 10.3 的不同运行界面

工具栏各按钮的图标及其排列为：   ，索引号（Index）为 1～6。建立工具栏后再在窗体上添加一个 RichTextBox 控件、一个通用对话框、一个状态栏（StatusBar）。上述各控件均采用默认名称。将窗体的 Caption 属性设为“自制 Word 文字处理程序”。最后为菜单项和工具栏按钮编写代码。

说明：RichTextBox 控件是一种 ActiveX 控件，可用于输入和编辑文本，并可设置丰富的格式。使用前需将其添加至工具箱：选择【工程】|【部件】菜单命令，弹出对话框，在【控件】选项卡中选择 Microsoft Rich TextBox Control 6.0，单击【确定】按钮。

菜单栏的设置如教材表 10.4 所示，代码详见教材。

10.3 多文档界面

10.3.1 多文档界面

多文档界面由父窗体和子窗体组成。父窗体也称 MDI 窗体，是子窗体的容器。子窗体亦称文档窗体，用来显示各自文档。多文档界面允许用户同时打开多个文档，并可在不同文档间快速切换。所有子窗体具有相同的功能，且所有子窗体都包含在 MDI 窗体中。

多文档界面主要特性如下：

(1) 所有子窗体均显示在 MDI 窗体的工作区中。用户可改变、移动窗体的大小，但被限制在 MDI 窗体中。

(2) 当最小化子窗体时，它的图标将显示于 MDI 窗体上而不是在任务栏中。当最小化 MDI 窗体时，所有的子窗体也被最小化。只有 MDI 窗体的图标出现在任务栏中。

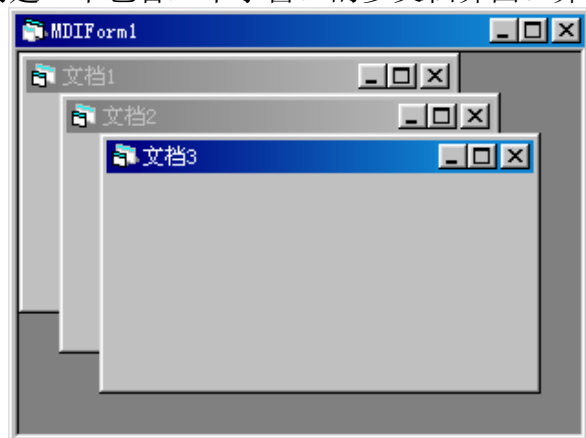
(3) 当最大化一个子窗体时，它的标题与 MDI 窗体的标题一起显示在 MDI 窗体的标题栏上。

(4) MDI 窗体和子窗体都可以有各自的菜单栏，子窗体加载时覆盖 MDI 窗体的菜单。

10.3.2 建立多文档界面

MDI 应用程序至少应有两个窗体：父窗体和一个子窗体。父窗体只能有一个，子窗体则可以有多个。子窗体就是 MDIChild 属性设置为 True 的普通窗体。

【例 10.4】创建一个包含三个子窗口的多文档界面。界面如图 10.11 所示。



生成 MDI 应用程序的具体操作步骤如下：

(1) 从【工程】菜单中选择【添加 MDI 窗体】菜单命令，系统打开【添加 MDI 窗体】对话框，选择【新建 MDI 窗体】图标，单击【打开】按钮，即完成创建 MDI 窗体。

(2) 将 MDI 窗体的 Caption 属性设为“MDI 窗体”。

(3) 创建一个新的普通窗体（或者打开一个已存在的普通窗体），将其 Caption 属性设为“文档 1”，并将该窗体的 MDIChild 属性设置为 True，则该窗

体变为一个子窗体。

(4) 重复步骤 3，创建子窗体“文档 2”和“文档 3”。

(5) 从【工程】菜单中选择【属性】菜单项，打开【工程属性】对话框，设置 MDI 窗体为启动窗体。

(6) 编写代码，在 MDI 窗体加载事件中显示所有子窗体：

```
Private Sub MDIForm_Load()  
    Form1.Show  
    Form2.Show  
    Form3.Show  
End Sub
```

VB 会自动将子窗体和父窗体相联系。子窗体只能在父窗体中打开。运行时，如果单击子窗体的最大化按钮，两个窗体将重合，窗体的标题变为父窗体的标题加上子窗体的标题。也可以将子窗体部分地移出父窗体，此时父窗体会自动加上相应的滚动条，并且子窗体的移出部分不予显示。

上述介绍的是预先建立了三个子窗体然后显示。实际上，也可在设计时先建一个子窗体作为模板，然后在运行时通过对象变量动态地创建多个子窗体。

假设先建的子窗体名称为“frmChild”，则具体实现的程序代码如下：

```
Private Sub MDIForm_Click() '单击 MDI 窗体  
    Dim frmX As New frmChild 'frmX 是对象变量  
    Static S As Integer  
    S = S + 1  
    frmX.Caption = "第" & S & "个子窗体"  
    frmX.Show  
End Sub
```

说明：使用 New 关键字声明一个对象变量相当于创建了该类对象的一个新实例。

当一个 VB 应用程序具有父窗体、子窗体和普通窗体时，其工程资源管理器窗口显示如图 10.12 所示。

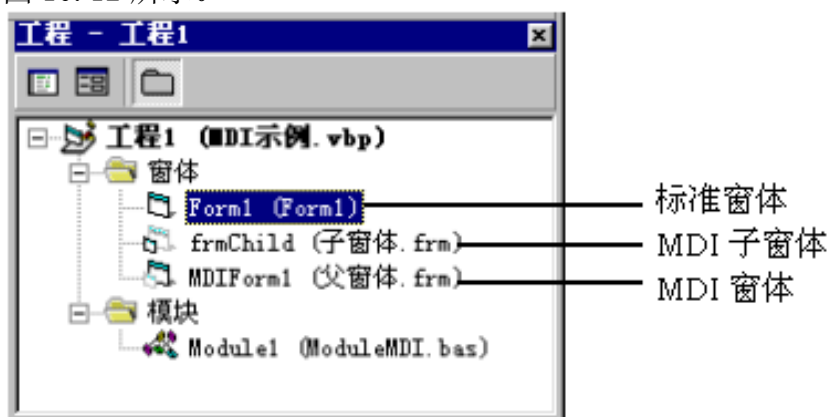


图 10.12 父窗体、子窗体和普通窗体

10.3.3 创建 MDI 应用程序的菜单

通过给 MDI 窗体和子窗体添加菜单控件，可以为 VB 应用程序创建菜单。为

MDI 应用程序创建菜单的思路或方法为：把希望在任何时候都显示的菜单控件放在 MDI 窗体上(即使没有子窗体可见时)。当运行该应用程序时，如果没有可见的子窗体，会自动显示 MDI 窗体菜单。把应用于子窗体的菜单控件放置到子窗体中。在运行时，只要有一个子窗体可见，这些菜单标题就会显示在 MDI 窗体的菜单栏中。为父窗体和子窗体设计菜单与普通窗体的菜单设计方法一样。

10.3.4 创建“窗口”菜单

大多数 MDI 应用程序(例如基于 Windows 的 Word、Excel 等)都设计了“窗口”菜单。这是一个显示所有打开的子窗体标题的特殊菜单。另外，有些应用程序将操纵子窗体的命令，比如“层叠”、“平铺”和“排列图标”等，也都放在这个菜单中。

在 MDI 窗体或子窗体上的任何菜单控件，只要将其 WindowList 属性设为 True，均可用于显示打开的子窗体清单。在运行时，VB 自动管理和显示标题清单，并在当前有焦点的子窗体对应的菜单标题前显示复选标志，同时在窗口清单上方自动添加一个分隔线。

要设置 WindowList 属性，请按照以下步骤执行：

(1) 选择适当的 MDI 窗体或 MDI 子窗体，执行【工具】菜单【菜单编辑器】命令。注意：WindowList 属性只应用于 MDI 窗体和 MDI 子窗体，它对标准窗体(非 MDI)不起作用。

(2) 在菜单编辑器列表框中，选择或建立所需的菜单，如“窗口”菜单。

(3) 选中【显示窗口列表】复选框。在运行时，这个菜单显示打开的子窗体的清单。另外，该菜单控件的 WindowList 属性返回 True。

10.3.5 排列子窗体

通常，MDI 应用程序含有“窗口”菜单，用于显示所有打开的子窗体标题。对于窗体或子窗体图标的层叠、平铺和排列图标命令通常也放在“窗口”菜单上。

在 MDI 窗体中使用 Arrange 方法来重新对齐子窗体，可以层叠，水平平铺、垂直平铺或沿着 MDI 窗体下部排列子窗体图标等方式来显示子窗体。Arrange 方法形式如下：

MDI 窗体对象.Arrange 排列方式

排列方式取值如下：

0-vbCascade：层叠所有非最小化 MDI 子窗体。

1-vbTileHorizontal：水平平铺所有非最小化的 MDI 子窗体。

2-vbTileVertical：垂直平铺所有非最小化的 MDI 子窗体。

3-vbArrangeIcons：排列所有子窗体图标。

下面的代码给出了“层叠”、“平铺”和“排列图标”菜单项的 Click 事件过程。

```
Private Sub mnu 层叠_Click()      ' 层叠子窗体
    MDIForm1.Arrange vbCascade
End Sub
Private Sub mnu 平铺_Click()      ' 平铺子窗体
```

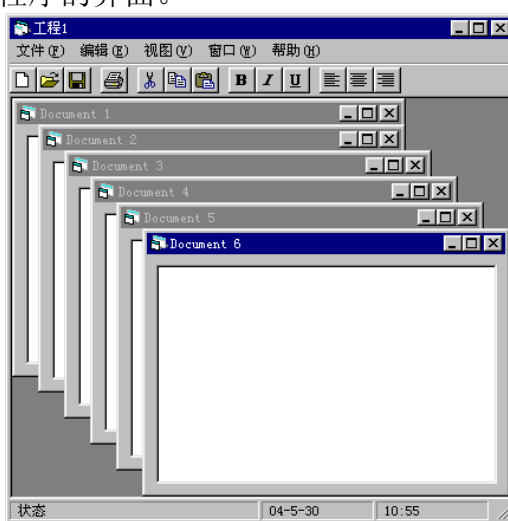
```

MDIForm1.Arrange vbTileHorizontal
End Sub
' 排列所有子窗体图标
Private Sub mnu_排列_Click()
    MDIForm1.Arrange vbArrangeIcons
End Sub

```

10.4 使用应用程序向导

为了提高应用程序开发的效率，VB 提供了“VB 应用程序向导”这一非常方便的程序生成器，用来生成一个应用程序的界面，图 10.13 是用该向导生成的一个多文档编辑器应用程序的界面。



执行【文件】菜单【新建工程】命令，打开如图 10.14 所示的【新建工程】对话框。在对话框中选择【VB 应用程序向导】并单击【确定】按钮，即可在向导的带领下快速设计应用程序的界面。



在向导的带领下，主要完成以下操作：

- ◆ 选择应用程序界面
- ◆ 选择菜单和子菜单项
- ◆ 选择工具栏按钮
- ◆ 生成 WWW 浏览器

1. 菜单名与菜单项有何区别？热键与快捷键有何区别？
2. 建立动态菜单的关键是使用什么？
3. 设计弹出菜单使用何种方法实现？
4. Toolbar 与 ImageList 的作用分别是什么？如何使他们连接？
5. 简述制作菜单的过程。
6. 简述制作工具栏的过程。
7. 使用应用程序向导创建一个多文档界面的应用程序。

第 11 章 实用扩展控件

本章要点：

- RichTextBox 控件
- TreeView
- ListView 控件
- SSTab 控件

11.1 RichTextBox 控件

概述

RichTextBox 控件又称为多格式文本框，使用该控件不仅可以输入和编辑文本，还可以对控件中任何部分的文本设置不同的格式，如对选定文本设置字体、字号、字形、颜色、下划线、删除线等。此外，在该控件中还可以设置左右缩进和悬挂式缩进等段落格式，插入图片，并以 RTF 和纯文本两种格式打开和保存文件。

加载 RichTextBox 控件的方法：右击工具箱，在弹出菜单中选择【部件】菜单项，打开【部件】对话框，在【控件】选项卡的列表中选中“Microsoft Rich Textbox Control 6.0”前面的复选框，单击【确定】按钮。此时工具箱中将增加该控件的图标



11.1.1 设置字体格式

下面通过实例说明如何设置 RichTextBox 控件中选定文本的字体格式。

【例 11.1】利用字体对话框设置 RichTextBox 控件中选定文本的字体格式。

新建工程，在窗体上添加一个 RichTextBox 和一个 CommonDialog 控件，均采用默认名称。将 RichTextBox 控件的 ScrollBars 属性设为 2。再添加一个命令按钮，名称为 cmdFont，Caption 属性为“字体”。以下是按钮单击事件过程的代码：

```
Private Sub cmdFont_Click()  
    On Error GoTo Quit  
    With CommonDialog1 ' 设置通用对话框相关属性  
        ' 显示所有字体和效果选项  
        .Flags = cdlCFBoth Or cdlCFEffects  
        ' 设置对话框默认字体名称  
        If .FontName = "" Then .FontName = "宋体"  
        ' 对用户单击“取消”按钮做出响应  
        .CancelError = True .ShowFont ' 打开字体对话框  
    End With  
    ' 设置 RichTextBox 控件中选定文本字体格式  
    With RichTextBox1  
        ' 字体名称(字符串型)  
        .SelFontName = CommonDialog1.FontName  
        .SelFontSize = CommonDialog1.FontSize ' 字号(整型)  
        .SelBold = CommonDialog1.FontBold ' 粗体(布尔型)  
        .SelItalic = CommonDialog1.FontItalic ' 斜体(布尔型)  
        ' 下划线(布尔型)  
        .SelUnderline = CommonDialog1.FontUnderline  
        ' 删除线(布尔型)  
        .SelStrikeThru = CommonDialog1.FontStrikethru  
        .SelColor = CommonDialog1.Color ' 颜色(长整型)  
    End With  
Quit:  
End Sub
```

在上述代码中，首先利用通用对话框控件打开字体对话框，用户在对话框中设置格式并确认后，通过代码中的第二个 With...End With 语句块将 RichTextBox 控件中的选定文本格式设置为由字体对话框返回的各种格式。代码中 RichTextBox 控件的 7 个以“Sel”为前缀的属性（代表选定文本的各种格式）分别由字体对话框的对应属性赋值。代码中的注释说明了各属性的含义。程序运行效果如图 11.1 所示。

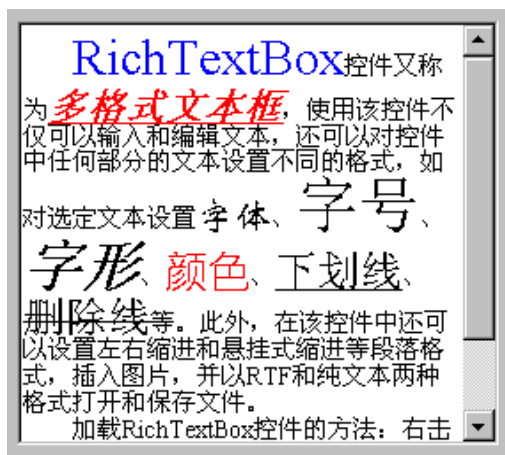


图 11.1 设置字体格式

11.1.2 设置段落格式

1. 段落缩进

RichTextBox 控件的 SelIndent、SelRightIndent 和 SelHangingIndent 属性分别用于设置选定段落的左缩进、右缩进和悬挂缩进，均为整型数值。缩进量的单位与窗体的 ScaleMode 属性有关，默认单位为缇（1 厘米=567 缇）。

【例 11.2】设置段落缩进。在例 11.1 中的窗体上增加一个按钮，Caption 属性为“左缩进”，在该按钮的单击事件过程中加入以下代码：

```
Dim sMargin As Single
sMargin = Val(TextBox1.Text)
RichTextBox1.SelIndent = sMargin * 567
```

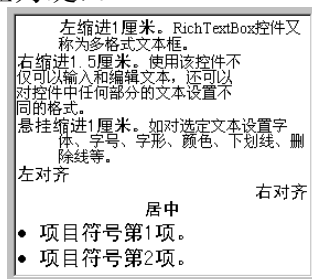
用同样的方法可设置右缩进和悬挂缩进。

2. 段落对齐方式

RichTextBox 控件的 SelAlignment 属性用于设置选定段落的对齐方式。将该属性值设为常数 rtfLeft 或 0 为左对齐，rtfRight 或 1 为右对齐，rtfCenter 或 2 为居中。

3. 项目符号

将 RichTextBox 控件的 SelBullet 属性设为 True 即可为选定段落添加项目符号，若同时设置 BulletIndent 属性，则可指定含有项目符号的段落的缩进量（默认单位为缇）。



段落格式设置效果如图 11.2 所示。

提示：可将设置字体和段落格式的功能汇集在“格式”菜单中。

11.1.3 使用剪贴板

RichTextBox 控件的 SelRTF 属性含有当前选定的 RTF 格式文本。利用 SelRTF 属性和剪贴板 (Clipboard) 对象可以实现多格式文本的剪切、复制和粘贴操作。

1. 操作多格式文本

剪贴板对象的 SetText 方法用于向剪贴板传送文本。以下代码可将多格式文本复制到剪贴板：

```
Clipboard.Clear ' 清空剪贴板  
' 向剪贴板传送 RTF 文本  
Clipboard.SetText RichTextBox1.SelRTF, _  
    vbCFRTF
```

剪切操作与复制相似，只需在上述代码之后加入以下代码：

```
' 删除 RichTextBox 控件中的选定内容  
RichTextBox1.SelText = ""
```

剪贴板对象的 GetText 方法用于从剪贴板中粘贴文本。例如：

```
RichTextBox1.SelRTF = Clipboard.GetText(vbCFRTF)
```

将上述代码分别置于对应菜单项的单击事件中即可实现剪切、复制和粘贴功能。

2. 插入图片

利用 LoadPicture 函数、剪贴板对象的 SetData 方法和模拟键盘输入的 SendKeys 语句可以在 RichTextBox 控件中插入图片。

【例 11.3】利用剪贴板在 RichTextBox 控件中插入图片。效果如图 11.3 所示。



在例 11.2 中添加一个命令按钮或菜单项，在它的单击事件过程中输入以下代码：

```
' 利用通用对话框选择图片文件  
CommonDialog1.ShowOpen  
Clipboard.Clear ' 清空剪贴板  
' 将图片文件发送到剪贴板
```

```
Clipboard.SetData _
    LoadPicture(CommonDialog1.FileName)
RichTextBox1.SetFocus
' 模拟组合键 Ctrl+V 从剪贴板粘贴图片
SendKeys "^v", True
Clipboard.Clear
```

11.1.4 查找文本

RichTextBox 控件的 Find 方法用于搜索特定字符串。若找到待查内容则将其反相显示，并返回其位置；若未找到则返回-1。Find 方法的调用格式为：

RichTextBox 控件名称.Find(待查字符串[, 起始位置, 结束位置, 选项])

【例 11.4】在 RichTextBox 控件中查找文本。

在例 11.3 中添加两个菜单项或命令按钮，标题 (Caption) 分别为“查找”和“查找下一个”。将 RichTextBox 控件的 HideSelection 属性设为 False，以便在控件失去焦点时仍可反相显示找到的字符串。

在代码编辑窗口的“通用-声明”部分声明一个窗体级的变量用于存放待查内容：

```
Dim strFind As String
下面是“查找”菜单项单击事件过程的代码：
'mnuFind 为“查找”菜单项的名称
Private Sub mnuFind_Click()
    strFind = InputBox("输入查找内容", "查找")
    If strFind = "" Then Exit Sub
    'Find 方法返回-1 说明未找到
    If RichTextBox1.Find(strFind) = -1 Then
        MsgBox "未找到“" & strFind & "”。", _
            vbInformation, "提示"
        strFind = ""
    End If
End Sub
```

在“查找下一个”菜单项或按钮的单击事件过程中加入以下代码：

```
Dim lngL As Long
' 若为首次查找则调用“查找”过程
If strFind = "" Then
    Call mnuFind_Click
Else
    With RichTextBox1
        lngL = .SelLength
        .SelStart = .SelStart + lngL
    End With
    If .Find(strFind, .Len(.TextRTF)) = -1 Then
        .SelStart = .SelStart - lngL
        .SelLength = lngL
        MsgBox "查找结束。", vbInformation, "提示"
```

```

End If
End With
End If

```

11.1.5 打开与保存文件

RichTextBox 控件的 LoadFile 和 SaveFile 方法分别用于装载和保存文件。文件格式可以是文本文件或 RTF 文件。下面通过实例说明这两个方法的调用。

【例 11.5】 在 RichTextBox 控件中打开和保存文件。

在例 11.4 中添加一个顶层菜单“文件”，在该菜单下添加“打开”和“保存”两个菜单项。“打开”菜单项单击事件过程的代码如下：

```

Private Sub mnuOpen_Click() ' 打开文件
    On Error GoTo Quit
    With CommonDialog1
        .CancelError = True
        .Filter = "文本文件(*.txt)|*.txt " _
            & "|RTF 文件(*.rtf)|*.rtf"
        .ShowOpen
        If UCase$(Right$(.FileName, 3)) = "RTF" Then
            ' 打开 RTF 文件
            RichTextBox1.LoadFile .FileName, rtfRTF
        Else
            ' 打开文本文件
            RichTextBox1.LoadFile .FileName, rtfText
        End If
    End With
Quit:
End Sub

```

“保存”菜单项单击事件过程的代码如下：

```

Private Sub mnuSave_Click()
    On Error GoTo Quit
    With CommonDialog1
        .CancelError = True
        .Filter = "文本文件(*.txt)|*.txt" _
            & "|RTF 文件(*.rtf)|*.rtf"
        .ShowSave
        ' 设置默认扩展名，在用户未输入扩展名时使用
        If .FilterIndex = 1 Then
            .DefaultExt = ".txt"
        Else
            .DefaultExt = ".rtf"
        End If
        If UCase$(Right$(.FileName, 3)) = "RTF" Then
            ' 保存文件为 RTF 格式

```

```

        RichTextBox1.SaveFile .FileName, rtfRTF
    Else
        ' 保存文本文件
        RichTextBox1.SaveFile .FileName, rtfText
    End If
End With
Quit:
End Sub

```

11.2 TreeView 和 ListView 控件

TreeView 和 ListView 均为 Microsoft Windows Common Controls 6.0 中的控件，需要加载后方可使用，加载方法与 RichTextBox 控件相似。

11.2.1 TreeView 控件

1. 理解 Node 对象与 Nodes 集合

在讨论 TreeView 控件的应用之前应当对 Node 对象和 Nodes 集合有所了解。TreeView 控件中的每个列表项都是一个 Node 对象（节点），节点可包含文本和图片。节点之间的关系可以是父子关系或兄弟关系。

如图 11.4 所示，系与其班级之间为父子关系，各系之间为兄弟关系（位于同一层次），一个系中的班级之间也是兄弟关系。

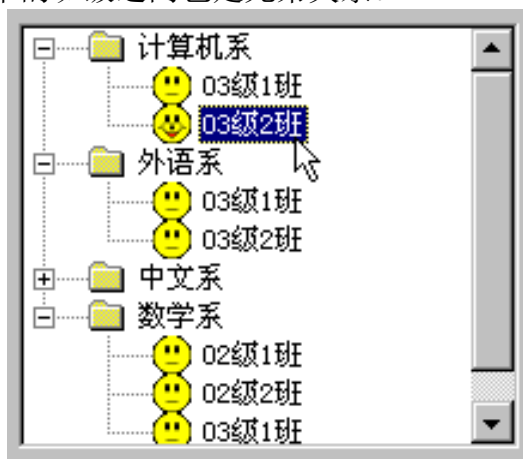


图 11.4 TreeView 控件

系是班级的父节点（Parent），班级是系的子节点（Child）。各系均为顶层节点，顶层节点没有父节点（Nothing）。控件中的所有 Node 对象构成 Nodes 集合，集合中的每一个 Node 对象具有一个惟一的索引（下界为 1），利用索引可以访问集合中的 Node 对象。例如，TreeView1.Nodes(1) 是指集合中的第一个节点。

2. 添加节点

Nodes 集合的 Add 方法用于添加节点。调用格式为：

TreeView 控件名.Nodes.Add([相关节点, 关系, 关键字, 文本, 图片, 选定图片])

Add 方法的 6 个参数均为可选参数。前两个参数共同指定新节点的位置。“相

关节点”为现有某节点的索引或关键字。

“关系”是指新节点与“相关节点”的位置关系，该参数的取值常数为：tvwFirst, tvwLast, tvwNext, tvwPrevious 或 tvwChild，分别对应整数 0~4。其中 tvwChild 为父子关系，即新节点是“相关节点”的子节点。其他常数均为兄弟关系，即新节点与“相关节点”位于同一层次，分别为首位、末位、后邻位和前邻位。如果省略了“相关节点”参数，则在所有顶层节点之后添加一个新节点，并且忽略“关系”参数。Add 方法的其他参数均不难理解

【例 11.6】在 TreeView 控件中建立系和班级的分层列表。

新建工程，在窗体上添加一个 TreeView 控件和一个 ImageList 控件，均采用默认名称。添加两个命令按钮，设 Caption 属性分别为“添加系”和“添加班级”。按第 10 章所述方法在 ImageList 控件中添加 4 张图片。右击 TreeView 控件，在弹出菜单中选择【属性】菜单项，打开如图 11.5 所示的【属性页】对话框，在对话框的【图像列表】中选择 ImageList1，设【线条样式】为 1，单击【确定】按钮关闭对话框。

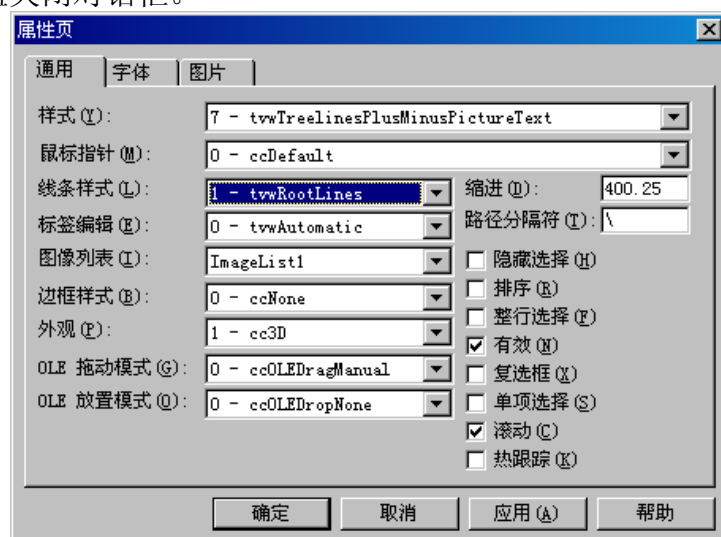


图 11.5 TreeView 控件属性页

在“添加系”按钮的单击事件过程中加入以下代码：

```
Dim mNode As Node ' 声明节点对象变量
' 若省略 Add 方法的第一个参数，则在所有顶层节点之后
' 添加一个新的顶层节点，同时忽略 Add 方法的第二个参数。
' 下面一行语句中的“1,2”为 ImageList 控件中的图片索引。
Set mNode = TreeView1.Nodes.Add(, , , "X 系", 1, 2)
' 添加节点并为变量赋值
mNode.Selected = True ' 选中新节点
' 使新节点标签处于编辑状态以便用户修改
```

TreeView1.StartLabelEdit

在“添加班级”按钮的单击事件过程中加入以下代码：

```
' 若控件中无节点退出此过程
If TreeView1.Nodes.Count = 0 Then Exit Sub
Dim mNode As Node ' 声明节点对象变量
Dim iIndex As Integer
' 若未选择节点将出错，转错误处理语句
```

```

On Error GoTo NodeErr
' 取当前选定节点的索引
iIndex = TreeView1.SelectedItem.Index
' 若选定的节点是“系”节点(无父节点)则添加子节点,
' 否则添加兄弟节点
If TreeView1.Nodes(iIndex).Parent Is Nothing Then
    Set mNode = TreeView1.Nodes.Add(iIndex, _
        tvwChild, , "X 级 X 班", 3, 4)
Else
    Set mNode = TreeView1.Nodes.Add(iIndex, _
        tvwLast, , "X 级 X 班", 3, 4)
End If
mNode.EnsureVisible      ' 使新节点可见
mNode.Selected = True    ' 选中新节点
' 使新节点标签处于编辑状态以使用户修改
TreeView1.StartLabelEdit
Exit Sub
NodeErr: ' 处理错误
MsgBox "请先选择一个系。", vbExclamation, "提示"

```

程序运行效果如图 11.6 和图 11.7 所示。



图 11.6 添加系图 11.7 添加班级

3. 删除和清空节点

Nodes 集合的 Remove 方法和 Clear 方法分别用于删除和清空节点。

【例 11.7】扩展例 11.6 的功能，使之能够删除和清空节点。

在例 11.6 中添加两个按钮，Caption 分别为“删除”和“清空”。在“删除”按钮的单击事件过程中加入以下代码：

```

' 若控件中无节点退出此过程
If TreeView1.Nodes.Count = 0 Then Exit Sub
Dim iIndex As Integer
' 取当前选定节点的索引
iIndex = TreeView1.SelectedItem.Index
' 删除选定节点及其子节点
TreeView1.Nodes.Remove iIndex

```

在“清空”按钮的单击事件过程中加入以下代码：

```

TreeView1.Nodes.Clear ' 清除所有节点

```

11.2.2 ListView 控件

ListView 控件可使用大图标、小图标、列表和报表（详细资料）四种不同视图显示列表项。Windows 资源管理器的右窗格就是 ListView 控件的典型例子。

1. ListView 控件的四种视图

ListView 控件的 View 属性决定它的视图显示方式，有 4 种取值。将该属性值设为常数 `lvwIcon` 或 0 为大图标，`lvwSmallIcon` 或 1 为小图标，`lvwList` 或 2 为列表，`lvwReport` 或 3 为详细资料。四种视图显示模式如图 11.8~图 11.11 所示。



图 11.8 大图标视图



图 11.9 小图标视图

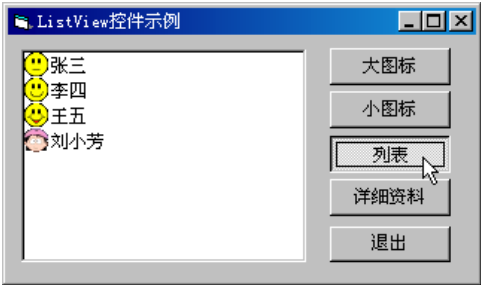


图 11.10 列表视图

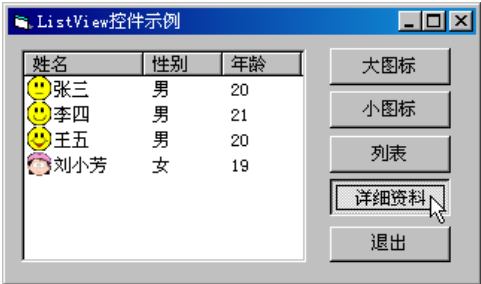


图 11.11 详细资料视图

2. 理解 ListView 控件中的对象与对象集合

(1) ListItem 对象与 ListItems 集合

ListView 控件中的每个列表项都是一个 ListItem 对象，列表项可包含文本和图片。控件中的所有 ListItem 对象构成 ListItems 集合，集合中的每个对象具有惟一索引。在程序代码中调用 ListItems 集合的 Add 方法可以在控件中添加列表项，调用格式为：

ListView 控件名.ListItems.Add([索引, 关键字, 文本, 大图标, 小图标])

(2) ColumnHeader 对象与 ColumnHeaders 集合

在如图 11.11 所示的详细资料视图中，第一行的标题【姓名】、【性别】和【年龄】即为 ColumnHeader 对象（列标头）。控件中的所有 ColumnHeader 对象构成 ColumnHeaders 集合。在列标头下面，左起第一列是在各种视图中均可显示的列表项，列表项右侧的各列均为列表子项（SubItem）。每个列表项可以有多个子项，它们构成子项数组（SubItems），数组类型为字符串型，下界为 1，上界为列标头总数-1。

调用 ColumnHeaders 集合的 Add 方法可以添加列标头，调用格式为：

ListView 控件名.ColumnHeaders.Add([索引, 关键字, 文本, 宽度, 对齐方式, 图标])

添加列标头后将自动确定列表子项数组的上界，此时可以为子项数组元素赋值。

3. 在 ListView 控件中使用图片

ListView 控件中所用的图片由 ImageList 控件提供。一个 ListView 控件可以使用三个 ImageList 控件，分别提供大图标、小图标（供小图标、列表和详细资料视图使用）和列标头图标。在设计时可以通过 ListView 控件的属性页指定 ImageList 控件。程序运行时可以通过代码指定要使用的 ImageList 控件，例如：

```
Set ListView1.Icons = Imagelist1 ' 大图标
Set ListView1.SmallIcons = Imagelist2 ' 小图标
' 列标头图标
Set ListView1.ColumnHeaderIcons = Imagelist3
```

【例 11.8】设计如图 11.8~图 11.11 所示的 ListView 控件的不同视图。

新建工程，在窗体上添加一个 ListView 控件和两个 ImageList 控件（本例中未使用列标头图标），均采用默认名称。创建一个含有四个元素的单选按钮数组，名称均为 optView，索引为 0~3，设 Style 属性均为 1，Caption 属性分别为“大图标”、“小图标”、“列表”和“详细资料”。ImageList 控件和 ListView 控件的属性分别通过图 11.12 和图 11.13 所示的属性页设置。



图 11.12 ImageList 属性页

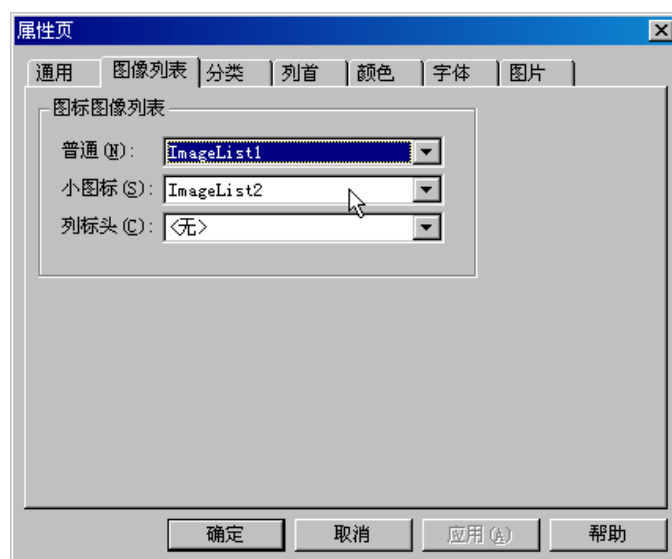


图 11.13 ListView 属性页

右击 ImageList1，在弹出菜单中选择【属性】菜单项，打开如图 11.12 所示的【属性页】对话框，在【通用】选项卡中选择【32 x 32】单选钮（此步骤设置图像大小），然后切换到【图像】选项卡添加 4 个图片。用同样的方法将 ImageList2 的图像大小设为【16 x 16】并添加图片。右击 ListView 控件，在弹出菜单中选择【属性】菜单项，打开如图 11.13 所示的【属性页】对话框，切换到【图像列表】选项卡，在【普通】组合框中选择 ImageList1，在【小图标】组合框中选择 ImageList2。

在窗体的 Load 事件中对 ListView 控件进行初始化：

```
Private Sub Form_Load()  
    ' 添加列标头。数字为宽度(缇)  
    ListView1.ColumnHeaders.Add , , "姓名", 1200  
    ListView1.ColumnHeaders.Add , , "性别", 800  
    ListView1.ColumnHeaders.Add , , "年龄", 800  
    Dim itmX As ListItem ' 声明列表项对象变量  
    Set itmX = ListView1.ListItems.Add( , _  
        "张三", 1, 1) ' 添加列表项  
    ' 设置子项，供“详细资料”视图使用  
    itmX.SubItems(1) = "男"  
    itmX.SubItems(2) = 20
```

```

' 添加其他列表项
.....
End Sub
利用单选按钮的单击事件切换视图，代码如下：
' 用单选按钮控件数组切换视图
Private Sub optView_Click(Index As Integer)
    ' 4 个单选按钮的索引号为 0~3，恰好与 ListView 控件
    ' View 属性的 4 个常数值相对应，
    ' 因此用单选按钮的索引号为 ListView 控件的 View 属性赋值
    ' 可简化代码。
    ListView1.View = Index
End Sub

```

11.3 其他扩展控件

11.3.1 SSTab 控件

SSTab 控件提供一组选项卡，每个选项卡都可作为其它控件的容器。该控件在 Microsoft Tabbed Dialog 6.0 部件中，加载后才能使用。

【例 11.9】制作如 189 页图 11.14 和图 11.15 所示含有两个选项卡的用户界面。

图 11.14 基本情况选项卡

图 11.15 附加信息选项卡

在窗体上添加一个 SSTab 控件，右击该控件，在弹出菜单中选择【属性】菜单项，打开如图 11.16 所示的【属性页】对话框。在对话框中将【选项卡数】设为 2，将【样式】设为 1。在【选项卡标题】文本框中输入第一个选项卡的标题“基本情况”。单击“>”按钮，输入第二个选项卡的标题“附加信息”。单击【确定】按钮关闭对话框。

根据图 11.14 和图 11.15 为两个选项卡分别添加相关控件并设置属性。其中，【基本情况】选项卡中用于输入姓名的文本框名称为 txtName，【附加信息】选项卡中用于显示姓名的标签名称为 lblName。

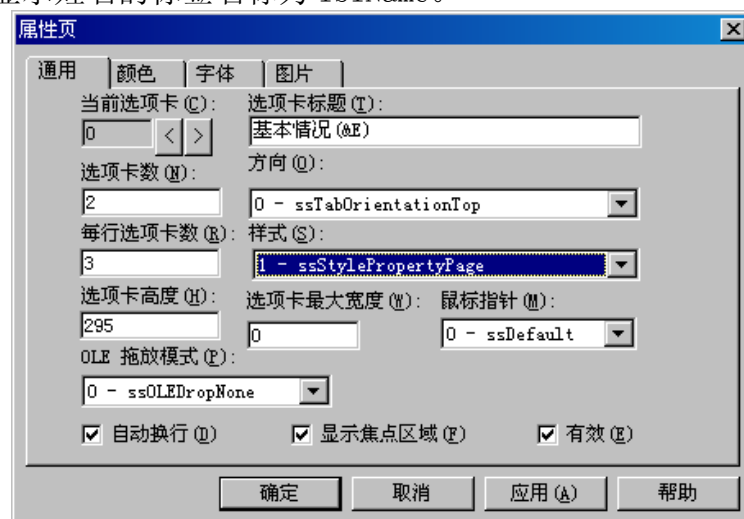


图 11.16 SSTab 控件属性页

在 SSTab 控件的单击事件中加入以下代码：

```
Private Sub SSTab1_Click(PreviousTab As Integer)
    ' Tab 属性返回当前活动选项卡的索引(下界为 0)
    If SSTab1.Tab = 1 Then ' 若单击“附加信息”选项卡
        lblName.Caption = "姓名：" & txtName.Text
    End If
End Sub
```

11.3.2 ProgressBar 控件

在应用程序中，当执行一个耗时较长的操作时，可用 ProgressBar 控件（进度条）显示事务的进程。ProgressBar 属于 Microsoft Windows Common Controls 6.0 中的控件，需要加载后使用。

ProgressBar 控件有三个最重要的属性 Min、Max 和 Value。Min 和 Max 用于设置进度条的起点和终点。Value 属性用于在运行时设置或返回进度条的填充量。在显示某操作的进展情况时，可以将 Value 值设为从 Min 值开始递增，直到由 Max 属性定义的最大值为止。下面通过实例说明该控件的使用。

【例 11.10】用进度条和定时器控件模拟数据处理的进度，如图 11.17 所示。

在窗体上添加一个框架 Frame1，设其 Caption 属性为空，Visible 属性为 False。在框架中添加两个标签，均采用默认名称。设 Label1 的 Caption 属性为“正在处理数据，请稍候...”。



图 11.17 进度条

输入以下代码：

```
Dim intValue As Integer ' 窗体级变量用于存放进度值
```

```
Private Sub cmdStart_Click() ' “开始” 按钮
    intValue = 0
    ProgressBar1.Value = 0 ' 进度条初始化
    Frame1.Visible = True ' 显示框架及其中的进度条等控件
    Timer1.Enabled = True ' 启动定时器
    cmdStart.Enabled = False ' 使“开始”按钮无效
End Sub

Private Sub Timer1_Timer() ' 定时器事件
    intValue = intValue + 1 ' 累加进度值
    If intValue > 100 Then ' 若超过最大值
        Timer1.Enabled = False ' 关闭定时器
        MsgBox "数据处理结束。", vbInformation, "提示"
        Frame1.Visible = False ' 隐藏框架及其中的控件
        cmdStart.Enabled = True ' 设“开始”按钮有效
    Else
        ' 设置 Value 属性值，显示进度
        ProgressBar1.Value = intValue
        ' 显示进度百分比
        Label2.Caption = intValue & "%"
    End If
End Sub
```

11.3.3 DateTimePicker 控件

DateTimePicker 控件 (DTPicker) 可以按指定格式显示日期或时间，并且作为修改日期和时间信息的界面。该控件属于 Microsoft Windows Common Controls-2 6.0 中的控件，加载后方可使用。DateTimePicker 控件有两种不同的显示模式：

- ① 下拉日历模式。单击控件右部的下拉箭头可显示日历，用于选择日期。
- ② 时间显示模式。用于显示或设置时间。可在控件中选择一个域（时、分、秒）后，用控件右部的上下箭头设置其值，亦可通过键盘输入数字或按箭头键设置其值。

通过 DateTimePicker 控件的 Format（格式）属性可以设置日期或时间的显

示格式。

Format 属性有 4 种取值：设为常数 dtpLongDate 或 0 为长日期格式，dtpShortDate 或 1 为短日期格式，dtpTime 或 2 为时间格式，dtpCustom 或 3 为自定义格式。当 Format 属性值为 0 或 1 时，控件以下拉日历模式显示日期；Format 属性值为 2 时，以时间模式显示时间。当 Format 属性值为 3 时，控件的显示模式取决于 CustomFormat（自定义格式）属性和 UpDown（上下箭头）属性。若 CustomFormat 属性为日期格式字符串，且 UpDown 属性为 False，则为下拉日历模式。若 CustomFormat 属性为时间格式字符串，且 UpDown 属性为 True，则为时间显示模式。

【例 11.11】使用 DateTimePicker 控件选择日期并设置时间，当到达预定的日期和时间时提示用户。

在窗体上添加两个 DateTimePicker 控件 DTPicker1 和 DTPicker2，分别用于设置日期和时间。右击 DTPicker1，在弹出菜单中选择【属性】菜单项，打开如图 11.18 所示的【属性页】，在【通用】选项卡中将【格式】设为 3-dtpCustom，将【自定义格式】设为“yyy-M-d”（yyy 为完整年份）。用同样的方法将 DTPicker2 的【格式】设为 2-dtpTime。



图 11.18 DateTimePicker 控件属性页

在两个 DateTimePicker 控件的上方各添加一个标签，用作简单说明。添加一个文本框和两个命令按钮，按钮的 Caption 属性分别为“确定”和“退出”。添加一个 Timer 控件，设其 Enabled 属性为 False，Interval 属性为 500。

在“确定”按钮的单击事件过程中加入以下代码：

```
Text1.Text = "提示日期：" _  
    & Format(DTPicker1.Value, _  
    "yyyy 年 m 月 d 日") _  
    & vbCrLf & "提示时间：" _  
    & TimeValue(DTPicker2.Value)
```

```
Timer1.Enabled = True
```

在 Timer1 控件的 Timer 事件中加入以下代码：

```
If DateValue(DTPicker1.Value) = Date _  
    And TimeValue(DTPicker2.Value) _  
    = Time Then  
    MsgBox "时间到。"  
    Timer1.Enabled = False
```

```
End If
```

说明：DateTimePicker 控件的 Value 属性用于返回或设置日期和时间。

程序运行效果如图 11.19 和图 11.20 所示。在图 11.19 中，单击控件的下拉箭头显示日历，单击年份和月份可修改年月，单击日历中的某个日期即完成设定。

在图 11.20 中，单击时间模式控件中的上下箭头可设置时间。



图 11.19 下拉日历

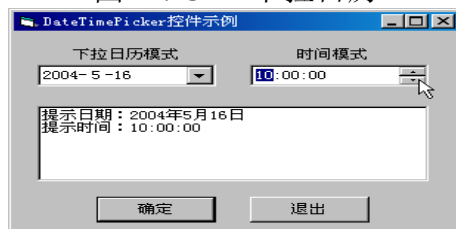


图 11.20 设置时间

【练习】

1. 将例 11.1～例 11.5 中所述 RichTextBox 控件的功能汇集在一个程序的菜单项中，利用所学知识，使该程序具有尽可能全面的多格式文本编辑功能。
2. 在 TreeView 控件中显示本书第 1 章～第 3 章的目录。
3. ListView 控件有哪几种显示方式？改变该控件的显示方式应设置何属性？请列出该属性的各种取值常数。
4. 当用户在 SSTab 控件中切换选项卡时发生何事件？如何判断哪一个选项卡是当前活动选项卡？
5. 如何使用 ProgressBar 控件中的填充块逐步递增？
6. 要使 DateTimePicker 控件以“2004 年 1 月 1 日”这种格式显示日期，应当设置哪些属性？

第 13 章 文件管理

本章要点：

- 文件系统控件
- 访问文件
- 文件系统对象

13.1 文件系统控件

13.1.1 驱动器列表框

驱动器列表框 (DriveListBox) 控件的作用是为用户提供一个选择有效磁盘驱动器的功能。该控件可以显示出用户系统中所有有效磁盘驱动器的列表。在图 13.1 的示意图中, 左边是驱动器列表框的原始状态, 右边是展开后的状态。

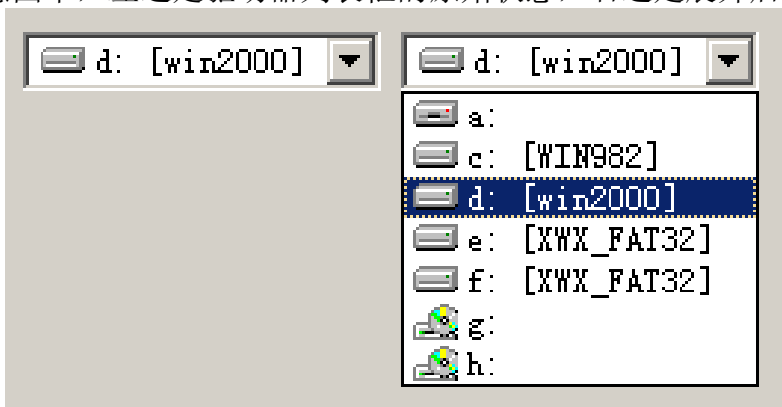


图 13.1 驱动器列表框示意图

DriveListBox 控件最重要的属性是 Drive 属性。程序运行时, 可以通过该属性获得当前所选择的驱动器, 也可以改变 Drive 属性的值, 指定某个驱动器。

例如:

将驱动器列表框 Drive1 中所选择的驱动器显示在文本框 Text1 中:

```
Text1.Text = Drive1.Drive
```

将驱动器列表框 Drive1 中的驱动器设置为 “e:\”:

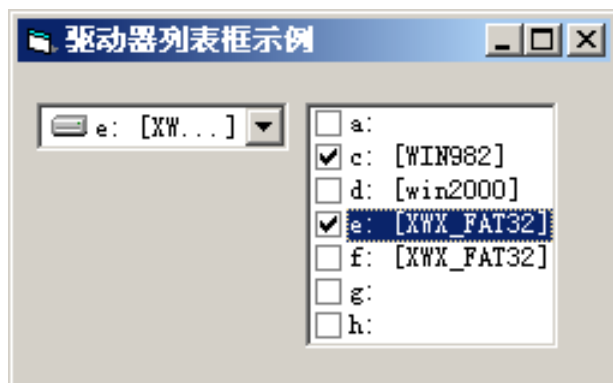
```
Drive1.Drive = "e:\"
```

访问驱动器列表框中的列表项目时, 其方式与普通列表框控件类似, 即可以使用 List 属性数组访问; ListCount 表示列表项目的个数; ListIndex 表示当前选中的项目在列表中的索引位置。

驱动器列表框的主要事件是 Change 事件。在程序的运行阶段, 如果选择了一个驱动器列表项目, 或者通过代码改变了 Drive 属性的值, 均将引发控件的 Change 事件。

【例 13.1】 将系统的驱动器信息收集到一个列表框中, 若在驱动器列表框中选择一个驱动器, 则列表框中的相应驱动器也要被选中。

在窗体上添加一个驱动器列表框 Drive1。添加一个普通列表框 List1, 设其 Style 属性为 1 (带有复选框)。设窗体的 Caption 为 “驱动器列表框示例”。



程序的运行结果见图 13.2。

代码如下：

```
Private Sub Form_Load()
    Dim i As Integer
    For i = 0 To Drive1.ListCount - 1
        List1.AddItem Drive1.List(i)
    Next i
End Sub

Private Sub Drive1_Change()
    List1.Selected(Drive1.ListIndex) = True
End Sub
```

13.1.2 目录列表框

目录列表框 (DirListBox) 控件用于显示当前或指定的驱动器的全部目录结构，其显示方式是分层的文件夹（目录）列表，类似于 Windows 的“资源管理器”。目录列表框默认显示的是与当前目录相关的目录结构，通过双击列表中的一个目录项，就可以打开该目录项的第一级子目录，从而浏览到全部的目录结构，如图 13.3 所示。

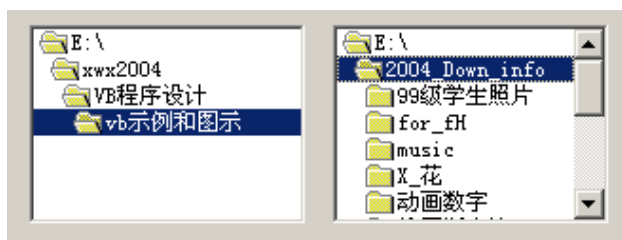


图 13.3 目录列表框

DirListBox 控件的 Path 属性用来返回或设置当前的目录路径，其值是一个指示路径的字符串。例如，输出目录列表框控件 Dir1 中的当前目录路径：

```
Print Dir1.Path
```

又如，将目录列表框控件 Dir1 中的当前目录路径设置为“e:\xwx2004”：

```
Dir1.Path = "e:\xwx2004"
```

目录列表框的 List 属性数组中包含了所有的目录列表项目，访问该数组的方式与普通列表框控件类似，也是通过索引值 ListIndex 进行。

目录列表框的索引值有以下特殊规定：

(1) Path 属性所指定的目录的索引值总是为-1，因此，通过 Dir1.Path 或 Dir1.List(-1) 都可以获得当前目录。

(2) 紧邻当前目录之上的目录，其索引是-2，再上一个为-3，依次类推。

(3) 紧邻当前目录下的第一个子目录，其索引是 0；若有多个一级子目录，则每个子目录的索引分别是 0、1、2、…，直到 ListCount-1。因此 ListCount 属性表示当前目录下的一级子目录个数，而不是目录列表框中列出的所有项目。

例如，一个目录列表框 Dir1 的索引值情况如图 13.4 所示，其中当前目录是“VB”，其下的一级子目录有 3 个，故 Dir1.ListCount 的值为 3。

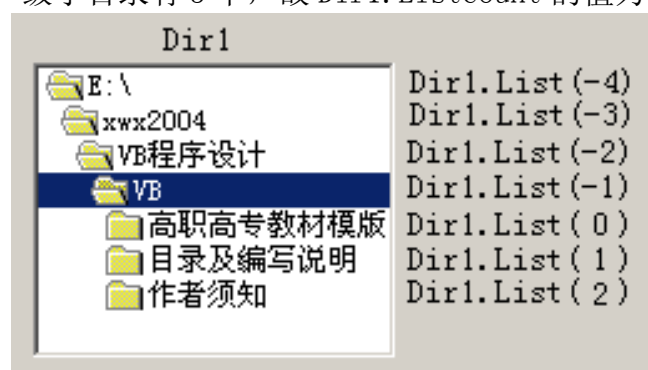


图 13.4 目录列表框的索引值

目录列表框的事件主要是 Change 事件和 Click 事件。单击目录列表框中的某个项目时，系统自动修改 ListIndex 属性值，同时触发 Click 事件。双击目录列表框中的某个项目时，自动设置 Path 属性为当前选择的目录，并修改 ListIndex 属性值为-1，同时触发 Change 事件。如果在代码中直接修改了 Path 属性的值，也会触发 Change 事件。

目录列表框的 Change 事件和 Click 事件之间没有冲突，但通常情况下，我们只对 Change 事件编程。

13.1.3 文件列表框

文件列表框 (FileListBox) 控件根据指定的目录，在程序运行时自动显示该目录下所有文件的文件名。文件列表框的外观与普通列表框相似，如图 13.5 所示。

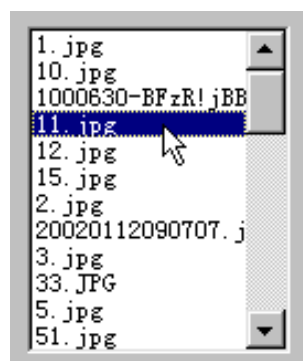


图 13.5 文件列表框

文件列表框的 Path 属性返回或设置当前指定的目录，一旦 Path 属性的值发生改变，文件列表框控件会自动刷新显示列表。

FileName 属性返回在文件列表框中选择的文件名。

Pattern 属性指定文件列表框控件显示的文件类型，默认值为“*.*”，即显示所有类型的文件。如果要显示特定的文件类型，则可设置该属性。例如要显示扩展名为.txt 的文件，可以将 Pattern 属性设置为“*.txt”。若需要同时显示多种类型的文件，则 Pattern 属性设置的文件类型要用分号分隔。例如设置 Pattern 属性为“*.txt ; *.doc”，表示显示扩展名为.txt 和.doc 的文件。

文件列表框的 List 属性返回或设置控件列表中的文件名，这是一个数组，0 是列表中第一个文件名的索引号；ListIndex 属性代表当前选中文件名的索引号。假如 File1 是一个 FileListBox 控件，则 File1.FileName 与 File1.List(File1.ListIndex) 等价。

ListCount 属性返回控件列表内文件名的个数。

文件列表框的事件主要是 Click 事件和 DblClick 事件。这两个事件以及 List 属性、ListIndex 属性和 ListCount 属性的用法及作用与普通列表框相同。

13.1.4 文件系统控件的同步操作

将上述三个文件系统控件组合起来，就可以查看整个驱动器文件系统，但要实现它们之间的同步显示，就需要编写特定的程序。下面我们通过一个例题说明如何实现三者之间的同步操作。

【例 13.2】使用文件系统控件获得文件名及其路径。

在窗体上添加一个 DriveListBox 控件 Drive1、DirListBox 控件 Dir1 和 FileListBox 控件 File1；再添加两个标签 Label1、Label2，设其 BorderStyle 均为 1（具有边框），Caption 均为空，用来显示路径和文件名。

设计的关键是，必须保证三个文件系统控件的相互协调联动。在程序的运行阶段，改变驱动器时，就会触发 Drive1 控件的 Change 事件，通过 Dir1.Path = Drive1.Drive 来改变 Dir1 的目录。当 Dir1 改变目录时，就会激活 Dir1 控件的 Change 事件，然后通过 File1.Path = Dir1.Path 来使 File1 控件显示改变目录后的文件列表。

另一个要注意的问题是 File1 的 Path 属性。按我们通常的想法，File1.Path 与 File1.FileName 进行字符串连接后应该得到完整的文件路径和文件名，但实际上并非如此。

如果选中的一个文件 test.txt 位于 E: 根目录下，File1 的 Path 属性所取得的目录路径是“E:\”，File1.Path 与 File1.FileName 进行字符串连接后的结果是“E:\test.txt”；然而，如果 test.txt 文件位于 E: 根目录下的 user 子目录下，则 Path 属性所取得的目录路径是“E:\user”，File1.Path 与 File1.FileName 连接后的结果是“E:\usertest.txt”，显然出错。人为地在后一种情况下的路径后面加上“\”符号，既能够统一形式，也符合我们通常的习惯，避免出错。这便是本例中定义函数 MakePath 的原因。

运行结果如图 13.6 所示

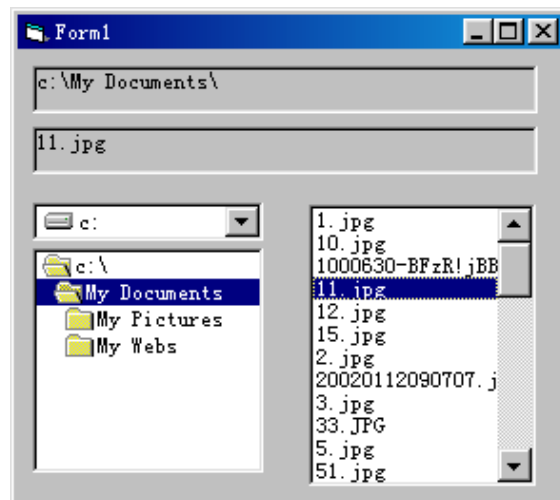


图 13.6 例 13.2 运行结果

代码如下：

```

' 自定义函数：生成以 “\” 结尾的路径字符串
Function MakePath(S As String) As String
    If Right(S, 1) = "\" Then
        MakePath = S
    Else
        MakePath = S & "\"
    End If
End Function

Private Sub Dir1_Change() ' 目录列表框
    File1.Path = Dir1.Path ' 使文件列表框与目录列表框同步
End Sub

Private Sub Drive1_Change() ' 驱动器列表框
    ' 设置错误捕获, 避免软驱或光驱中无盘出错导致程序中断
    On Error GoTo ErrDrive
    ' 使目录列表框与驱动器列表框同步
    Dir1.Path = Drive1.Drive
    Exit Sub
ErrDrive:
    MsgBox Drive1.Drive & " 驱动器未准备好。", _
        vbCritical, "提示"
End Sub

Private Sub File1_Click() ' 文件列表框
    Label1.Caption = MakePath(File1.Path) ' 显示路径
    Label2.Caption = File1.FileName ' 显示文件名
End Sub

```

常用文件处理语句和函数：

(1) ChDrive 语句

改变当前驱动器。例如，将当前驱动器改为应用程序（工程）所在驱动器：

```
ChDrive App.Path
```

'App 是 VB 默认的全局对象, 指当前正在运行的应用程序或工程

(2) ChDir 语句

改变驱动器的当前目录。例如:

ChDir "E:\xwx2004" ' 改变 E 盘驱动器的当前目录。

ChDir "\book" ' 改变当前驱动器的当前目录。

ChDir App.Path ' 将当前目录设为应用程序(工程)所在目录

(3) Mkdir 语句

在指定的驱动器上创建一个新目录, 如果不指定驱动器, 则在当前驱动器上创建新目录。例如:

在当前驱动器上创建新目录:

Mkdir "VBtest"

在 D 盘驱动器根目录下创建新目录:

Mkdir "D:\test"

(4) Rmdir 语句

Rmdir 语句的功能是删除一个存在的而且为空的目录, 语法结构如下:

Rmdir <目录或文件夹名>

其中, 目录或文件夹名参数是一个字符串表达式, 用来指定要删除的目录或文件夹。如果在参数中没有指定驱动器, 则 Rmdir 会在当前驱动器上执行删除操作。

(5) CurDir 函数

CurDir 函数返回当前目录。例如:

返回当前驱动器的当前目录: S1=CurDir()。

返回 D 盘驱动器上的当前目录:

S2=CurDir("D:").

(6) Dir 函数

返回指定路径下的第一个子目录或文件, 若再次调用, 则返回该路径下的下一个子目录或文件。例如:

返回 D 盘驱动器上的第一个子目录:

MyPath=Dir("D:\", vbDirectory)

其中 vbDirectory 是系统定义的 VB 常数, 指定无属性文件及其路径和文件夹。

(7) Kill 语句

Kill 语句的功能是从磁盘中删除一个或多个文件, 它的语法结构如下:

Kill <文件名>

其中, 文件名参数是字符串表达式, 可包含文件所在的目录以及驱动器。被删除的文件必须存在, 否则会出现错误提示。Kill 支持通配符*和?, 可实现一次删除多个文件。要注意的是 Kill 语句正常执行时没有提示, 为了避免误操作, 用户应该在程序中执行 Kill 语句前加入自己的提示。例如:

删除 D 盘驱动器上根目录下的一个文件:

Kill "D:\a.txt"

删除当前驱动器上当前目录下的多个.tmp 文件:

Kill "*.tmp"

(8) FileCopy 语句

FileCopy 语句的功能是复制一个文件，语法结构如下：

FileCopy <源文件名> , <目标文件名>

源文件名和目标文件名中都可以包含文件所在的目录以及驱动器。源文件必须存在，且目标文件名的路径也必须存在。例如：

将 D 盘根目录下的文件 a.txt 同名复制到 C 盘根目录下：

FileCopy "D:\a.txt", "C:\a.txt"

在当前目录下，将文件 form1.frm 复制成文件 frmMain.frm：

FileCopy "form1.frm", "frmMain.frm"

(9) Name 语句

Name 语句的功能是重新命名一个文件，它的语法结构如下：

Name <原文件名> As <新文件名>

原文件必须已经存在，新文件名不能与当前目录下的文件同名。当原文件名和新文件名相同且不在同一目录或分区时，将把原文件移动到新的目录下，原文件将被删除。

Name 语句还能够对目录重命名。

例如，将 D 盘根目录下的文件 a.txt 改名为 a.dat：

Name "D:\a.txt" As "a.dat"

【例 13.3】在例 13.2 的基础上，应用 Kill 语句和 Rmdir 语句，实现删除一个非空目录的功能。

首先为例 13.2 的窗体添加两个命令按钮 Command1 和 Command2，其 Caption 属性分别设为“删除文件”和“删除目录”，然后为 Command1 和 Command2 添加代码：

```
Private Sub Command1_Click()  
    Str = MakePath(File1.Path)  
    Kill Str & File1.FileName '删除一个文件  
End Sub  
Private Sub Command2_Click()  
    '删除目录下的所有文件  
    Kill MakePath(File1.Path) & "*. *"   
    Rmdir Dir1.Path '删除空的目录  
End Sub
```

13.2 用传统语句和函数访问文件

13.2.1 访问顺序文件

1. 顺序文件的打开和关闭

(1) 打开顺序文件使用 Open 语句，其语法结构如下：

Open pathname For mode As # Filenumber

在 Open 语句的参数中，pathname 指定文件路径和文件名；mode 指定文件打开方式，对顺序文件有 Append 方式（向文件尾追加）、Input（从文件读取）、Output（向文件输出）方式；Filenumber 是指定的一个有效的文件号，范围在 1 到 511 之间。

从文件中读取数据，应该以 Input 方式打开文件，并且文件必须已经存在；基于将文件作为输入设备的特点，可以用不同的文件号打开同一个文件。

把数据写入文件，应该以 Output 或 Append 方式打开文件；若文件不存在，Open 语句会先创建该文件，然后再打开该文件；若文件存在，Output 方式先清除文件内容，使之成为一个空文件，Append 方式则保留原内容。

下面是几种使用 Open 语句的示例：

以顺序读模式打开 C:\XU\testa.txt 文件：

```
Open "C:\XU\testa.txt" For Input As #1
```

以顺序写模式打开 C:\XU\testa.txt 文件：

```
Open "C:\XU\testa.txt" For Output As #1
```

(2) 关闭文件使用 Close 语句，语法格式如下：

```
Close # Filenumber
```

Close 语句按文件号关闭文件，实际上，所有用 Open 语句打开的文件均使用该语句关闭。

例如：语句 Close #1 表示关闭文件号为 1 的文件；语句 Close #2, #7, #8 表示关闭文件号为 2, 7, 8 的文件；不带参数的语句 Close 表示关闭所有已打开的文件。

2. 顺序文件的读操作

从顺序文件中读取数据，使用 Input 语句和 Line Input 语句。语法结构如下：

```
Input # Filenumber, varlist
```

```
Line Input # Filenumber, var
```

其中，Filenumber 是指定的文件号；varlist 可以是一个变量，也可以是用逗号分隔的变量列表；var 是一个字符串变量。

Input 语句和 Line Input 语句都可从指定的文件中读取数据值，分别赋值给指定的变量。不同的是 Input 语句一次读取一个数据项，Line Input 语句一次读取一个数据行（不包括回车换行符）。

例如，如果顺序文件 c:\score.dat 的内容如下：

```
"zhang", "wang", "li"
```

```
78, 99, 67
```

则执行程序段：

```
Open "c:\score.dat" For Input As #1
```

```
Input #1, x, y, z
```

```
Input #1, a, b, c
```

```
Close #1
```

```
Print x, y, z
```

```
Print a, b, c, a + b + c
```

在窗体上显示的内容为：

```
zhang    wang    li
```

```
78      99      67      244
```

也可用 Input() 函数读取数据。该函数可以从文件中读取任意数量的字符，一次性地存入字符串变量中。它要求字符串变量具有足够大的空间。语法格式如下：

```
字符串变量 = Input(读出的字符数, Filenumber)
```

3. 顺序文件的写操作

向顺序文件中写入数据，使用 Write 语句和 Print # 语句。语法结构如下：

```
Write #filenumber, outputlist  
Print #filenumber, outputlist
```

其中 Filenumber 是指定的文件号；outputlist 是要写入文件的表达式，并可以含有 Spc(n)、Tab(n)等函数，如果是多个表达式，则用逗号将这些表达式分隔开，若该参数为空，则将在文件中写入一空行。

用 Write 语句向顺序文件写入数据时，能自动在各数据项之间插入逗号，并给各字符串加上双引号。例如：

```
Open "d:\shu.dat" For Output As #6  
Write # 6, "zhang"; "wang"; "li"  
Write # 6, 78; 99; 67  
Close #6
```

执行上面的程序段后，写入到文件 d:\shu.dat 中的数据如下：

```
"zhang", "wang", "li"  
78, 99, 67
```

Print # 语句可以把格式化显示的数据写入到顺序文件中。例如：

```
Open "d:\shu.dat" For Output As #2  
Print # 2, "zhang"; "wang"; "li"  
Print # 2, 78; 99; 67  
Close #2
```

执行以上程序段后，写入到文件中的数据如下：

```
zhangwangli  
78 99 67
```

4. 文件的结束

读文件时，使用 EOF 函数可以判断是否已经到达了文件的结尾。对于以 Input 方式打开的顺序文件、以 Random 方式打开的随机文件，如果该函数返回值是 True，则表明已经到达文件的结尾，此时再执行读操作，将引发错误。语法：

```
X = EOF(filenumber)
```

【例 13.4】编写学生成绩录入程序。可以在输入一个学生的姓名和成绩后，将其存入 E:\Test.txt 中，也可读出文件中保存的数据并显示出来。

用两个文本框 Text1 和 Text2 进行姓名和成绩的输入；用命令按钮 cmdSave 执行“写文件”操作；用命令按钮 cmdRead 执行“读文件”操作；用列表框 List1 显示读出的数据。运行情况如图 13.7。

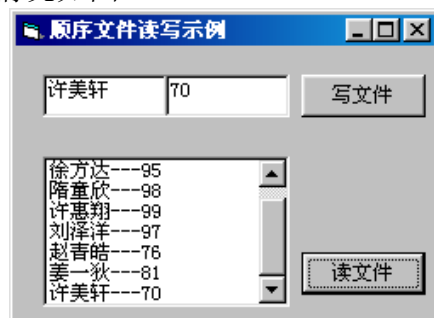


图 13.7 例 13.4 运行结果

代码如下：

```
Private Sub cmdSave_Click()
```

```

' 无姓名或无成绩时的输入无效
If Text1.Text = "" Or Text2.Text = "" Then
    Exit Sub
End If
' 按追加方式打开文件，定为 1 号
Open "e:\test.txt" For Append As #1
' 将文本框内容成对地写到 1 号文件中
Write #1, Text1.Text, Text2.Text
Close #1
End Sub
Private Sub cmdRead_Click()
    ' 定义存放姓名和成绩的变量
    Dim name As String, score As String
    ' 按读方式打开文件，定为 1 号
    Open "e:\test.txt" For Input As #1
    List1.Clear          ' 清空列表框
    Do While Not EOF(1)   ' 若 1 号文件还有数据则循环
        Input #1, name, score ' 成对地读出数据（字符串）
        ' 将读出数据添加到列表框中
        List1.AddItem name & "---" & score
    Loop
    Close #1
End Sub

```

有两点需要注意：一是录入的数据被成对地写到文件中，所以也相应的成对地读出，使得数据带有明显的含义。二是数据一经写入到顺序文件，便被转换为 ASCII 码字符，所以，从顺序文件中读出的数据也都是字符或字符串；对于读出的成绩字符串，应该使用函数 Val() 将其转换为数值，以便进行数值运算。本例只是将其加入的 List1 的列表中，故没有进行数据类型的转换

13.2.2 访问随机文件

1. 记录变量的声明

在存取随机文件的记录数据之前，首先应声明用来处理该文件数据所需的变量。通常是声明成用户自定义类型（称为记录类型）的变量，它对应着该文件中的记录。

注意：可以在标准模块中用 Type 语句声明公共的记录类型，如果在窗体的通用段定义记录类型，一定要使用带有 Private 关键字的 Type 语句，即声明成私有的记录类型。

例如，利用随机文件存放若干用户的编号、姓名、密码，则可以在窗体的通用段定义如下的记录类型 Record，并声明 Record 类型的变量 MyRecord：

```

Private Type Record    ' 定义记录类型
    id As Integer       ' 编号
    name As String * 10 ' 姓名
    pass As String * 10 ' 密码

```

```
End Type  
Dim MyRecord As Record    ' 声明变量
```

2. 打开随机文件

打开随机文件也使用 Open 语句，语法格式如下：

```
Open pathname For Random As #FileNumber Len = reclength
```

其中的关键字 For Random 表示随机访问方式，因为它是默认的访问类型，所以 For Random 关键字可以省略。表达式 Len=reclength 指定了每个记录的长度（字节数），可以用 Len 函数获得。例如：

```
Open "e:\tool\pwfile.bin" For Random As #1 _  
    Len = Len(MyRecord)
```

3. 读写随机文件

与顺序文件不同的是，随机文件打开后，可同时进行写入与读出操作。

使用 Get 语句把记录从文件复制到变量。使用 Put 语句把记录添加或者替换到随机文件。语法格式：

```
Put # filenumber, Position, 记录变量  
Get # filenumber, Position, 记录变量  
Position 的值表示文件中记录的记录号，取值 1, 2, ……。
```

例如：

读入 1 号文件中的第 4 个记录数据，存放到 MyRecord 变量中：

```
Get #1, 4, MyRecord
```

将 MyRecord 变量中的数据写入到 1 号文件的第 1 个记录上：

```
Put #1, 1, MyRecord
```

添加新记录时，写入记录变量的位置 Position 应设置为文件中的记录数加 1。例如，要在一个包含五个记录的文件中的添加一个记录，则把 Position 设置为 6。

修改记录时，先按记录号把记录读出到记录变量中，然后改变记录变量成员的值，最后，把变量以相同的记录号写回该文件。

删除一个记录，可按照以下步骤执行：创建一个新文件；把所有需保留的记录从原文件中读出，并写入到新文件；关闭原文件并用 Kill 语句删除它；使用 Name 语句把新文件以原文件的名字重新命名。

【例 13.5】使用前面已经定义的记录类型 Record，编写用户密码管理程序，用户信息保存到 e:\user.rec 文件中。

用户信息包括用户的编号、姓名、密码，用三个文本框 Text1、Text2、Text3 完成用户信息的输入；用列表框 List1 进行文件记录的浏览。四个命令按钮分别是：“添加记录”按钮 cmdAdd、“浏览记录”按钮 cmdBrows、“修改记录”按钮 cmdChange、“保存修改”按钮 cmdSave。

程序运行后，当单击“添加记录”按钮时，将三个文本框中的用户信息追加到文件中保存；当单击“浏览记录”按钮时，从随机文件 e:\user.rec 中读出所有记录，并依次添加到列表框 List1 中；当单击“修改记录”按钮时，首先要求输入记录号，然后将对应的记录读到文本框中，使用户可以通过文本框修改信息；修改完成后，单击“保存修改”按钮，将信息保存到文件的原来位置上，从而更新了随机文件 e:\user.rec 中的一条指定记录。

程序运行结果见图 13.8，代码详见教材。

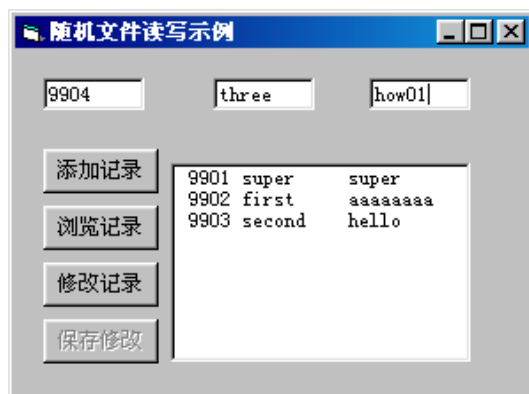


图 13.8 例 13.5 运行结果

13.2.3 访问二进制文件

1. 打开二进制文件

打开二进制文件也是使用 Open 语句。其格式为：

```
Open pathname For Binary As # Filenumber
```

其中，参数 pathname 和 Filenumber 分别表示文件名或文件号，关键字 Binary 表示打开的是二进制文件。例如：

```
Open "student.dat" For Binary As #1
```

该语句用二进制方式打开 student.dat 文件。

关闭二进制文件仍使用 Close 语句。

对于二进制文件，不能指定字节长度。每个打开的二进制文件都有一个自己的文件指针，文件指针是一个数字值，指向下一次读写操作的文件中的位置，每个“位置”对应一个数据字节，因此，有 n 个字节的文件，就有 1 到 n 个位置。

可以用 Seek() 函数返回当前的文件指针位置（即下一个要读写的字节）；用 Loc() 函数返回上一次读写的字节位置，除非用 Seek 语句移动了文件指针，Loc() 返回值总比 Seek() 的小 1。

2. 读写二进制文件

读写二进制文件的方法和读写随机文件的方法基本相同，用 Get 语句读二进制文件，用 Put 语句写二进制文件。它们的格式如下：

```
Get # FileNumber , [Pos], Var
```

```
Put # FileNumber , [Pos], Var
```

其中，参数 FileNumber 是以二进制方式打开的文件号；Pos 用来指定读写操作发生时的字节位置，若省略，则使用当前的文件指针位置。

Get 语句从文件的中指定的位置开始读取数据，并将读取的数据存放在变量 Var 中，所读取的数据的长度由 Var 变量包含的字节长度决定。

Put 语句从二进制文件的中指定的位置开始写入数据，Var 是用来存放被写入的数据的变量，所写入的数据的长度由 Var 变量的字节长度决定。

对于二进制文件长度的判断，可以使用 LOF() 函数和 EOF() 函数检查文件的结尾位置，LOF() 函数返回文件的长度，EOF() 函数判断是否到达文件的结尾。

【例 13.6】编制一个通用过程，实现复制指定文件的功能。

所谓文件复制，就是将源文件按二进制打开，逐字节地读出数据，并依次写入给定的目标文件中。

```

Sub MyCopy(str1 As String, str2 As String)
    Dim ar As Byte, i As Integer
    Open str1 For Binary As #1
    Open str2 For Binary As #2
    For i = 1 To LOF(1)
        Get #1, , ar
        Put #2, , ar
    Next i
    Close #1, #2
End Sub

```

上述通用过程 MyCopy 中的字符串参数 str1 代表源文件名称，str2 代表目标文件名称。例如，将文件 face01.ico 复制到 face0011.ico 中，则调用格式如下：

```
MyCopy "face01.ico", "face0011.ico"
```

此外，VB 还提供了若干有关文件读写操作的函数，以下是几个比较常用函数，其中有的已经在前面使用过。

(1) LOF 函数

格式：LOF (文件号)

功能：返回一个已打开文件的大小，类型为 Long，单位是字节。

(2) FileLen 函数

格式：FileLen (文件名)

功能：返回一个未打开文件的大小，类型为 Long，单位是字节。文件名可以包含驱动器以及目录。

(3) EOF 函数

格式：EOF (文件号)

功能：用于判断读取的位置是否已到达文件尾。当读到文件尾时，返回 True，否则返回 False。对于顺序文件，用 EOF 函数测试是否到达文件尾；对于随机文件和二进制文件，如果读不到最后一个记录的全部数据，返回 True，否则返回 False。对于以 Output 方式打开的文件，EOF 函数总是返回 True。

(4) Loc 函数

格式：Loc (文件号)

功能：返回文件当前读/写的位置，类型为 Long。对于随机文件，返回最近读/写的记录号；对于二进制文件，返回最近读/写的字节的位置。对于顺序文件，返回文件中当前字节位置除以 128 的值。对于顺序文件而言，Loc 的返回值无实际意义。

(5) Shell 函数

格式：Shell(<文件名> [, 窗口样式])

功能：执行一个可执行文件，如果成功的话，返回一个代表这个程序的任务 ID，若不成功，则会返回 0。窗口样式表示在程序运行时窗口的样式，取值 0~6，通常取值 1，表示程序以正常大小的窗口打开并且拥有焦点。

例如，调用 c:\pwin98\calc.exe 程序，则语句为：

```
str1=Shell("c:\pwin98\calc.exe",1)
```

(6) FileDateTime 函数

格式：FileDateTime(文件名)

功能：返回文件被创建或最后修改后的日期和时间。

(7) FreeFile 函数

格式：FreeFile

功能：返回一个可用的文件号。

13.3 文件系统对象

13.3.1 文件系统对象的种类

FSO 对象模型包含 Drive、Folder、File、TextStream 和 FileSystemObject 五个对象。

- Drive 对象：驱动器对象。用来收集驱动器信息，如可用磁盘空间或驱动器的类型。

- Folder 对象：文件夹对象。用于创建、删除或移动文件夹，同时可以进行向系统查询文件夹的路径等操作。

- File 对象：文件对象。用于创建、删除或移动文件，同时可以进行向系统查询文件的路径等操作。它的操作和 Folder 基本相同，所不同的是针对磁盘上的文件进行的。

- TextStream 对象：用来完成对文本文件的读写操作。

- FileSystemObject 对象：是 FSO 对象模型中的主对象，它提供了一套完整的可用于创建、删除文件和文件夹，收集驱动器、文件夹和文件相关信息的方法

FSO 对象模型提供的方法是有冗余的。也就是说，FileSystemObject 对象的某些方法与其他对象的某些方法执行的是同样的操作。例如，使用 FileSystemObject 对象的 CopyFile 方法或者使用 File 对象的 Copy 方法，均可以完成文件复制操作。

FSO 对象模型包含在 Scripting 类型库 (Scrrun.Dll) 中，所以在使用前需要在工程中引用这个文件：选择【工程】菜单【引用】命令，然后在【引用】对话框中选中“Microsoft Scripting Runtime”前的复选框，然后单击【确定】按钮。

在工程中引用了 FSO 对象模型后，便可以创建 FileSystemObject 对象，这是使用 FSO 对象模型的第一步。创建 FileSystemObject 对象可以采用两种方法，一种是将一个变量声明为 FSO 对象类型：

```
Dim fsoTest As New FileSystemObject
```

或

```
Dim fsoTest As FileSystemObject
```

另一种是通过 CreateObject 方法创建一个 FSO 对象：

```
Set fsoTest = New FileSystemObject
```

```
Set fsoTest = CreateObject("Scripting.FileSystemObject")
```

上述语句均创建了 FileSystemObject 对象 fsoTest。在实际使用中具体采用哪种声明方法，可根据个人的使用习惯而定。

完成了 FSO 对象模型的创建之后，就可以使用 FileSystemObject 对象的方法，或者使用某个具体对象的方法及属性来获取所需信息或进行相关操作

了。

在下面的叙述中，假定已经创建了 FileSystemObject 对象 fsoTest。

13.3.2 使用文件系统对象

1. 访问驱动器

有两种访问驱动器的方式：通过 Drive 对象和通过 FileSystemObject 对象访问访问。

(1) 通过 FileSystemObject 对象访问驱动器

Drives 属性：该属性是一个 Drive 对象的集合，它包含了所有有效的驱动器。若要取得每个驱动器信息，需使用 For Each ... Next 循环语句，并将循环变量定义为 Variant 类型。例如，下列程序段显示出系统中的所有驱动器：

```
Dim fsoTest As New FileSystemObject
Dim drv As Variant
For Each drv In fsoTest.Drives
    Print drv
Next
```

DriveExists 方法：判断一个指定的驱动器是否存在，如果存在返回 True，否则返回 False。

GetDrive 方法：返回一个与指定的驱动器相对应的 Drive 对象。

GetDriveName 方法：从指定的路径中返回驱动器名称。

例如：

```
Dim fsoTest As New FileSystemObject
Dim drv As Drive, str As String
str = fsoTest.GetDriveName("e:\abc.txt")
Print str          ' 显示字符串 "e:"
If fsoTest.DriveExists("D") Then
    Print "驱动器 E 存在"
End If
' 获得一个 Drive 对象
Set drv = fsoTest.GetDrive("D")
```

(2) 通过 Drive 对象访问驱动器

通过 Drive 对象可以访问一个具体驱动器的详细信息。取得一个 Drive 对象的通常做法如下：

```
Dim drv1 As Drive
Set drv1 = fsoTest.GetDrive("C:\")
```

Drive 对象没有方法，其应用都是通过属性表现出来的，见教材表 13.1，这些属性基本上包含了日常操作所需的全部信息，因此在使用中非常方便。

【例 13.7】使用 Drive 对象显示 C 盘驱动器的有关信息。

首先在 VB 中建立一个工程，然后添加一个命令按钮，将其 Caption 设置为“驱动器信息”，然后在按钮的 Click 事件中加入以下代码：

```
Dim fsoTest As New FileSystemObject
Dim drv1 As Drive, sReturn As String
Set drv1 = fsoTest.GetDrive("C:\")
```

```

sReturn = "驱动器：" & "C:\\" & vbCrLf
sReturn = sReturn & "卷标名：" & _
    drv1.VolumeName & vbCrLf
sReturn = sReturn & "总空间：" & _
    FormatNumber(drv1.TotalSize / 1024, 0)
sReturn = sReturn & "Kb" & vbCrLf
sReturn = sReturn & "可用的磁盘空间：" & _
    FormatNumber(drv1.FreeSpace / 1024, 0)
sReturn = sReturn & "Kb" & vbCrLf
sReturn = sReturn & "文件系统类型：" & _
    drv1.FileSystem & vbCrLf

```

MsgBox sReturn

程序运行时，单击“驱动器信息”按钮就会弹出一个消息框显示 C 盘的信息，如图 13.9 所示。

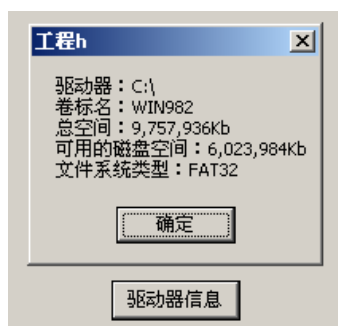


图 13.9 例 13.7 运行结果

2. 访问文件夹

(1) 通过 FileSystemObject 对象访问文件夹

FileSystemObject 对象中提供的有关文件夹的主要方法如下：

CreateFolder 方法：创建一个文件夹。

DeleteFolder 方法：删除文件夹。可使用通配符。

MoveFolder 方法：移动文件夹。可使用通配符。

CopyFolder 方法：复制文件夹。可使用通配符。

FolderExists 方法：查找一个文件夹是否存在。

GetFolder 方法：获得一个 Folder 对象。

GetParentFolderName 方法：获得一个文件夹的父文件夹的名称。

(2) 通过 Folder 对象访问文件夹

通过 Folder 对象也可以访问一个文件夹，下列语句获得一个 Folder 对象：

```
Dim fld As Folder
```

```
Set fld = fsoTest.GetFolder("E:\xwx2004")
```

Folder 对象常用的方法和属性见教材表 13.2。注意 FS0 对象模型包含的方法是冗余的，表中 FileSystemObject 对象的 DeleteFolder、MoveFolder、CopyFolder 方法和 Folder 对象的 Delete、Move、Copy 方法，在功能上是分别对应的，可以任选其中的一种使用。

【例 13.8】Folder 对象应用实例。

新建一个工程，在 Form1 的“通用-声明”部分加入以下代码：

```
Option Explicit
```

```
Dim fsoTest As New FileSystemObject
```

```
Dim folder1 As Folder
```

在窗体上添加三个命令按钮，分别是“建立文件夹”按钮 CmdCreate、“删除文件夹”按钮 CmdDelete、“文件夹信息”按钮 CmdGetPro，它们的功能分别是建立文件夹 C:\Test、删除文件夹 C:\Test、显示文件夹 C:\Windows 的部分信息。

三个命令按钮的 Click 事件中的代码详见教材。

运行后，当单击【文件夹信息】按钮时的状态如图 13.10 所示。



图 13.10 例 13.8 的运行结果

3. 访问文件

(1) 通过 FileSystemObject 对象访问文件

FileSystemObject 对象中提供的有关文件操作的主要方法如下：

DeleteFile 方法：删除文件。可使用通配符。

MoveFile 方法：移动文件。可使用通配符。

CopyFile 方法：复制文件。可使用通配符。

FileExists 方法：查找一个文件是否存在。

GetFile 方法：获得一个现有 File 对象。

以下是几个应用示例：

删除一个文件：

```
fsoTest.DeleteFile "e:\test\t1.txt"
```

删除多个文件：

```
fsoTest.DeleteFile "e:\test\*.txt"
```

同名移动一个文件：

```
fsoTest.MoveFile "e:\t2.txt", "e:\"
```

改名移动一个文件：

```
fsoTest.MoveFile "e:\t2.txt", "e:\test\t22.txt"
```

复制一个文件：

```
fsoTest.CopyFile "e:\test\t22.txt", "e:\"
```

复制多个文件：

```
fsoTest.CopyFile "e:\test\*.txt", "e:\"
```

判断文件是否存在：

```
If fsoTest.FileExists("e:\test\t22.txt") Then
```

```
    Print "yes"
```

```
Else
```

```
    Print "no"
```

```
End If
```

(2) 通过 File 对象访问文件

下列语句获得一个 File 对象: `Dim f As File`

`Set f= fsoTest.GetFile ("E:\xwx2004\daan.doc")`

有关 File 对象的属性以及复制、删除、移动等操作均与 Folder 对象类似, 见教材表 13.3。

【例 13.9】设计一个通用过程, 接收以字符串表示的文件名 (包含路径) 后, 显示指定文件的相关信息。

在窗体的通用段建立 GetFileInfo 过程, 并在窗体的 Click 事件中调用该过程 (代码详见教材), 可显示指定文件的相关信息, 如图 13.11 所示。

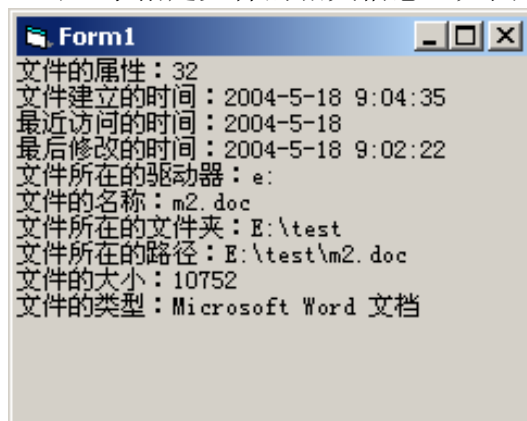


图 13.11 例 13.9 运行结果

4. 文件读写

FSO 对象模型尚不支持随机文件和二进制文件的读写操作, 仅支持文本文件。对随机文件和二进制文件的读写仍然需要使用传统的 Open 等语句完成, 对文本文件则可以使用 FSO 对象模型提供的 TextStream 对象 (文本流) 进行读写操作。

下面仍然假定已经创建了 FileSystemObject 对象 fsoTest。

(1) 建立 TextStream 对象

可以使用三种方法之一建立 TextStream 对象。

① 使用 FileSystemObject 对象的 CreateTextFile 方法。该方法新建一个空文本文件, 并返回相应的一个 TextStream 对象。

语法格式:

```
object.CreateTextFile(filename [, overwrite [, unicode]])
```

其中, 参数 filename 是一个字符串, 指明文件的名称。参数 overwrite 表示如果指定的文件已经存在, 是否允许覆盖, 为 True 则覆盖, 为 False 则不能覆盖, 它的默认值是 False。参数 unicode 表示文件是否作为 Unicode 文件创建, 为 True 则创建一个 Unicode 文件, 为 False 则创建一个普通 ASCII 文件, 默认值是 False。

例如, 在 E:\test 目录下创建文本文件 op.txt:

```
Dim txtf As TextStream
```

```
Set txtf = fsoTest.CreateTextFile("e:\test\op.txt")
```

② 使用 FileSystemObject 对象的 OpenTextFile 方法。该方法打开一个现有的文本文件, 并返回相应的一个 TextStream 对象。语法格式:

```
object.CreateTextFile(filename [, iomode [, create]])
```

其中, 参数 filename 指明文件的名称。参数 iomode 表示文件的打开方式,

取值 ForReading 代表用于读操作, ForWriting 用于写操作, ForAppend 用于追加数据。参数 create 表示如果指定的文件不存在, 是否创建新文件, 为 True 则创建, 为 False 则不创建, 默认值是 False。例如, 以读方式打开 E:\test 目录下的文本文件 op.txt:

```
Dim txtf As TextStream
Set txtf = fsoTest.OpenTextFile("e:\test\op.txt", _
    ForReading, False)
```

若需产生一个临时文件, 可以使用 FileSystemObject 对象的 GetTempName 方法生成临时文件的名称, 进而使用上述两种方法之一打开一个临时文件。比如:

```
Dim txtf As TextStream
Dim strFileName As String
' 获取随机产生的临时文件名
strFileName = fsoTest.GetTempName()
Print strFileName
' 建立临时文件
Set txtf = fsoTest.CreateTextFile(strFileName)
```

③ 使用 File 对象的 OpenAsTextStream 方法。通过 File 对象的 OpenAsTextStream 方法也可以获得 TextStream 对象, 语法格式:

```
object.OpenAsTextStream([iomode])
```

参数 iomode 表示文件的打开方式, 取值 ForReading 代表用于读操作 (默认值), ForWriting 用于写操作, ForAppend 用于追加数据。

在下面的程序段中, 首先创建了 File 对象, 然后使用 OpenAsTextStream 方法建立 TextStream 对象:

```
Dim txtf As TextStream
Dim str As String
Dim f As File
Set f = fsoTest.GetFile("e:\test\op.txt")
Set txtf = f.OpenAsTextStream
```

(2) 使用 TextStream 对象

对于文本文件, 一旦建立了相应的 TextStream 对象, 就可以使用如表 13.4 (见教材) 所列该对象的方法及属性进行文件读写操作。

假定有如下声明和赋值语句:

```
Dim fsoTest As New FileSystemObject ' 声明 FSO 对象
' 声明文本流对象
Dim txtf1 As TextStream, txtf2 As TextStream
Dim str As String
Dim f1 As File, f2 As File ' 声明文件对象
' 获取文件对象
Set f1 = fsoTest.GetFile("e:\test\op.txt")
Set f2 = fsoTest.GetFile("e:\test\m2.txt")
' 读方式打开文件
Set txtf1 = f1.OpenAsTextStream(ForReading)
' 追加方式打开文件
Set txtf2 = f2.OpenAsTextStream(ForAppending)
```

执行上述语句后，txtf1 是以读方式打开的 TextStream 对象，对应文件“e:\test\op.txt”；txtf2 是以写方式打开的 TextStream 对象，对应文件“e:\test\m2.txt”，以下是几个用法示例：

从 txtf1 中读出 5 个字符存入 str 中：

```
str = txtf1.Read(5)
```

从 txtf1 中读出一行存入 str 中：

```
str = txtf1.ReadLine
```

从 txtf1 中读出全部数据存入 str 中：

```
str = txtf1.ReadAll
```

跳过 txtf1 中的 5 个字符：

```
txtf1.Skip(5)
```

跳过 txtf1 中的一行：

```
txtf1.SkipLine
```

关闭 txtf1：

```
txtf1.Close
```

将字符串“hello”写入到 txtf2 中：

```
txtf2.Write "hello"
```

向 txtf2 中写入 2 个空行：

```
txtf2.WriteLine(2)
```

将字符串“sky”写入到 txtf2 中并占一行：

```
txtf2.WriteLine "sky"
```

关闭 txtf2：

```
txtf2.Close
```

【练习】

1. 文件类型控件是如何实现联动。
2. 举例说明 VB 中的 Open 语句如何打开顺序文件、随机文件和二进制文件？
3. 设计一个如图 13.12 所示的图像显示程序。当用户依次选择了驱动器、目录、图像文件名后，将图像显示出来。
4. 使用 OpenFileDialog 控件和 FileCopy 语句，设计一个完成文件复制功能的程序。
5. 试使用随机文件编写一个同学通讯录管理程序，具有显示、添加功能。

第 14 章 数据库应用基础

本章要点：

- 数据库的建立
- 访问数据库
- SQL 语句
- 数据库记录的操作

14.1 创建数据库

14.1.1 关系型数据库的基本结构

在讨论关系型数据库的结构之前，先来看一个记载学生基本信息的表格。

表 14.1 学生基本信息表

学号	姓名	性别	出生日期	专业	班级
030101001	张三	男	1983-6-5	计算机应用	03 级 1 班
030102002	李四	男	1983-10-8	计算机应用	03 级 2 班
030201001	李梅	女	1983-8-12	计算机网络	03 级 1 班
...	...	...	...	...	...

表 14.1 是一个由若干行和列组成的二维表格，一个关系型数据库由多个这样的二维表格组成。关系型数据库使用以下术语描述数据库中的信息：

记录(Record)：二维表中的每一行为一条记录(表 14.1 中的第一行“学号”、“姓名”等列标题除外)。一个表中不允许含有完全相同的两条记录。

字段(Field)：二维表中的每一列为一个字段。列标题为字段名(必须惟一)。

数据表(Table)：二维表中的所有记录构成数据表，简称为表。

数据库(Database)：多个相互关联但不同名的数据表构成数据库。

目前较流行的桌面数据库 MS Access, 大型网络数据库 MS SQL Server、Oracle 和 Sybase 等都属于关系型数据库。本章以 MS Access 数据库为例讨论 VB 的数据库应用技术。

14.1.2 在 VB 环境中创建 Access 数据库

1. 启动数据管理器

在 VB 环境中执行【外接程序】菜单中的【可视化数据管理器】命令，打开可视化数据管理器 (VisData)，如图 14.1 所示。

2. 建立数据库

在 VisData 窗口执行菜单命令【文件】|【新建】|Microsoft Access Ver 7.0 MDB，打开【选择要创建的 Microsoft Access 数据库】对话框，在对话框中输入数据库文件名(如“Student.mdb”)并保存后，VisData 窗口的工作区将出现如图 14.2 所示的【数据库窗口】(此时为空库，无表)。



图 14.1 可视化数据管理器



图 14.2 数据库窗口

3. 建立数据表

右击【数据库窗口】空白处，在弹出菜单中选择【新建表】菜单项，打开如图 14.3 所示的【表结构】对话框，输入表名称（如“基本情况”）后，单击【添加字段】按钮，打开如图 14.4 所示的【添加字段】对话框，输入字段名称，设置类型和大小（仅 Text 类型可设置大小）。添加了所有字段后，单击图 14.3 中的【生成表】按钮即可建立数据表。在一个库中可建立多个不同名称的表。

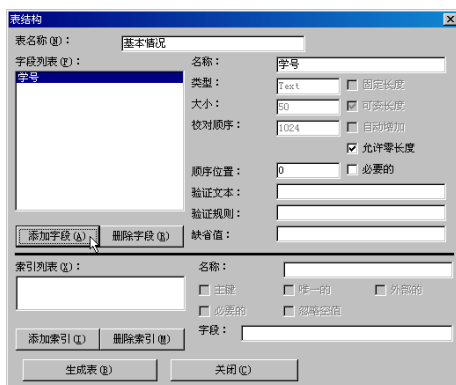


图 14.3 表结构



图 14.4 添加字段

4. 添加索引

为数据表添加索引可以提高数据检索的速度。在图 14.3 所示的【表结构】对话框中单击【添加索引】按钮，打开如图 14.5 所示的【添加索引到 基本情况】对话框。在【名称】文本框中输入索引名称（如“sNo”），在【可用字段】列表框中选择需要为其设置索引的字段（如“学号”），并设置是否为主索引或惟一索引（无重复）。

5. 输入记录

双击【数据库窗口】中数据表名称左侧的图标，打开如图 14.6 所示的记录操作窗口，可以对记录进行增、删、修改等操作。

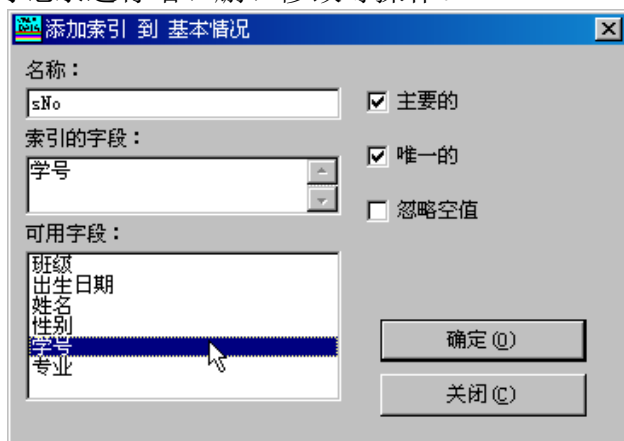


图 14.5 添加索引

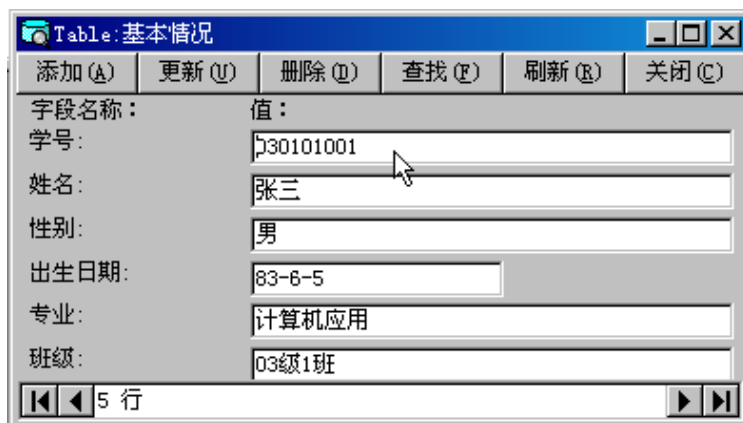


图 14.6 记录操作

14.1.3 用 MS Access 建立数据库

下面以 MS Access 2000 为例简介数据库的创建。

1. 建立数据库

启动 MS Access，在对话框中选定【空 Access 数据库】单选按钮。单击【确定】按钮后，在【文件新建数据库】对话框中选择保存位置并输入文件名，然后单击【创建】按钮。

2. 建立数据表

新建一个空白数据库后，在 MS Access 主窗口中将会出现如图 14.9 所示的数据库窗口。在此窗口中可以管理 Access 数据库的各组成部分。

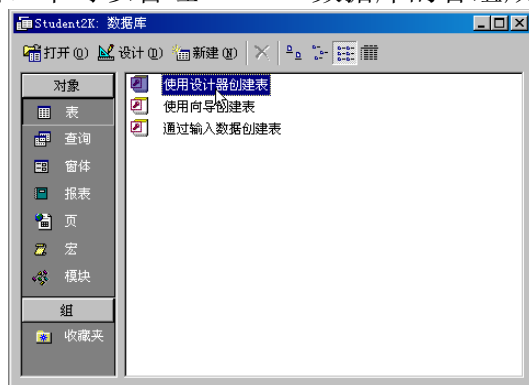


图 14.9 创建数据表

在数据库窗口中双击【使用设计器创建表】图标，打开如图 14.10 所示的表设计器窗口【表 1: 表】，输入字段名称，设置字段的数据类型、字段大小及其他属性。



图 14.10 设计表的结构

若需设置主键，可选定拟设为主键的字段，然后单击 MS Access 主窗口工具栏中的【主键】图标，此时，被设为主键的字段名左侧会出现钥匙状的图标，同时，【字段属性】中的【索引】属性将自动设为【有(无重复)】。

全部字段设置结束后，关闭表设计器窗口，系统将显示如图 14.11 所示的对话框，可根据提示保存新建的数据表并设置表的名称。

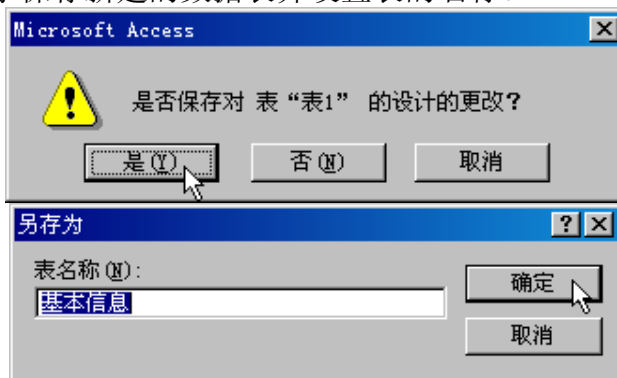


图 14.11 保存数据表

若需修改数据表的结构定义（如添加、删除或修改字段），可在如图 14.12 所示的数据库窗口选定数据表（如“基本信息”），然后单击该窗口工具栏中的【设计】按钮，打开前面图 14.10 所示的表设计器窗口进行操作。

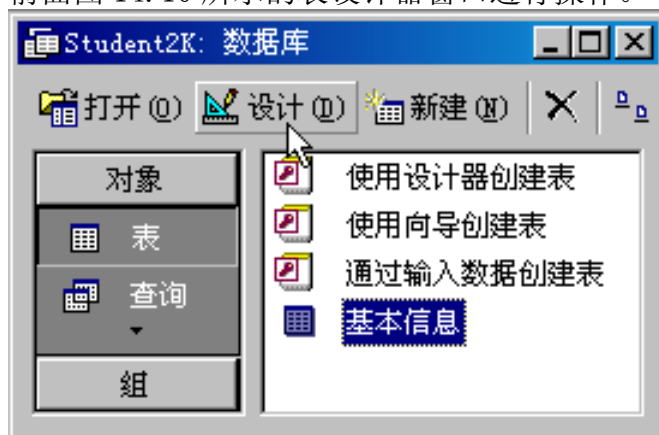


图 14.12 修改表结构

如果要添加一个新表，可再次双击【使用设计器创建表】图标，或者单击工具栏【新建】按钮，在如图 14.13 所示的【新建表】对话框中选择【设计视图】后，单击【确定】按钮，均可打开如图 14.10 所示的表设计器窗口。

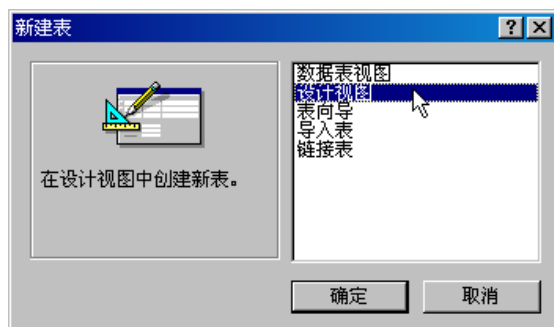


图 14.13 添加新表

3. 输入记录

在数据库窗口中双击数据表, 或者选定表后单击工具栏中的【打开】按钮, 打开如图 14.14 所示的数据表窗口, 向表中输入数据。输入结束后关闭该窗口, 根据系统提示保存数据表。



图 14.14 输入记录

4. 建立表间关联关系

在一个数据库中, 一般需要用多个表存放不同类别而又相互关联的信息。例如, 在学生信息数据库中用“基本信息”表存放学生的学号、姓名、性别等基本情况, 用“成绩”表存放学生的各科成绩, 用“课程”表存放已开的课程。假设这三个表中含有如表 14.3~表 14.5 所示的信息, 当需要查询某位学生的一门或几门课程的成绩时, 就要从上述三个表中获取数据。假如某位学生的学号在最初输入时有误, 需要修改, 则必须确保“基本信息”表和“成绩”表中的“学号”字段进行同步更改。因此, 应当为三个表建立必要的关联关系

表 14.3 基本信息

学号*	姓名	性别	出生日期
030101001	张三	男	1983-6-5
030102002	李四	男	1983-10-8
030201001	李梅	女	1983-8-12
030201002	王五	男	1983-3-22
...	...	...	...

表 14.4 成绩

学号	课号	分数
030101001	001	78
030101001	004	82
030201001	002	80
030201001	003	76
...	...	...

表 14.5 课程

课号*	课程名称
001	英语
002	数据结构
003	C 语言程序设计
004	VB 程序设计
...	...

注: * 为主键。

建立表间关联关系的前提是两个表各含有一个关联字段 (属性必须相同), 其中一个表的关联字段必须被设为主键或具有惟一索引, 该表称为“主表”, 另

一个表称为“从表”。下面以表 14.3～表 14.5 为例，简介建立数据表之间关联关系的一般步骤。

① 单击 Microsoft Access 主窗口工具栏【关系】按钮，若数据库中尚未定义任何关系，则在打开【关系】窗口的同时弹出如图 14.15 所示的【显示表】对话框。

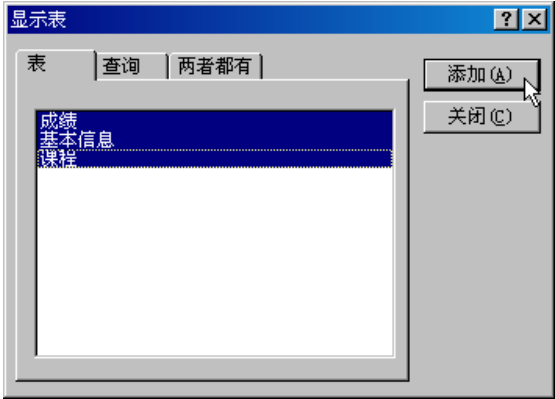


图 14.15 选择拟建立关系的表

② 在【显示表】对话框中选定需要建立关系的表，单击【添加】按钮，然后单击【关闭】按钮，屏幕显示如图 14.16 所示的【关系】窗口。

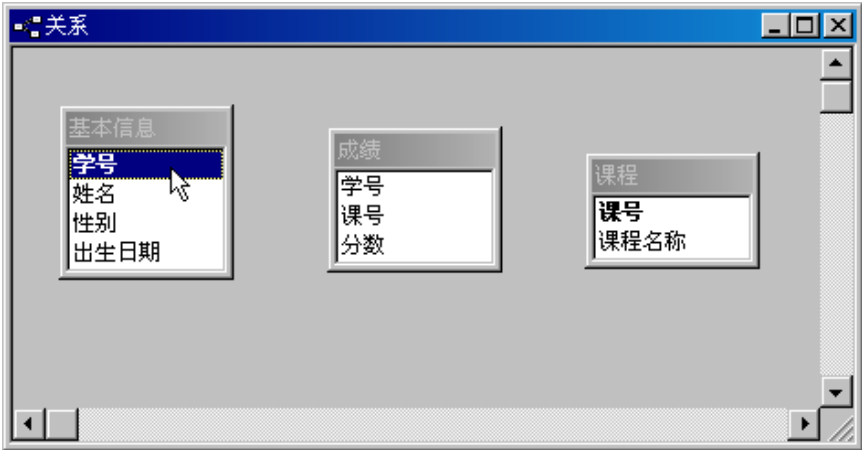


图 14.16 【关系】窗口

③ 在【关系】窗口将【基本信息】表中的【学号】字段拖放到【成绩】表中的【学号】字段，弹出如图 14.17 所示的【编辑关系】对话框。

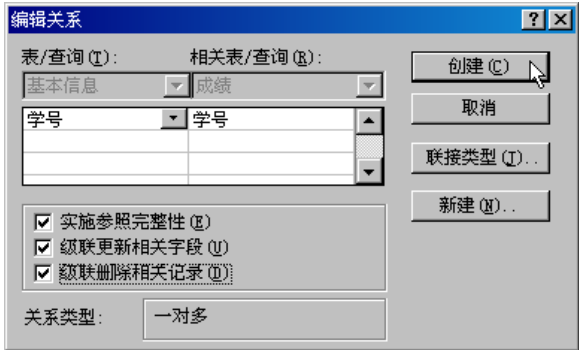


图 14.17 编辑关系

④ 在【编辑关系】对话框中，将【实施参照完整性】、【级联更新相关字段】和【级联删除相关记录】三个复选框全部选中，单击【创建】按钮。

⑤ 重复第③、④步的操作，建立【成绩】表中的【课号】字段与【课程】

表中的【课号】字段的关联。

建立表间关联关系后的效果如图 14.18 所示。



图 14.18 表间关系

14.2 VB 如何访问数据库

目前，Visual Basic 访问数据库的主流技术是 ADO。ADO 是一种基于对象的数据访问接口，在 VB 中提供了利用 ADO 访问数据库的两种主要形式：ADO 数据控件（ADODC）和 ADO 对象编程模型（ADO 代码）。这两种方式可以单独使用，也可以同时使用。

使用 ADO 数据控件的优点是代码少，一个简单的数据库应用程序甚至可以不用编写任何代码。它的缺点是功能简单，不够灵活，不能满足编制较复杂的数据库应用程序的需要。

使用 ADO 对象编程模型的优点是具有高度的灵活性，可以编制复杂的数据库应用程序。它的缺点是代码编写量较大，对初学者来说有一定困难。

无论采用哪种方式访问数据库，都要经历以下基本步骤：

- ◆ 与数据库建立连接，打开数据库。
- ◆ 从数据库中读取数据并在适当的控件中显示。
- ◆ 对所获数据进行浏览以及增、删、改等操作，并将修改后的数据存入数据库。

在后面的几节中将以 ADO 数据控件为主详细介绍 VB 访问数据库的基本操作。

14.3 用控件访问数据库

14.3.1 ADO 数据控件

1. 加载 ADO 数据控件

ADO 数据控件属于 ActiveX 控件，加载后才能使用：

右击工具箱，在弹出菜单中选择【部件】菜单项，打开【部件】对话框，在【控件】选项卡的列表中选中“Microsoft ADO Data Control 6.0”前面的复选框，单击【确定】按钮。

2. 连接数据库及指定记录源

ADO 数据控件与数据库的连接有 3 种方式：数据链接文件（.UDL）、ODBC（DSN）和字符串连接。与 Access 数据库建立连接的常用方式是字符串连接。

通常通过属性页一次完成连接数据库和指定记录源的设置。操作步骤如下：

① 将 ADO 数据控件 (Adodc) 添加到窗体上，右击该控件，在弹出菜单中选择【ADODC 属性】菜单项，打开如图 14.20 所示的【属性页】对话框。

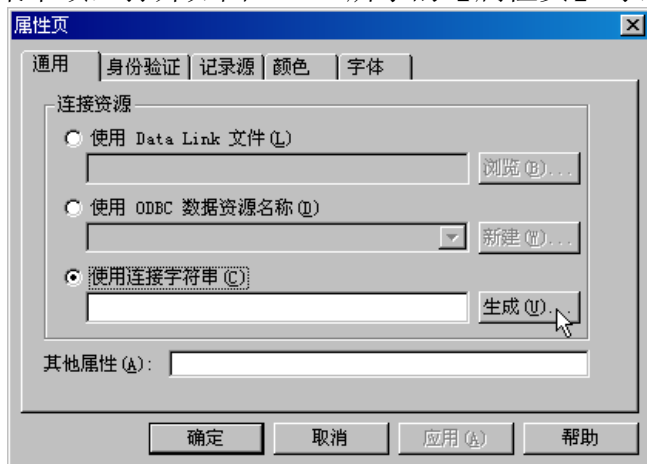


图 14.20 ADODC 属性页

② 在对话框【通用】选项卡中选择【使用连接字符串】，单击【生成】按钮，打开如图 14.21 所示的【数据链接属性】对话框。在【提供程序】选项卡的列表中选择“Microsoft Jet 4.0 OLE DB Provider”，单击【下一步】，切换到如图 14.22 所示的【连接】选项卡。



图 14.21 选择提供程序

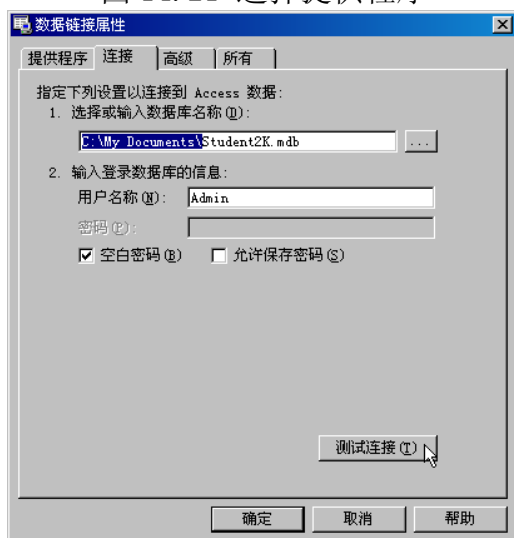


图 14.22 连接数据库

③ 在【连接】选项卡中单击【1. 选择或输入数据库名称】输入框右侧的按钮，在弹出的【连接 Access 数据库】对话框中选择数据库，单击【打开】按钮后返回【连接】选项卡，单击【测试连接】按钮，成功后单击【确定】，完成连接数据库的设置，返回【属性页】对话框。

④ 单击【属性页】对话框【记录源】选项卡，显示如图 14.23 所示的界面，在【记录源】选项卡中设【命令类型】为“2-adCmdTable”，然后在【表或存储过程名称】下拉列表中选择数据表。也可以设【命令类型】为“1-adCmdText”或“8-adCmdUnknown”，然后在【命令文本(SQL)】文本框中输入 SQL 语句(如图 14.24 所示)。最后单击【确定】按钮完成设置。

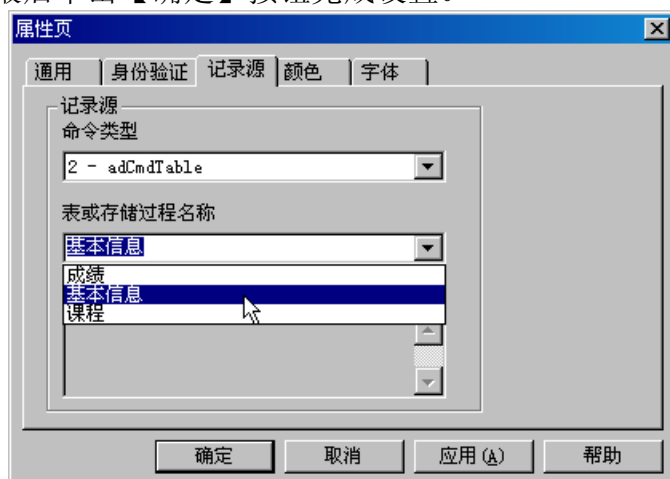


图 14.23 用数据表作记录源

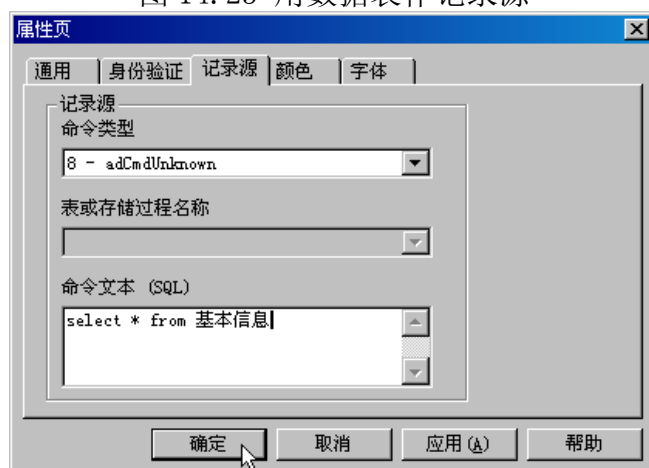


图 14.24 用 SQL 语句作记录源

上述操作实际上是设置了 ADO 数据控件的两个重要属性：

ConnectionString（连接字符串）属性用于建立与数据库的连接。

RecordSource（记录源）属性用于指定记录源。

除了使用属性页之外，也可以通过属性窗口或程序代码设置这两个属性。

在设置 ADO 数据控件与数据库的连接时，有一点要提请读者注意：

如图 14.22 所示，在【数据链接属性】窗口的【连接】选项卡中指定数据库时采用的是绝对路径。为了保证数据库应用程序移植到其它计算机上仍可正常使用，应当采用相对路径，即在测试连接成功后删除数据库名称前面的所有路径(图 14.22 输入框中的反相显示部分)，仅保留数据库文件名。将数据库文件与工程文件存放在同一文件夹下，在工程启动窗体的 Initialize 事件过程中进行路径

初始化处理:

```
Private Sub Form_Initialize()  
    ChDrive App.Path ' 设当前驱动器为工程所在驱动器  
    ChDir App.Path   ' 设当前目录为工程所在目录  
End Sub
```

3. 用代码设置或改变记录源

ADO 数据控件一旦建立了与数据库的连接, 就可以通过设置或改变其 RecordSource (记录源) 属性访问数据库中的任何表, 亦可访问由一个或多个表中的部分或全部数据构成的记录集。在实际应用中, 常常在程序运行时用代码设置 RecordSource 属性及其相关属性 (如 CommandType), 从而使 ADO 数据控件具有更大的灵活性。例如:

① 用数据表名称作为记录源:

```
Adodc1.CommandType = adCmdTable ' 设置命令类型为数据表  
Adodc1.RecordSource = "基本信息"  
Adodc1.Refresh
```

② 用 SQL 语句生成的记录集作为记录源:

```
Adodc1.CommandType = adCmdText ' 设置命令类型为 SQL 语句  
Adodc1.RecordSource = "SELECT * FROM 基本信息"  
Adodc1.Refresh
```

上述两段代码的效果相同。有关 SQL 语言的应用将在 14.5 节介绍。

注意: 设置记录源后, 必须调用 ADO 数据控件的 Refresh 方法刷新对数据库的访问。

14.3.2 数据绑定控件

ADO 数据控件本身不能显示数据, 需通过绑定具有显示功能的其他控件显示数据, 这些控件称为数据绑定控件或数据识别 (感知) 控件, 如文本框、DataGrid、标签、图像 (片) 框、列表框、组合框、复选框等。其中最常用的是 DataGrid 和文本框。

1. 数据绑定控件的相关属性

DataSource (数据源) 属性: 指定 (绑定到) ADO 数据控件。

DataField (数据字段) 属性: 绑定到特定字段。绑定后只要移动指针, 自动将修改内容写入数据库。

2. 在属性窗口设置绑定控件属性

在属性窗口将数据绑定控件的 DataSource 属性设为 ADO 数据控件 (如 Adodc1)。如果是单字段显示控件 (如文本框等), 还需将控件的 DataField 属性设置为特定字段。DataGrid 控件属于多字段显示控件, 没有 DataField 属性。

【例 14.1】用 ADO 数据控件和 DataGrid 控件创建一个简单的数据访问窗体, 显示 14.1.3 节创建的 Student2K.mdb 数据库中 “基本信息” 表的内容。

右击工具箱, 在弹出菜单中选择【部件】命令, 在对话框【控件】选项卡的列表中选中 “Microsoft ADO Data Control 6.0” 和 “Microsoft DataGrid 6.0”, 单击【确定】。选择工具箱中新增加的 ADO 数据控件和 DataGrid 控件, 将其添加到窗体上, 默认名称分别为 Adodc1 和 DataGrid1。按 14.3.1 小节所述步骤建立 Adodc1 与 Student2K.mdb 数据库的连接, 并设 Adodc1 的记录源为 “基本信息”

表。将 DataGrid1 控件的 DataSource 属性设为 Adodc1。程序运行效果如图 14. 25 所示。



图 14.25 使用 DataGrid 控件

3. 用代码设置绑定控件属性

程序运行时可以动态地设置数据绑定控件的属性。例如：

```
Set Text1.DataSource = Adodc1
Text1.DataField = "姓名"
Set DataGrid1.DataSource = Adodc1
```

说明：DataSource 是对象类型的属性，必须用 Set 语句为其赋值。

4. 不用绑定方法如何显示和处理数据

不使用绑定的方法处理数据是指不对数据显示控件的 DataSource 和 DataField 属性进行设置，而是通过代码将当前记录某个字段的值显示在控件（如文本框）中。这种方法比较灵活，缺点是代码编写量较大，其中涉及到记录集对象的操作。

（1）字段内容的显示

控件属性 = 记录集（“字段”）

例如：

```
Text1.Text = Adodc1.Recordset("学号")
Text2.Text = Adodc1.Recordset("姓名")
```

每当记录指针移动时均需对控件属性重新赋值。若需要显示的字段较多，可以编制一个自定义过程用于记录指针移动时显示各字段内容。

（2）为字段赋值

记录集（“字段”）= 控件属性

例如：

```
Adodc1.Recordset("学号") = Text1.Text
Adodc1.Recordset("姓名") = Text2.Text
Adodc1.Recordset.Update
```

说明：为字段赋值后，应调用记录集的 Update 方法更新数据库。

14. 3. 3 使用数据窗体向导

使用“数据窗体向导”可以快速创建一个数据访问窗体。

执行【工程】菜单中【添加窗体】命令，打开如图 14. 26 所示的对话框，在

【新建】选项卡中选择【VB 数据窗体向导】，单击【打开】按钮后将会出现向导的第一个对话框。

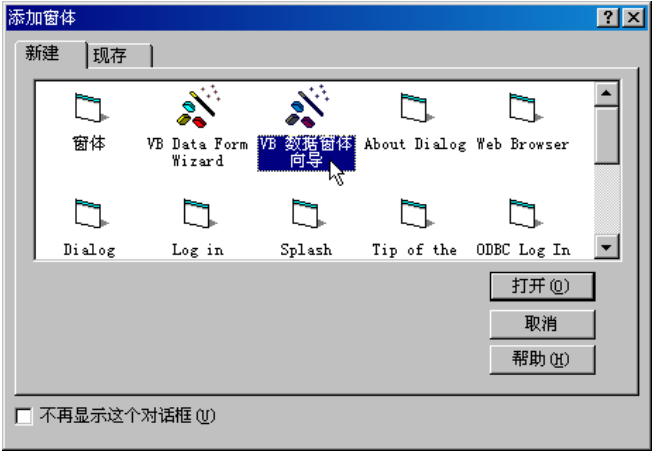


图 14.26 启动数据窗体向导

如果是创建单表访问窗体，数据窗体向导将有七个步骤：【介绍】、【数据库类型】、【数据库】、【窗体 (Form)】、【记录源】、【控件选择】和【完成】，可根据向导提示操作。

图 14.27 是利用数据窗体向导创建的数据窗体，数据库为 Student2K.mdb，记录源为“基本信息”表中的所有字段。向导中的其他步骤均采用默认设置。



图 14.27 用数据窗体向导创建的数据窗体

14.4 初识记录集对象

无论是使用 ADO 数据控件，还是使用 ADO 对象编程模型，都会涉及到记录集对象。因此，在进一步讨论数据库操作之前，有必要初步了解一下记录集对象。

请看下面来自学生数据库的几个集合：

取“基本信息”表中所有学生的记录构成一个集合；

取“基本信息”表中所有男生的记录构成一个集合；

取“基本信息”表中张三的学号和姓名，根据其学号取“成绩”表（含学号、课号和分数三个字段）中该学生的各科成绩构成一个集合。

以上几个集合都是“记录的集合”，由此得出以下概念：

将数据库中一个或多个表中的部分或全部数据构成一个“记录的集合”，这个集合就称为“记录集”（Recordset）。记录集由行（记录）和列（字段）构成。若将记录集看作一个对象，这个对象就是记录集对象。记录集对象具有特定的属性、方法和事件。

ADO 数据控件的 Recordset 属性代表属于本控件的记录集对象。

记录集对象是 ADO 中的一个功能强大的对象，对数据库的绝大部分操作，如记录指针的移动，记录的查找、添加、删除和修改等，都是针对记录集对象进行的。

14.5 用 SQL 语句生成记录集

14.5.1 最简单的 SQL 语句

1. 最简单的查询语句

下面的 SQL 语句是最简单的查询形式，生成的记录集包含整个表的全部数据：

```
SELECT * FROM 基本信息
```

其中“*”指表中所有字段(列)。FROM 子句用于指定数据表。

2. SELECT 语句的基本语法

在实际应用中，往往需要从一个或多个表中选择符合特定条件的记录构成记录集，因此应对 SELECT 语句的语法有一定的了解。以下是 SELECT 语句的基本语法。

```
SELECT * | 字段列表 FROM 表名 [WHERE 查询条件] [GROUP BY 分组字段  
[HAVING 分组条件]] [ORDER BY 排序字段 [ASC | DESC]]
```

说明：

* | 字段列表：“*”表示所有字段；“字段列表”指定字段，多个字段间用逗号分隔，来自不同表的同名字段前须加表的名称和圆点。

FROM 子句：指定表。若指定多个表，用逗号分隔。

WHERE 子句：指定选择记录的条件。

GROUP BY 及 HAVING 子句：分组过滤，将分组字段中同值记录合并为一条记录。

ORDER BY：排序。ASC 为升序(默认)；DESC 为降序。

【例 14.2】选择“基本信息”表中的“学号”和“姓名”字段，“成绩”表中的“课号”和“分数”字段构成记录集。

```
SELECT 基本信息.学号,姓名,课号,分数 FROM 基本信息,成绩 WHERE 基  
本信息.学号=成绩.学号
```

14.5.2 限定记录集筛选条件

在 SELECT 语句的各子句中，WHERE 子句使用频率最高。该子句指明查询的条件。在 WHERE 子句中可使用各种关系(比较)运算符表示筛选记录的条件。

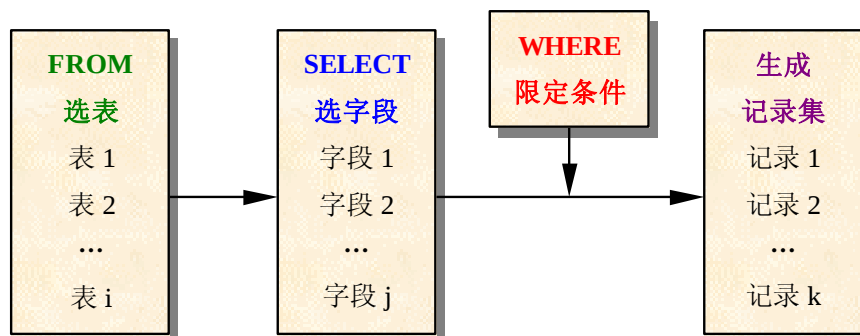


图 14.28 用SQL语句生成记录集

【例 14.3】选择“基本信息”表中所有男生构成记录集：

```
SELECT * FROM 基本信息 WHERE 性别 = "男"
```

【例 14.4】取“基本信息”表中张三的学号和姓名，根据其学号取“成绩”表中该学生的各科成绩构成记录集：

```
SELECT 基本信息.学号, 姓名, 课号, 分数 FROM 基本信息, 成绩 WHERE 基本信息.学号=成绩.学号 AND 姓名="张三"
```

在 WHERE 子句中使用 Like 运算符可实现模糊查询。SQL 语句中 Like 运算符的通配符是“%”，可代表任何字符，字符数不限。

【例 14.5】用 Like 运算符进行模糊查询。

① 查询所有姓“张”的学生：

```
SELECT * FROM 基本信息 WHERE 姓名 Like "张%"
```

② 查询所有姓名中含有“小”字的学生：

```
SELECT * FROM 基本信息 WHERE 姓名 Like "%小%"
```

③ 查询姓名最后一个字为“刚”的所有学生：

```
SELECT * FROM 基本信息 WHERE 姓名 like "%刚"
```

14.5.3 在 VB 程序中如何使用 SQL 语句

在程序代码中，SQL 语句必须以字符串形式提供。例如：

```
Adodc1.RecordSource = "SELECT * FROM 基本信息"
```

如果 SQL 语句中含有字符串常量，必须将字符串常量放在一对单引号中。

【例 14.6】在 SQL 语句中引用字符串常量。

```
Dim strSQL As String
```

```
strSQL = "SELECT * FROM 基本信息 WHERE 性别 = '男'"
```

```
Adodc1.RecordSource = strSQL
```

如果 SQL 语句中引用了 String 型变量或其他控件的字符串类型的属性（如文本框的 Text 属性），应当采用下面的引用形式（注意单引号的位置）：

```
" ... '" & 字符串变量或控件属性 & "' ... "
```

【例 14.7】在 SQL 语句中引用字符串变量。

```
Dim strSQL As String, strSex As String
```

```
strSex = "男"
```

```
strSQL = "SELECT * FROM 基本信息 WHERE 性别 = '" _  
    & strSex & "'"
```

```
Adodc1.RecordSource = strSQL
```

【例 14.8】在 SQL 语句中引用控件的字符串类型的属性。

```
Dim strSQL As String
Text1.Text = "张"
strSQL = "SELECT * FROM 基本信息 WHERE 姓名 Like ' " _
    & Text1.Text & "%' "
Adodc1.RecordSource = strSQL
```

如果 SQL 语句中引用了非字符串类型的变量或控件属性，不使用单引号。

【例 14.9】在 SQL 语句中引用非字符串类型的变量。

```
Dim strSQL As String, intGrade As Integer
intGrade = 60
strSQL = "SELECT * FROM 成绩 WHERE 分数 >= " _
    & intGrade & " AND 课程='英语'"
Adodc1.RecordSource = strSQL
```

14.5.4 记录排序

用 ORDER BY 子句（ASC 为升序，DESC 为降序）对记录排序。例如：

```
SELECT * FROM 基本信息 ORDER BY 学号 ASC
SELECT * FROM 基本信息 ORDER BY 姓名 DESC
```

14.5.5 记录分组

用 GROUP BY 子句将指定字段中数值相等的多条记录合并为一条记录。可用 HAVING 子句附加条件。

【例 14.10】以“班级”作为分组字段，查询各班女生人数。

```
SELECT 班级, Count(*) AS 女生 FROM 基本情况 GROUP BY 班级, 性别
HAVING 性别="女"
```

说明：上面的语句中 Count(*) 表示统计记录总数，AS 子句表示存放统计结果的临时字段别名。其他 SQL 函数还有 Sum、Avg、Max 和 Min 等。

14.5.6 过滤重复记录

过滤重复记录是指忽略字段值相同的重复记录。例如，假定学籍表中含有 400 名学生的信息，这些学生来自 10 个班级，现在要查询学籍表中的“班级”字段生成一个班级名称记录集，如果不进行筛选过滤，则生成的记录集将含有 400 条记录，会有很多班级名称相同的重复记录。过滤重复记录后，可以生成仅含 10 个记录的记录集，每个班级的名称都是惟一的。过滤重复记录实际上是对记录进行分类，可以通过下面的方法实现。

1. 用 DISTINCT 关键字

在 SELECT 语句中使用 DISTINCT 关键字忽略重复记录。例如，以下语句可以过滤重复记录和空字段：

```
SELECT DISTINCT 班级 FROM 学籍 WHERE 班级<>NULL AND 班级<>""
```

2. 用 GROUP BY 子句

如前所述，用 GROUP BY 子句可以对记录进行分组，实现重复记录的过滤。

例如：

```
SELECT 班级 FROM 学籍 GROUP BY 班级 HAVING 班级<>NULL AND 班级<>""
```

14.6 数据库记录的操作

14.6.1 如何移动记录指针

1. 使用 ADO 数据控件

ADO 数据控件上有 4 个导航按钮，依次为：首记录、上一记录、下一记录、末记录。使用这些按钮移动记录指针无须编写代码。

2. 使用程序代码

有时因为用户界面的需要，将 ADO 数据控件隐藏 (Visible=False)，只能通过编写代码实现记录指针的移动。具体步骤如下：

在窗体上放置 4 个命令按钮，分别为：首记录、上一记录、下一记录、末记录。

在上述按钮的单击事件中，分别用记录集对象 (ADO 数据控件的 Recordset 属性) 的 Move 方法组移动记录：

```
Adodc1.Recordset.MoveFirst      ' 首记录
Adodc1.Recordset.MovePrevious   ' 上一记录
Adodc1.Recordset.MoveNext       ' 下一记录
Adodc1.Recordset.MoveLast       ' 末记录
```

记录指针的移动如图 14.29 所示。

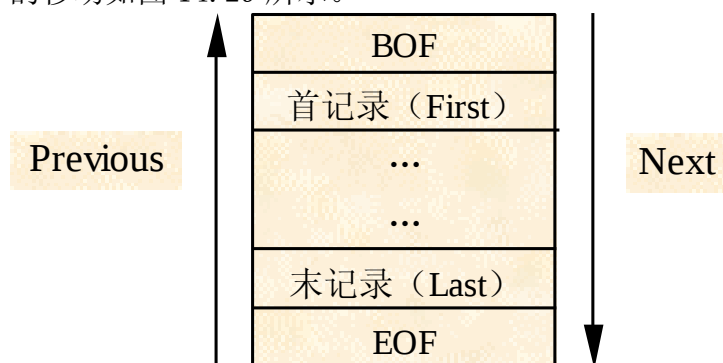


图 14.29 移动记录指针

BOF 和 EOF 是记录集的两个特殊位置，BOF 位于第一个记录之前 (记录集头)，EOF 位于最后一个记录之后 (记录集尾)。当记录指针移到 BOF 时，若再向前 (MovePrevious) 移动，或移到 EOF 时，再向后 (MoveNext) 移动，将会引发错误。采用下面的措施可以防止发生此类错误。

在“上一记录”按钮的单击事件中作如下处理：

```
Adodc1.Recordset.MovePrevious
```

```

If Adodc1.Recordset.BOF = True Then _
    Adodc1.Recordset.MoveFirst
    在“下一记录”按钮的单击事件中作如下处理：
Adodc1.Recordset.MoveNext
If Adodc1.Recordset.EOF = True Then _
    Adodc1.Recordset.MoveLast

```

14.6.2 如何查找记录

1. 用 Find 方法

记录集的 Find 方法搜索记录集中满足指定条件的记录（不区分大小写）。如果条件满足，则记录集指针定位于找到的记录上，否则指针定位于记录集的末尾（EOF）。

语法：

记录集.Find 条件[, 跳行, 搜索方向, 开始位置]

参数说明：

① 条件：字符串，类似于 SQL 语句中 WHERE 子句的条件，包括字段名、比较操作符和比较值。

如果比较操作符为“like”，则比较值可以含有一个“%”（只能放在字符串最后），实现模糊比较。例如：

“姓名 like ‘王%’”

若与变量或控件属性值比较，需用&连接。例如：

```

Adodc1.Recordset.Find “用户名= ‘” & _
    txtUserID.Text & “’”

```

② 跳行：可选，长整型值，其默认值为零，指定搜索的位移量。若该选项设为 1，连续执行 Find 方法，可依次找出每个符合条件的记录。

③ 搜索方向：可选。其值可为 adSearchForward（头→尾）或 adSearchBackward（尾→头）。

④ 开始位置：可选，变体型书签，用作搜索的开始位置。0：从当前行开始，1：从首记录开始，2：从末记录开始。。

调用 Find 方法时，通常只带有条件参数即可，调用前先将记录指针指向首记录。

【例 14.11】用记录集的 Find 方法查找记录。RecordCount 为记录总数。

’若非空记录集

```

If Adodc1.Recordset.RecordCount > 0 Then
    Adodc1.Recordset.MoveFirst
    Adodc1.Recordset.Find “姓名 = ‘张三’”
    If Adodc1.Recordset.EOF = True Then
        MsgBox “未查到姓名为张三的记录。”
    End If
End If

```

2. 用循环结构

通过循环结构遍历记录集（适用于各种类型），查找符合条件的记录（默认区分大小写）。

【例 14.12】用循环结构查找记录。

```
Adodc1.Recordset.MoveFirst
Do While Not Adodc1.Recordset.EOF
    ' 找到，退出循环
    If Adodc1.Recordset("姓名") = "李四" Then Exit Do
    Adodc1.Recordset.MoveNext
Loop
' 若指针指向记录集尾，说明未找到
If Adodc1.Recordset.EOF = True Then
    ...
Else    ' 否则，说明找到符合条件的记录
    ...
End If
```

3. 用 SQL 语句

用 SQL 语句生成只包括待查记录的记录集，若 BOF 和 EOF 均为 True，则为空记录集。

【例 14.13】用 SQL 语句查找符合条件的记录。

```
Dim strSQL As String
strSQL = "SELECT * FROM 用户 WHERE 用户名=' " _
        & txtUserID.Text & "' "
Adodc1.RecordSource = strSQL
Adodc1.Refresh
If Adodc1.Recordset.BOF = True _
    And Adodc1.Recordset.EOF = True Then
    MsgBox "无此用户！"
End If
```

14.6.3 如何添加记录

用记录集的 AddNew 方法添加记录。

语法：

记录集.AddNew [字段名，字段值]

可以先用无参数的 AddNew 方法添加一个空记录，然后为字段赋值，最后调用记录集的 Update 方法更新数据库。

【例 14.14】用记录集的 AddNew 方法添加记录。

```
Adodc1.Recordset.AddNew
Adodc1.Recordset("学号") = "030101001"
Adodc1.Recordset("姓名") = "张三"
Adodc1.Recordset.Update
' 也可以通过控件属性或变量为字段赋值，
' 例如用文本框中的内容为字段赋值：
Adodc1.Recordset.AddNew
Adodc1.Recordset("学号") = txtNo.Text
Adodc1.Recordset("姓名") = txtName.Text
```

Adodc1.Recordset.Update

注意：如果在调用 Update 方法之前移动了记录指针，系统将自动调用 Update 方法。

14.6.4 如何修改记录

为记录的字段重新赋值即可实现记录的修改。

为字段赋值的常用形式：

记录集 (“字段名”) = 新值

记录集!字段名= 新值

记录集.Fields(索引) = 新值

例如：

Adodc1.Recordset (“学号”) = “030101001”

Adodc1.Recordset!姓名 = “张三”

Adodc1.Recordset.Update

或者：

Adodc1.Recordset.Fields(0) = “030101001”

Adodc1.Recordset.Fields(1) = “张三”

Adodc1.Recordset.Update

说明：Fields 是记录集的字段集合，索引从 0 开始。修改记录后，应调用记录集的 Update 方法更新数据库。

14.6.5 如何删除记录

用记录集的 Delete 方法删除记录。

语法：

记录集.Delete [AffectRecords]

其中，AffectRecords 参数（受影响的记录）可取以下值：

AdAffectCurrent：默认。仅删除当前记录。

AdAffectGroup：删除满足记录集 Filter 属性设置的记录。

AdAffectAll：删除所有记录。

AdAffectAllChapters：删除所有子集记录。

【例 14.15】用记录集的 Delete 方法删除当前记录。

Adodc1.Recordset.Delete

Adodc1.Recordset.MoveNext

If Adodc1.Recordset.EOF And _

Adodc1.Recordset.RecordCount > 0 Then

Adodc1.Recordset.MoveLast

End If

说明：删除当前记录后，数据绑定控件（如文本框等）仍将保持已被删除的记录内容而不刷新。将记录指针移动到下一条记录，可以让用户感觉到记录已被删除，同时自动调用 Update 方法更新数据库。

14.7 ADO 编程模型简介

ADO 的核心是 Connection、Recordset 和 Command 对象。ADO 编程模型不使用 ADO 数据控件，直接用代码通过 ADO 对象访问数据库。

使用 ADO 编程模型需添加 ADO 对象类库的“引用”：

执行【工程】菜单中的【引用】命令，打开【引用】对话框，在对话框的列表选中“Microsoft ActiveX Data Objects 2.x Library”前的复选框，单击【确定】按钮。

添加“引用”后，应声明 ADO 对象变量：

Dim 变量名 As New ADODB.对象

14.7.1 ADO 的主要对象

1. Connection 对象

Connection 对象用于建立与数据源的连接。

对象变量声明示例：

Dim cnn As New ADODB.Connection

一个 Connection 对象可以为多个 Recordset 和 Command 对象提供数据库连接服务。可以在程序的标准模块中声明一个全局 Connection 对象变量，供其他模块调用。例如：

Public pubCnn As New ADODB.Connection

2. Recordset 对象

该对象表示记录集，用于记录指针的移动和记录的查找、添加、修改或删除。

对象变量声明示例：

Dim rs As New ADODB.Recordset

3. Command 对象

该对象用于对数据源执行指定的命令，如数据的添加、删除、更新或查询。

对象变量声明示例：

Dim cmm As New ADODB.Command

14.7.2 使用 ADO 编程模型的一般步骤

ADO 对象模型中对象的功能有交叉(冗余),对于一般的数据库应用程序,可不必使用 Command 对象。

1. 声明 ADO 对象变量

Dim cnn As New ADODB.Connection ' 声明连接对象

Dim rs As New ADODB.Recordset ' 声明记录集对象

2. 与数据库建立连接

为了保证数据库应用程序移植到其它计算机上仍可正常使用，应将当前工程与数据库文件保存在同一目录，并进行以下初始化处理

Dim MyPath As String ' 用于存放路径

```

MyPath = App.Path      ' 取本工程所在路径
' 若非根目录, 路径后加"\
If Right$(MyPath, 1) <> "\" Then MyPath _
    = MyPath & "\"
然后建立与数据库的连接:
cnn.Provider = "Microsoft.Jet.OLEDB.4.0" ' 指定提供者
' 设置数据源 (指定数据库)
cnn.ConnectionString = "Data Source=" & _
    MyPath & "Student.mdb"
cnn.Open      ' 与数据库建立连接

```

3. 设置记录集相关属性

设置记录集的锁定类型(LockType)、游标类型(CursorType)、记录源(Source)和使用的连接对象(ActiveConnection):

```

' 锁定类型: 开放式记录锁定
rs.LockType = adLockOptimistic
' 游标类型: 键集游标, 允许在记录集中进行所有类型的移动
rs.CursorType = adOpenKeyset
' 设置记录集使用的连接对象为打开的 Connection 对象
Set rs.ActiveConnection = cnn
' 设置记录集的记录源
rs.Source = "SELECT * FROM 学籍"

```

说明: 以上属性的设置必须在记录集关闭状态下方可进行。首次设定记录集的锁定类型、游标类型和连接对象后, 如果不需要改变设置内容, 则不必重复设定。经常发生变化的是记录源(Source 属性)。

4. 打开记录集

用记录集的 Open 方法打开记录集。

' 若记录集处于关闭状态则将其打开

```
If rs.State = adStateClosed Then rs.Open
```

说明: 如果记录集已经打开, 调用 Open 方法将引发错误。

5. 对记录集进行操作

在前面的“数据库记录的操作”一节中已作了较详细的讨论, 在此仅举一例:

```
rs.MoveFirst      ' 指针移到首记录
```

6. ADO 对象的关闭和释放

调用 Close 方法可关闭 Connection 或 Recordset 对象。例如:

```
rs.Close
cnn.Close
```

关闭对象并非将其从内存中删除, 对象关闭后可以更改其属性设置, 然后再次打开。要将对象从内存中完全删除, 可将对象变量设置为 Nothing。例如:

```
Set rs = Nothing
Set cnn = Nothing
```

14.7.3 记录集对象的 Open 方法简介

前面讨论打开记录集时, 事先已对记录集的有关属性进行了设置, 使用的是

没有参数的 Open 方法。若使用带有参数的 Open 方法，可以在打开记录集的同时设置记录集的相关属性。

语法：

记录集.Open [记录源, 活动连接, 游标类型, 锁定类型, 命令类型]

记录集的 Open 方法有五个可选参数，其中前四个参数代表了前面已经讨论过的记录集四个重要属性。

最后一个参数“命令类型”与 ADO 数据控件的 CommandType 属性相似，默认值为 adCmdUnknown（未知命令类型），可以将其设置为 adCmdText（SQL 语句）或 adCmdTable（数据库中表的名称），但必须与“记录源”参数的内容相对应。

设 cnn 为已经与数据库建立连接的 Connection 对象，可以用以下程序段打开记录集：

```
Dim rs As New ADODB.Recordset
Dim strSQL As String
' 用数据库中的表作为记录源
strSQL = "学籍"
rs.Open strSQL, cnn, adOpenKeyset , _
    adLockOptimistic, adCmdTable
...
rs.Close
' 用 SQL 语句作为记录源
strSQL = "SELECT * FROM 学籍 " & _
    " WHERE 性别='男'"
rs.Open strSQL, cnn, adOpenKeyset , _
    adLockOptimistic, adCmdText
```

14.8 数据报表

14.8.1 创建简单报表

创建数据报表时通常需要借助数据环境设计器（Data Environment）为报表提供数据源。下面通过实例说明创建报表的一般步骤。

【例 14.16】预览和打印 Sudent2K.mdb 数据库中的“基本信息”表。

① 新建工程，在窗体上放置两个命令按钮，设其 Caption 属性分别为“报表预览”和“报表打印”。

② 执行【工程】|【更多 ActiveX 设计器】|【Data Environment】菜单命令，在当前工程中添加一个默认名称为 DataEnvironment1 的数据环境对象，系统自动打开如图 14.30 所示的数据环境设计器窗口。

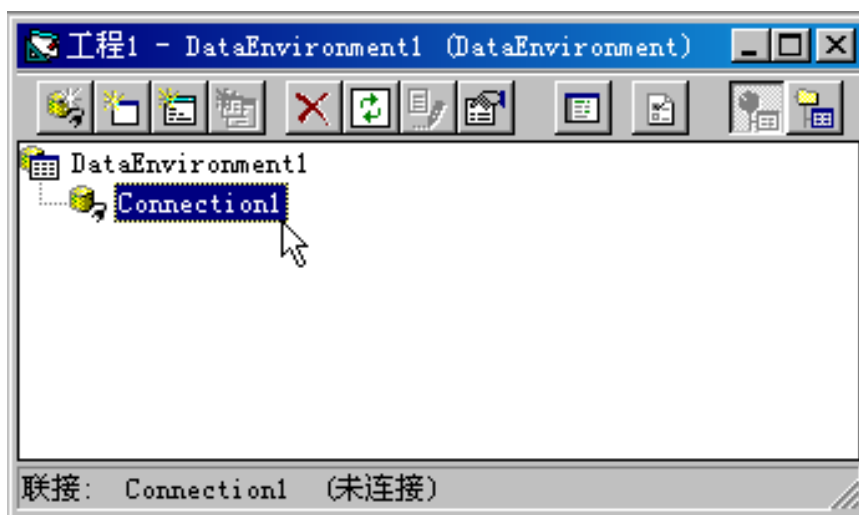


图 14.30 数据环境设计器

右击该窗口中的连接对象 Connection1，选择弹出菜单中的【属性】菜单项，打开如 14.3.1 节中图 14.21 所示的【数据链接属性】对话框。在【提供程序】选项卡的列表中选择“Microsoft Jet 4.0 OLE DB Provider”，单击【下一步】按钮，切换到如 14.3.1 节图 14.22 所示的【连接】选项卡。在【连接】选项卡

中单击【1. 选择或输入数据库名称】输入框右侧的  按钮，在弹出的【连接 Access 数据库】对话框中选择数据库，单击【打开】按钮后返回【连接】选项卡，单击【测试连接】按钮，成功后单击【确定】，完成连接数据库的设置，返回数据环境设计器窗口。

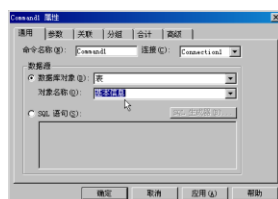


图 14.31 设置命令对象属性

③ 再次右击连接对象 Connection1，选择弹出菜单中的【添加命令】菜单项，在 Connection1 下创建一个默认名称为 Command1 的命令对象。右击该对象，打开如图 14.31 所示的【Command1 属性】对话框。在对话框中选定【数据源】项下的【数据库对象】单选按钮，在【数据库对象】右侧的下拉式列表框中选择【表】，在【对象名称】下拉式列表框中选择【基本信息】，单击【确定】按钮，返回数据环境设计器窗口。在数据环境设计器中展开 Command1 对象，可以看到来自“基本信息”表中的所有字段，如图 14.32 所示。

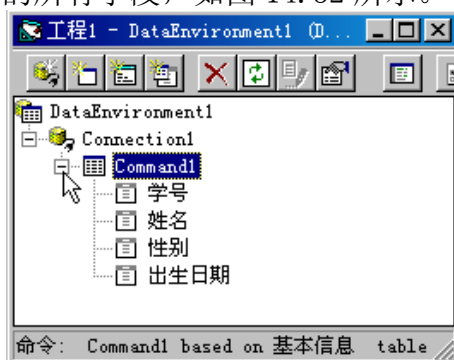


图 14.32 展开命令对象

④ 执行【工程】|【添加 Data Report】菜单命令，在当前工程中添加一个默认名称为 DataReport1 的报表对象，系统自动打开如图 14.33 所示的报表设计器窗口，同时工具箱切换为【数据报表】专用控件。在如图 14.34 所示的【属性】窗口中，将数据报表对象 DataReport1 的 DataSource 属性设为第②步建立的数据环境对象 DataEnvironment1，将 DataMember 属性设为第③步建立的命令对象 Command1。为了便于对齐数据报表中的控件，可将 GridX 和 GridY 属性均设为 5~10 之间的整数。

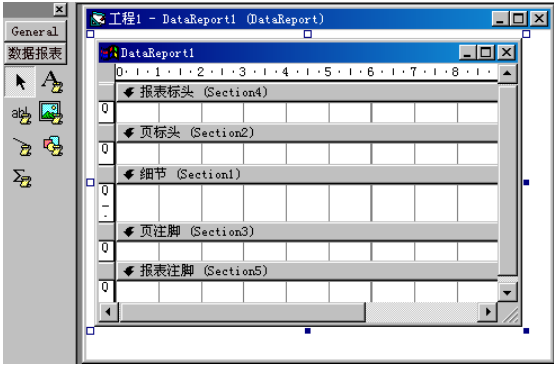


图 14.33 报表设计器及其专用控件



图 14.33 报表设计器及其专用控件

⑤ 将图 14.32 所示的数据环境设计器中 Command1 对象下面的字段（如【学号】）拖放到报表设计器的【细节】区，此时该区将同时添加两个控件（如图 14.35 所示），左侧是用作标题的标签 (RptLabel1)，右侧是用于显示字段数据的文本框 (RptTextBox，含有 “[Command1]” 字样)。将标签控件拖放到【页标头】区，并将标签与对应的文本框对齐。用同样的方法将需要在报表中显示的字段添加到报表设计器中。报表中各记录之间的行距取决于【细节】区的高度，拖动该区的下边界即可调整行距(应尽量使【细节】区的高度接近文本框控件的高度)。

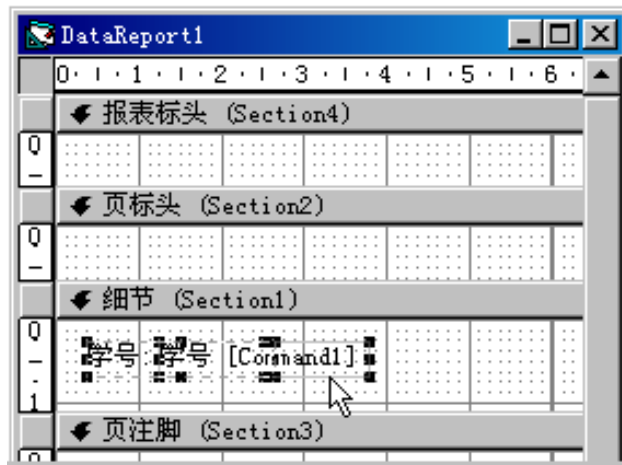


图 14.35 在报表中添加字段

⑥ 选择【数据报表】工具箱中的标签控件 ，在【报表标头】区添加一个标签作为报表的标题，设其 Caption 属性为“学生基本信息表”，通过 Font 属性为其设置字体。在【页标头】区中的字段标题下面用直线(RptLine)控件  画一条横线。设计完成的报表如图 14.36 所示。

⑦ 在工程主窗体【报表预览】按钮的单击事件过程中加入以下代码：

DataReport1.Show

在【报表打印】按钮的单击事件过程中加入以下代码：

' True 参数表示显示打印对话框

DataReport1.PrintReport True



图 14.36 设计报表布局

图 14.37 为程序运行时单击“报表预览”按钮后显示的界面。在该窗口中单

击左上角的“打印”按钮  可以直接打印报表，单击“导出”按钮  可以将报表导出为文本文件或 HTML 格式的文件。



图 14.37 报表打印预览窗口

14.8.2 创建含有分层结构的报表

图 14.38 是一个含有分层结构的报表，该报表显示了每位学生的各科成绩。在数据库应用中经常会处理类似的信息，如每位客户的所有定单，同类商品的不同生产厂家等。下面通过实例说明创建分层结构报表的一般步骤。

【例 14.17】扩展“例 14.16”的功能，创建如图 14.38 所示含有分层结构的报表“学生成绩表”。

① 在主窗体上添加一个命令按钮，设置 Caption 属性为“成绩表预览”。

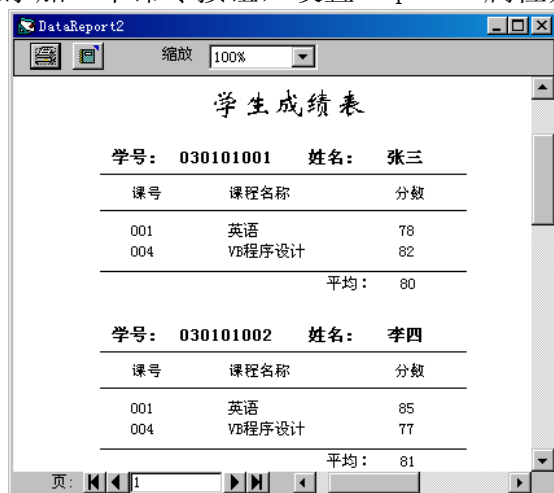


图 14.38 分层结构报表

② 在 14.8.1 节图 14.32 所示的数据环境设计器中右击连接对象 Connection1，在弹出菜单中选择【添加命令】菜单项，在 Connection1 下添加一个默认名称为 Command2 的命令对象，按 14.8.1 节第③步所述为该对象设置数据源。

③ 在数据环境设计器中右击 Command2，在弹出菜单中选择【添加子命令】菜单项，在 Command2 下添加一个默认名称为 Command3 的命令对象。右击该对象，打开如图 14.39 所示的【Command3 属性】对话框。在对话框中选定【数据源】框架中的【SQL 语句】单选按钮，单击【SQL 生成器】按钮，系统将同时打开如图 14.40 所示的【数据视图】窗口和图 14.41 所示的 SQL 生成器窗口。

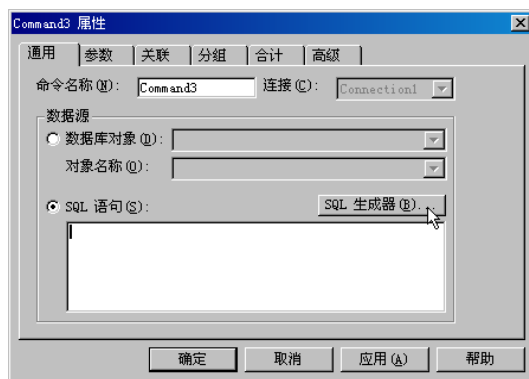


图 14.39 设置命令对象数据源



图 14.40 数据视图

【练习】

6. 使用 VB 内置的“可视化数据管理器”创建 Student.mdb 数据库，在其中建立“基本情况表”，按表 14.2 设置字段并输入记录。
7. 利用 MS Access 创建 Student2k.mdb 数据库，根据表 14.3~14.5 中的数据建立“基本信息”、“成绩”和“课程”三个数据表，并设置表间关联关系。
8. 如何使用 ADO 数据控件连接数据库？如何在设计时和程序运行时设置 ADO 数据控件的记录源？
9. 要使用文本框和 DataGrid 等数据绑定控件自动显示数据库中的数据，应当设置哪些属性？请使用 ADO 数据控件和 DataGrid 控件显示第 1 题“基本情况”表中的记录。
10. 用 SQL 语句查询第 2 题“基本信息”表中 1981 年 8 月 31 日以后出生的所有姓“李”的学生，并将查询结果显示在 DataGrid 控件中。
11. 利用 4 个命令按钮实现记录指针的移动，分别指向首记录、上一记录、下一记录和末记录，并通过 DataGrid 控件观察指针移动效果。
12. 扩展第 6 题的功能，使用命令按钮实现记录的添加、删除和修改。
13. 使用数据环境设计器和数据报表设计器制作如图 14.37 和图 14.38 所示的报表。

第 15 章 综合应用实例

15.1 系统功能总体设计

15.1.1 设计目的

管理信息系统(MIS, Management Information System)是进行信息的采集、存储、加工、维护和使用的系统,在现代信息社会中,它的应用越来越普及。本章通过一个经过简化的 MIS 应用实例“学生信息管理系统”的创建,使读者掌握用 VB 和 ADO 技术编制数据访问应用程序的基本过程和方法,同时巩固和提高对各种常用控件的综合应用能力。

15.1.2 系统功能

本实例采用 VB+ADO+Access,创建一个简单的学生信息管理系统,系统的主要功能如下:

- ◆ 学籍管理:包括基本学籍信息的输入、修改和查询,可随时根据查询结果动态生成学生基本信息查询报表。
 - ◆ 班级管理:包括系、专业和班级的添加和修改。
 - ◆ 课程及成绩管理:包括课程信息的输入和修改;成绩信息的输入、修改和查询,可随时根据查询结果动态生成学生成绩查询报表。
 - ◆ 用户管理:包括添加用户、删除用户、设置用户权限和修改密码。
- 系统功能模块如图 15.1 所示。

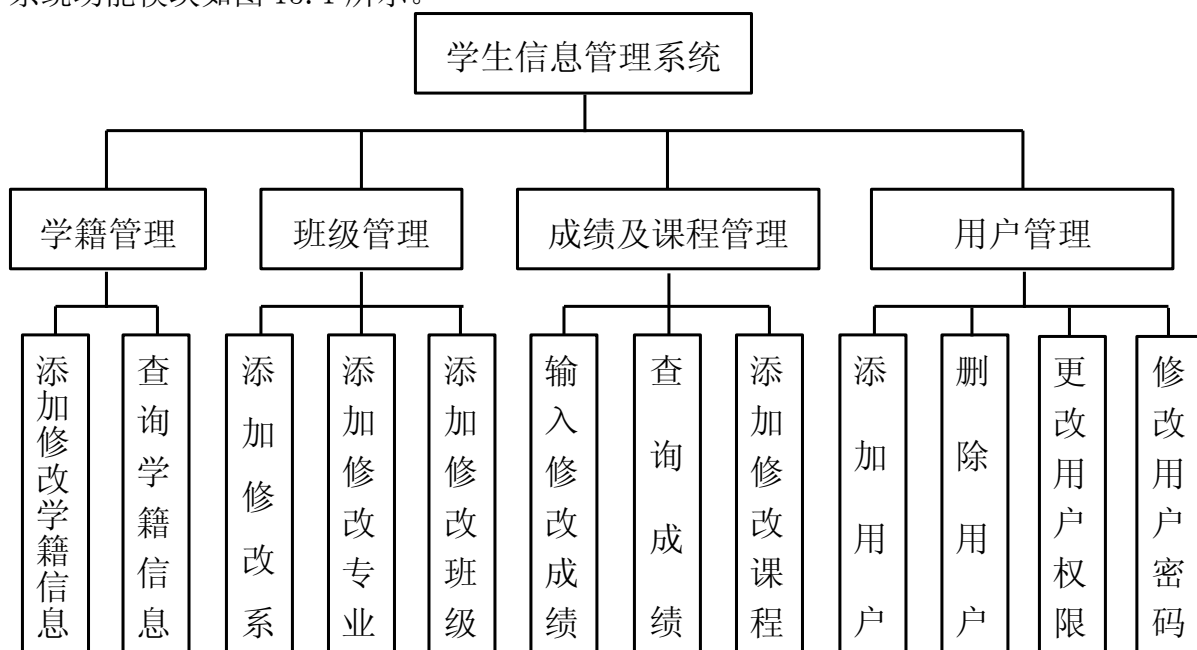


图 15.1 系统功能模块

15.2 数据库设计

15.2.1 建立数据库

利用 Microsoft Access 或 VB 中的“可视化数据管理器”建立数据库，名称为“Student.mdb”。

15.2.2 建立数据表

根据系统功能模块的需要，在 Student.mdb 数据库中建立 8 个表。

1. 学籍表

该表存放学生的基本信息，名称为“学籍”，结构如表 15.1 所示。学号由 11 位数字组成，前 8 位与班号相同，后 3 位为本班学生的序号。

表 15.1 学籍表结构

字段名	类型	大小	说明	字段名	类型	大小	说明
学号	文本 (Text)	11	主键	出生日期	日期 (Date)		
姓名	文本 (Text)	10		班号	文本 (Text)	8	与班级表中的班号关联
性别	文本 (Text)	2					

2. 系表

该表存放各系的编号和名称。表的名称为“系”，结构如表 15.2 所示。系号为两位数字，必须惟一。

表 15.2 系表结构

字段名	类型	大小	说明	字段名	类型	大小	说明
系号	文本 (Text)	2	主键	系名	文本 (Text)	20	

3. 专业表

该表存放各系中的专业编号和专业名称。表的名称为“专业”，结构如表 15.3 所示。专业号由 4 位数字组成，前 2 位与系号相同，后 2 位为本系中的专业序号。

表 15.3 专业表结构

字段名	类型	大小	说明	字段名	类型	大小	说明
专业号	文本 (Text)	4	主键	系号	文本 (Text)	2	与系表中的系号关联
专业名称	文本 (Text)	20					

4. 班级表

该表存放各专业中的班级编号和班级名称。表的名称为“班级”，结构如表

15.4 所示。班号由 8 位数字组成，前 2 位为年级，第 3~6 位与专业号相同，最后 2 位为本专业中的班级序号。

表 15.4 班级表结构

字段名	类型	大小	说明	字段名	类型	大小	说明
班号	文本 (Text)	8	主键	专业号	文本 (Text)	2	与专业表中的专业号关联
班级名称	文本 (Text)	20					

5. 成绩表

该表存放学生成绩，名称为“成绩”，结构如表 15.5 所示。为减少数据冗余，成绩表中仅存储学号(与学籍表学号字段关联)，不存储学生姓名，需要时根据学号从学籍表中获取姓名。课号与课程名称亦作同样处理。

表 15.5 成绩表结构

字段名	类型	大小	字段名	类型	大小
学号	文本 (Text)	11	分数	整型 (Integer)	
课号	文本 (Text)	3			

6. 课程信息表

该表存放课程信息，名称为“课程”，结构如表 15.6 所示。

表 15.6 课程信息表结构

字段名	类型	大小	说明
课号	文本 (Text)	3	主键
课程名称	文本 (Text)	20	

上述 6 个表之间具有一定的关联，为了保证数据参照完整性，应当建立表间关联关系并设置参照完整性。各表之间的关系如图 15.2 所示。

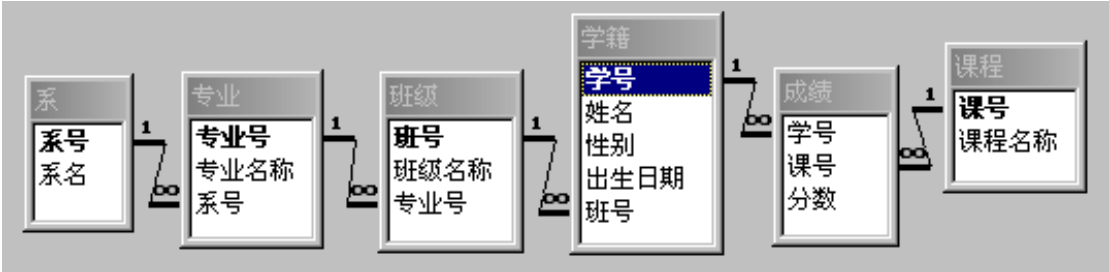


图 15.2 表间关系

7. 用户表

该表存放用户登录信息，名称为“用户”，结构如表 15.7 所示。表中暂时存放两条记录，内容如表 15.8 所示。

表 15.7 用户表结构

表 15.8 用户表内容

字段名	类型	大小	说明
用户名	文本 (Text)	16	主键
密码	文本 (Text)	16	
权限	文本 (Text)	10	

用户名	密码	权限
Admin	123456	管理员
User	123	普通

8. 临时表

该表作为临时工作表，名称为“临时”，用于输入成绩，结构如表 15.9 所示。在数据库中设计“临时”表的目的是兼顾 DataGrid 控件的使用和减少数据冗余。

表 15.9 临时表结构

字段名	类型	大小	字段名	类型	大小
学号	文本 (Text)	20	分数	整型 (Integer)	
姓名	文本 (Text)	10	课号	文本 (Text)	3

15.3 用户登录及主窗体设计

15.3.1 创建工程

创建一个 VB 工程，名称为 StudInfo.vbp，将该工程与 15.2 节中创建的数据库 Student.mdb 保存在同一文件夹中。

15.3.2 设计用户登录窗体

本窗体 (frmLogin) 作为系统的启动窗体，用于验证用户是否合法，运行时界面如图 15.3 所示。

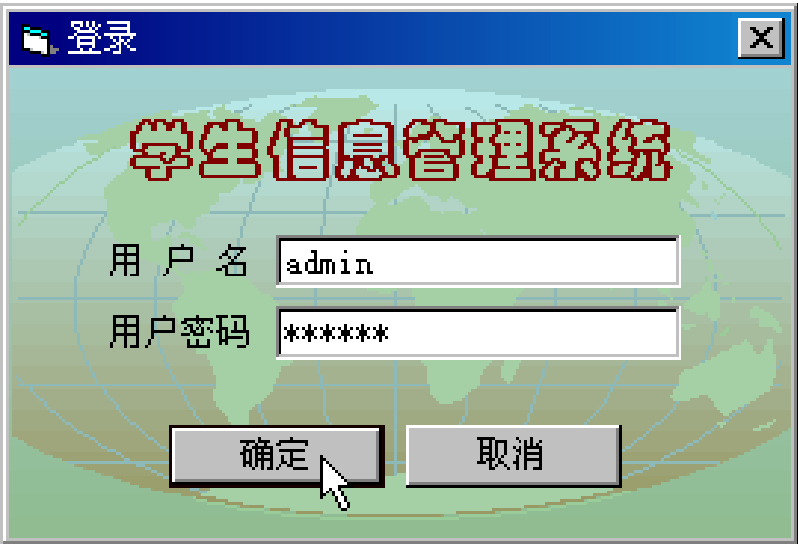


图 15.3 用户登录

窗体上的两个文本框分别用于输入用户名和密码，设密码文本框 PasswordChar=*. 添加一个 ADO 数据控件，设 Visible=False，使用连接字符串与 Student.mdb 数据库连接。

单击“确定”按钮后，用 SQL 语句查询“用户”表中是否有相符的用户名和密码，若不符，提示重新输入，焦点返回文本框。如果 3 次输入错误，退出系统。若输入正确，将用户名和用户权限保存在全局变量中，显示系统主窗体，卸

载本窗体。单击“取消”按钮时结束程序运行。

15.3.3 设计系统主窗体

系统主窗体(frmMain)作为学生信息管理系统的主界面，如图 15.4 所示。窗体背景采用 Image 控件(Stretch=True，BorderStyle=1)显示图片。



图 15.4 主窗体

菜单结构如表 15.10 所示。

表 15.10 菜单结构

系统	学籍管理	班级管理	成绩与课程管理	帮助
添加用户	添加或修改学籍信息	添加或修改系	输入或修改成绩	关于
删除用户	查询学籍信息	添加或修改专业	查询成绩	
更改权限		添加或修改班级	添加或修改课程	
修改密码				
退出系统				

单击某一菜单项时，显示对应窗体。“系统”菜单中的“添加用户”、“删除用户”和“更改权限”三个菜单项的功能仅供权限为“管理员”的用户使用。因此，应在窗体加载时根据保存在全局变量中的用户权限确定是否显示这三个菜单项。

15.4 功能模块设计

15.4.1 学籍管理模块

学籍管理模块的功能通过【学籍管理】窗体(frmEss)实现。该窗体为主窗体【学籍管理】菜单下的两个菜单项【添加或修改学籍信息】和【查询学籍信息】所共用，通过选项卡区分其功能。选择【添加或修改学籍信息】菜单项时，将【添加或修改】选项卡设为活动选项卡；选择【查询学籍信息】菜单项时，将【查询】选项卡设为活动选项卡。

1. 使用 TreeView 控件和 SSTab 控件

为了便于用户操作，学籍管理窗体界面设计中采用 TreeView 控件供用

户选择系、专业和班级，用 SSTab 控件切换管理功能，界面效果如图 15.5 所示。

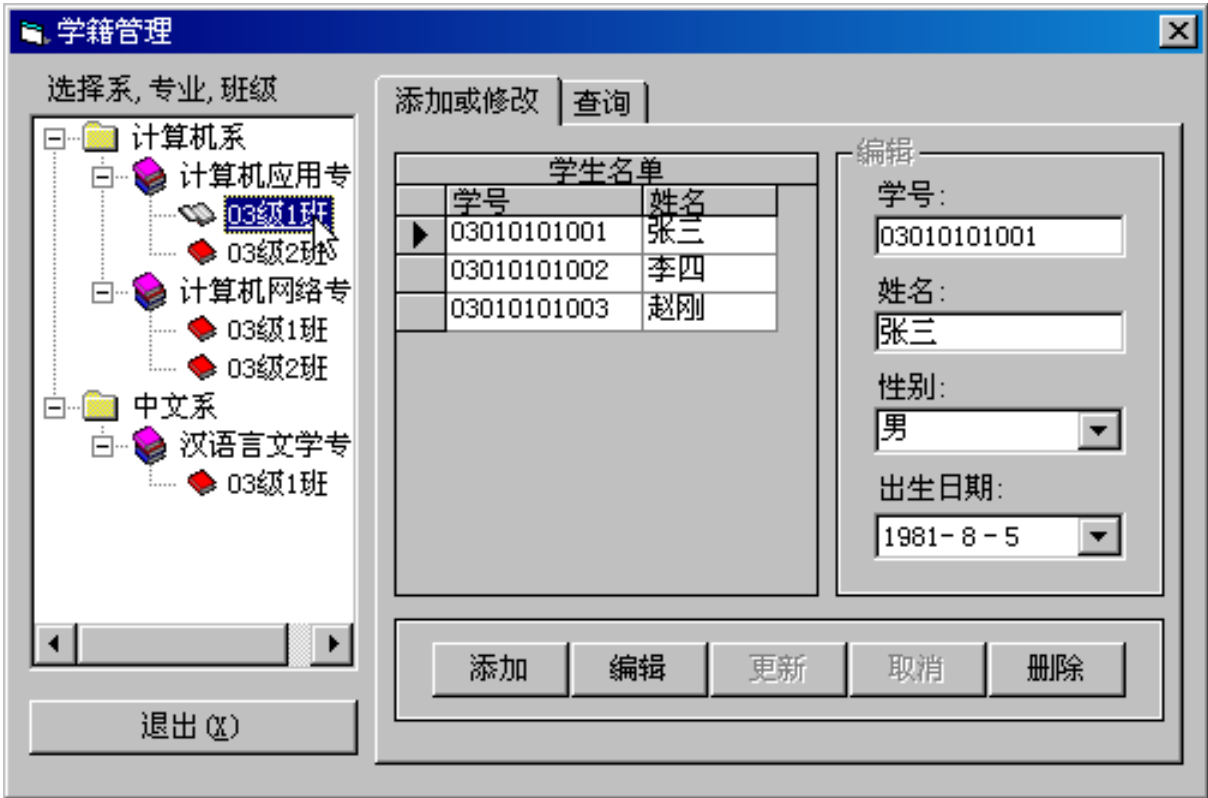


图 15.5 浏览状态

TreeView 控件相关属性的设置见第 11 章 11.2.1 小节,注意将 LabelEdit(标签编辑)属性设为 1-tvwManual,即不允许用户修改节点标签。由于成绩管理窗体亦采用类似的界面设计,因此在标准模块中将 TreeView 控件添加系、专业和班级节点的程序段编制成全局过程 MakeTree(tvwX As TreeView)供各窗体调用。该过程以当前窗体的 TreeView 控件作为参数,依次从数据库的系表、专业表和班级表中获取相关数据生成系-专业-班级目录树。

具体做法是首先利用记录集对象读取系表中的全部记录,将其添加为 TreeView 控件中的顶层节点,节点关键字(Key)为“X”+系号(关键字必须以字母开头),节点文本(Text)为系名,节点标志(Tag)均为“系”。

然后读取专业表中的记录,根据每条记录的系号字段值和 TreeView 中系节点的关键字,将其添加为对应系的子节点,节点关键字为“X”+专业号,文本为专业名称,标志均为“专业”。

最后读取班级表中的记录,根据每条记录的专业号字段值和专业节点关键字,将其添加为对应专业的子节点,节点关键字为“X”+班号,文本为班级名称,标志均为“班级”。

SSTab 控件用于切换学籍管理功能,将其选项卡数设为 2,标题分别为“添加或修改”和“查询”。该控件的属性设置方法见第 11 章 11.3.1 小节。

2. 添加或修改学籍信息

添加或修改学籍信息的功能在【添加或修改】选项卡中实现。程序运行时,在 TreeView 控件中选择系、专业和班级,若当前节点为班级,则在 DataGrid 中显示本班学生名单,此时可通过按钮进行添加、编辑、删除等操作,界面如图 15.5 和图 15.6 所示。

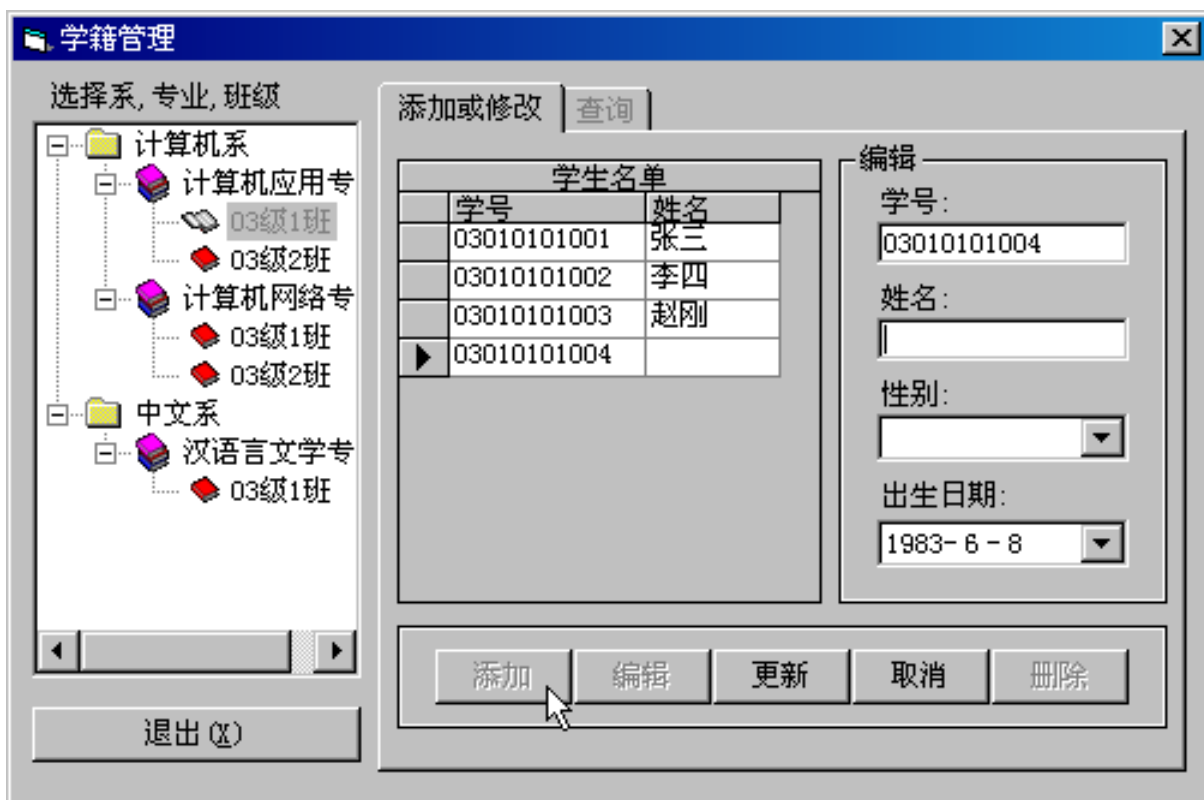


图 15.6 添加或修改状态

(1) 设置 ADO 数据控件

在窗体上添加一个 ADO 数据控件，名称为 adoEdit，设 Visible=False, LockType=4(批更新)。将其与数据库连接，设记录源为 SQL 语句：
SELECT * FROM 学籍 WHERE 学号=''

在上述语句中，等号后面是一对单引号，表示空字符串。由于在设计学籍表时已将学号字段设为主键，且不允许为空字符串，因此表中不会有学号为空的记录。该语句的作用是生成含有各字段结构的空记录集，以供 DataGrid 控件检索字段使用。程序运行时由 adoEdit 为各数据绑定控件提供数据源。

(2) 设置 DataGrid 控件

在【添加或修改】选项卡中放置一个 DataGrid 控件，名称为 DataGrid1，用于显示学号和姓名以及移动指针选择记录。设置其 DataSource 属性为 adoEdit, AllowUpdate 属性为 False(不允许修改), Caption 属性为“学生名单”。右击该控件，在快捷菜单中选择【检索字段】菜单项，在对话框中单击【是】。再次右击该控件，选择【属性】菜单项，在如图 15.7 所示的【属性页】对话框中单击【布局】选项卡，选择【列】组合框中的“性别”，清空【可见】复选框。对“出生日期”和“班号”作同样处理，目的是将其隐藏。

3) 设置其他控件

在 DataGrid 控件右侧放置一个框架(Frame)，名称为 fraEss, Caption 为“编辑”。框架中放置两个文本框，名称分别为 txtID 和 txtName，用于输入学号和姓名；一个组合框，名称为 cboSex，用于选择性别。将这三个控件的 DataSource 属性均设为 adoEdit, DataField 属性分别设为学号、姓名和性别字段。再添加一个 DateTimePicker 控件 DTPicker1，用于显示和设定出生日期。采用这一控件的优点是不必编写验证代码即可保证用户的输入是合法的日期型数据，该控件属性的设置见 11.3.3 小节。

为了避免出生日期字段为 NULL 值时导致程序出错, 对该字段的显示和赋值不采用绑定方式, 而是通过代码完成相关操作。

记录的添加、编辑、更新、取消和删除操作通过 5 个命令按钮实现。

窗体加载时, 设置【编辑】框架(fraEss)的 Enabled=False(可使框架中的控件无效)。将【更新】和【取消】按钮设置为无效, 其他按钮有效, 此时为浏览状态(图 15.5)。单击【添加】或【编辑】按钮时, 将上述控件的 Enabled 属性取反, 此时为编辑状态(图 15.6)。

3. 查询学籍信息

查询学籍信息的功能在【查询】选项卡中实现。程序运行时, 在 TreeView 控件中选择查询范围, 在【查询方式】框架中设置查询条件。单击【查询】按钮后, 显示查询结果, 若有记录则使【报表】按钮有效, 可随时打印查询结果。界面如图 15.8 和图 15.9 所示。

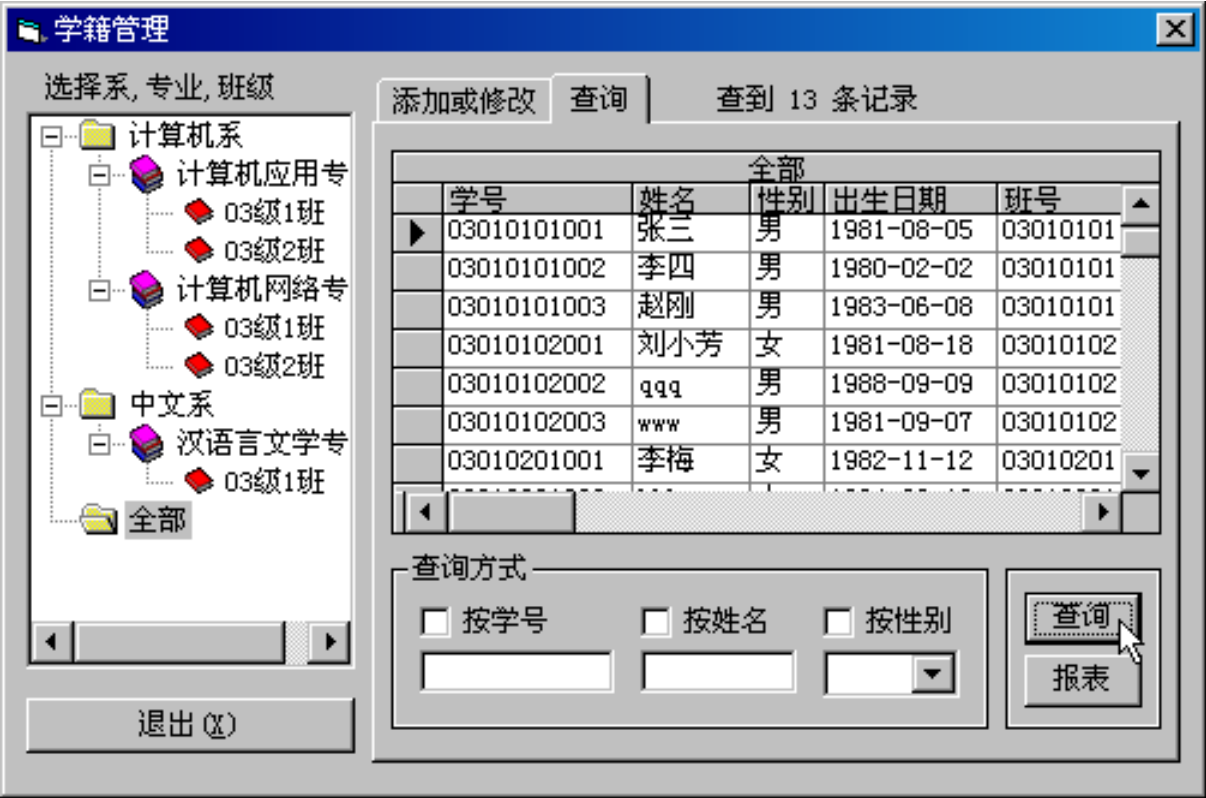


图 15.8 全部显示

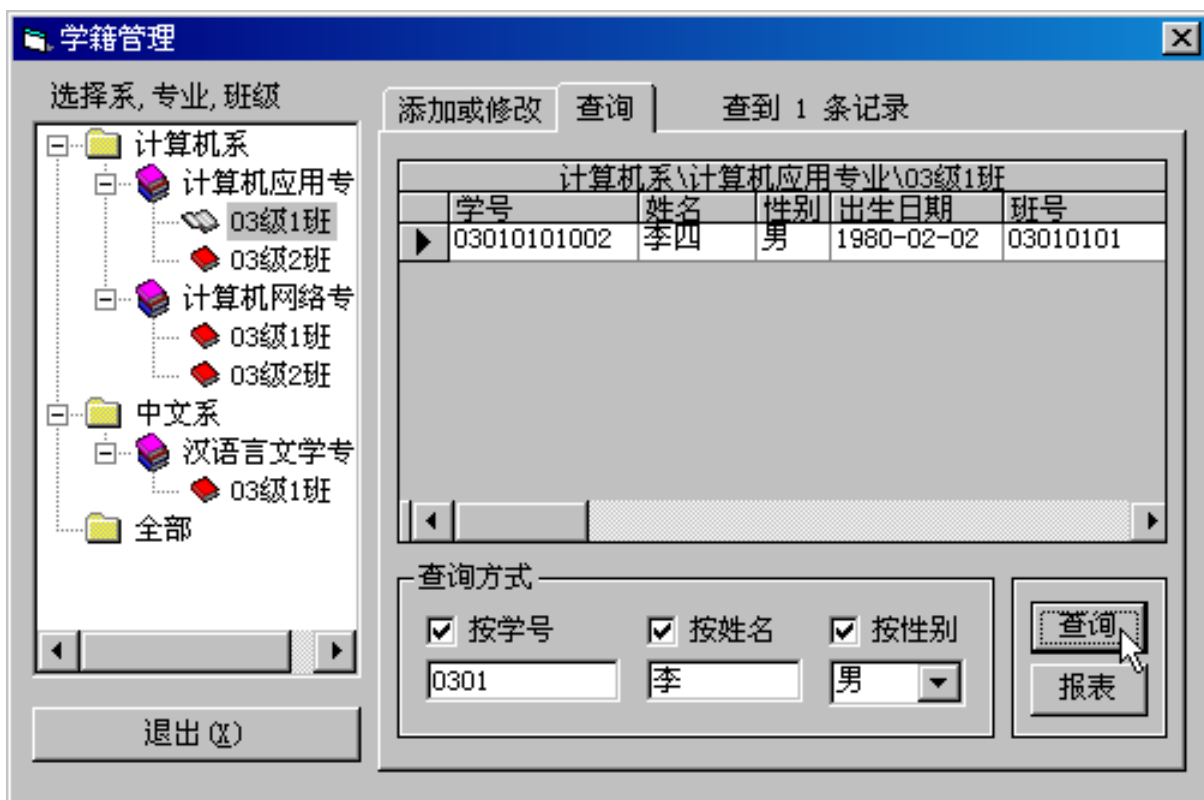


图 15.9 多条件复合查询

1) 设置 ADO 数据控件和 DataGrid 控件

在窗体上添加一个 ADO 数据控件, 名称为 adoQuery, 设 Visible=False, 将其与数据库连接, 记录源的设置与 adoEdit 相同。在【查询】选项卡中添加一个 DataGrid 控件, 名称为 DataGrid2, 用于显示查询结果。设 DataSource=adoQuery, AllowUpdate=False, Caption 为“查询结果”。右击该控件检索字段, 将全部字段名显示在列标头中。

(2) 设置查询方式控件

在【查询方式】框架中添加 3 个复选框, 名称分别为 chkNo、chkName 和 chkSex, 设 Caption 分别为“按学号”、“按姓名”和“按性别”。添加两个文本框, 名称分别为 txtNoQ 和 txtNameQ, 用于输入学号和姓名。添加一个组合框, 名称为 cboSexQ, 预设两个列表项“男”、“女”, 用于选择性别。

3) 设置查询和报表按钮

查询按钮的名称为 cmdQuery。在查询按钮的单击事件中, 首先根据 TreeView 中的当前节点确定查询范围, 将其显示在 DataGrid2 的标题中, 然后根据各复选框的选中状态判断查询条件, 若有复选框被选中, 则根据文本框和组合框中的内容, 用 SQL 语句的模糊查询、多条件复合查询功能生成查询语句, 为 adoQuery 的 RecordSource 属性赋值。

报表按钮的名称为 cmdReport。

为了实现“即查即览”功能, 须将数据环境中为报表提供数据的命令对象的数据源设为 SQL 语句:

```
SELECT * FROM 学籍
```

数据环境中的命令对象有一个对应的记录集对象, 名称为“rs”+命令对象名。在报表按钮的单击事件中, 将 adoQuery 的 RecordSource 属性赋值给报表

数据源记录集对象的 Source 属性，调用记录集对象的 Requery 方法重新执行查询即可动态改变报表数据源。

查询结果的显示和报表预览如图 15.10 和图 15.11 所示。

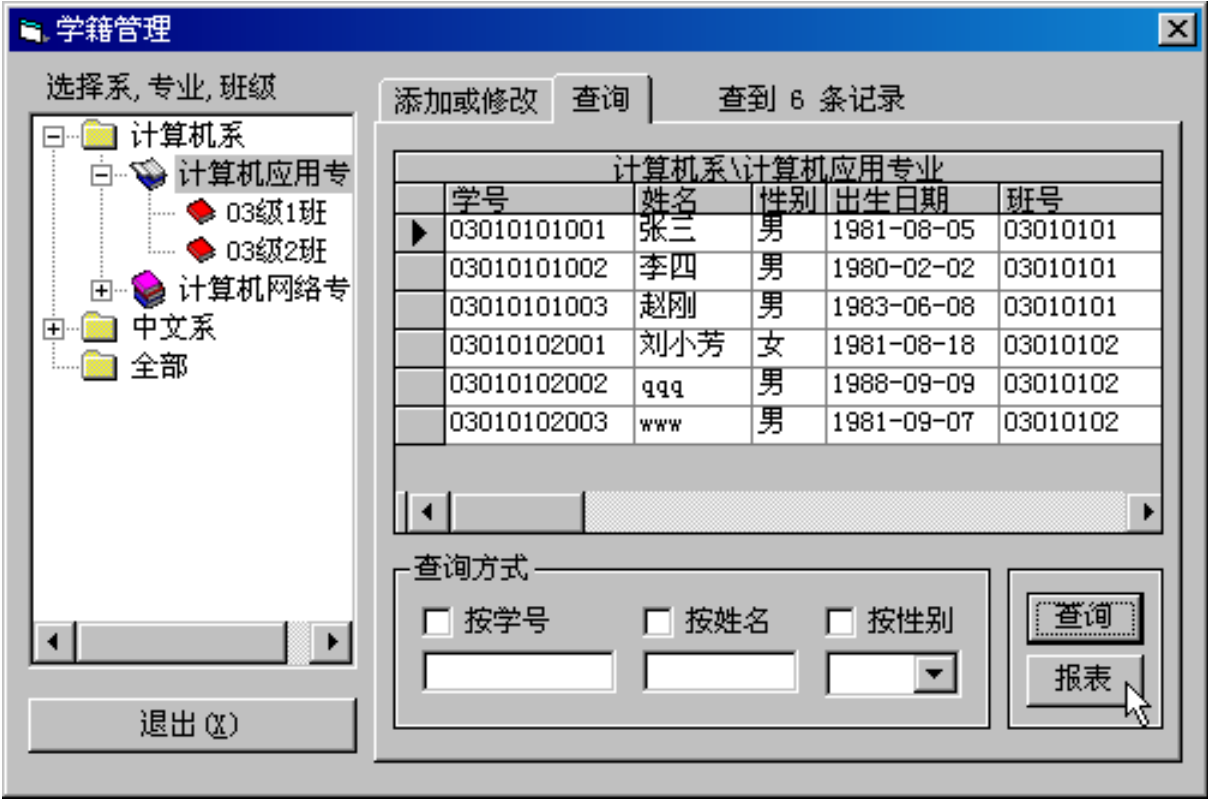


图 15.10 显示查询结果



图 15.11 报表打印预览

15.4.2 班级管理模块

班级管理模块的功能通过【班级管理】窗体 (frmClass) 实现。该窗体为主窗体【班级管理】菜单下的 3 个菜单项【添加或修改系】、【添加或修改专业】和【添加或修改班级】所共用，通过选项卡区分其功能。菜单项与活动选项卡的关系跟学籍管理模块相似。界面如图 15.12、图 15.13 和图 15.14 所示。各选项卡的左半部分用于导航，右半部分用于添加或编辑数据。

班级管理

添加或修改系 | 添加或修改专业 | 添加或修改班级

现有系	
系号	系名
01	计算机系
02	中文系

添加或修改

系号: 01

系名: 计算机系

说明

系号为2位数字, 不得有重号
系名不得与现有系重名

添加 编辑 更新 取消 删除 退出

图 15.12 添加或修改系

班级管理

添加或修改系 | 添加或修改专业 | 添加或修改班级

选择系: 计算机系

本系现有专业	
专业号	专业名称
0101	计算机应用
0102	计算机网络

添加或修改

专业号: 0101

专业名称: 计算机应用

说明

本系系号: 01
专业号 = 系号 + 专业序号 (2位)
专业号不得有重号

添加 编辑 更新 取消 删除 退出

图 15.13 添加或修改专业

图 15.14 添加或修改班级

1. 添加或修改系

【添加或修改系】选项卡的界面设计如图 15.12 所示。在选项卡中添加一个 ADO 数据控件, 名称为 adoDept, Visible=False, 记录源为 SQL 语句“SELECT * FROM 系”。

DataGrid 控件的名称为 dgdDept, Caption 为“现有系”, AllowUpdate=False, DataSource=adoDept。【添加或修改】框架中两个文本框的名称分别为 txtDeptNo 和 txtDeptName, 设 DataSource=adoDept, 分别与系号和系名字段绑定。本窗体的 3 个选项卡共用一组数据操作按钮, 在它们的单击事件中根据选项卡的 Tab 属性判断当前选项卡, 完成相关记录的添加、编辑和删除等操作。

2. 添加或修改专业

【添加或修改专业】选项卡的界面设计如图 15.13 所示。在选项卡中添加一个 ADO 数据控件, 名称为 adoSpecial, Visible=False, 记录源为 SQL 语句“SELECT * FROM 专业 WHERE 专业号='’”。

DataGrid 控件的名称为 dgdSpcl, Caption 为“本系现有专业”, AllowUpdate=False, DataSource=adoSpecial。【添加或修改】框架中两个文本框的名称分别为 txtSpclNo 和 txtSpclName, 设 DataSource=adoSpecial, 分别与专业号和专业名称字段绑定。【选择系】组合框中的系名列表通过调用标准模块中的 AddDeptItem(cboX As ComboBox) 过程填充。

3. 添加或修改班级

【添加或修改班级】选项卡的界面设计如图 15.14 所示。在选项卡中添加一个 ADO 数据控件, 名称为 adoClass, Visible=False, 记录源为 SQL 语句“SELECT * FROM 班级 WHERE 班号='’”。DataGrid 控件的名称为 dgdClass, Caption 为“本专业现有班级”, AllowUpdate=False, DataSource=adoClass。【添加或修改】框架中两个文本框的名称分别为 txtClassNo 和 txtClassName, 设 DataSource=adoClass, 分别与班号和班级名称字段绑定。

【选择系】组合框的处理与专业选项卡相同。在该组合框中选择系后调用标准模块中的 AddSpecItem(cboX As ComboBox, sDeptNo As String) 过程填充【选择专业】组合框中的专业名称列表。窗体加载时用循环语句填充【入学年份】和【班级序号】组合框，供自动生成班号使用。

15.4.3 成绩及课程管理模块

主窗体【成绩及课程管理】菜单下有 3 个菜单项：【输入或修改成绩】、【查询成绩】和【添加或修改课程】。其中前两项功能共用【成绩管理】窗体(frmGrade)，通过选项卡区分其功能。第三项功能通过【课程管理】窗体(frmCourse)实现。

1. 输入或修改成绩

【输入或修改成绩】选项卡的界面如图 15.15 和图 15.16 所示。程序运行时，在 TreeView 控件中选择班级，在组合框中选择课程，在 DataGrid 中输入或修改成绩。

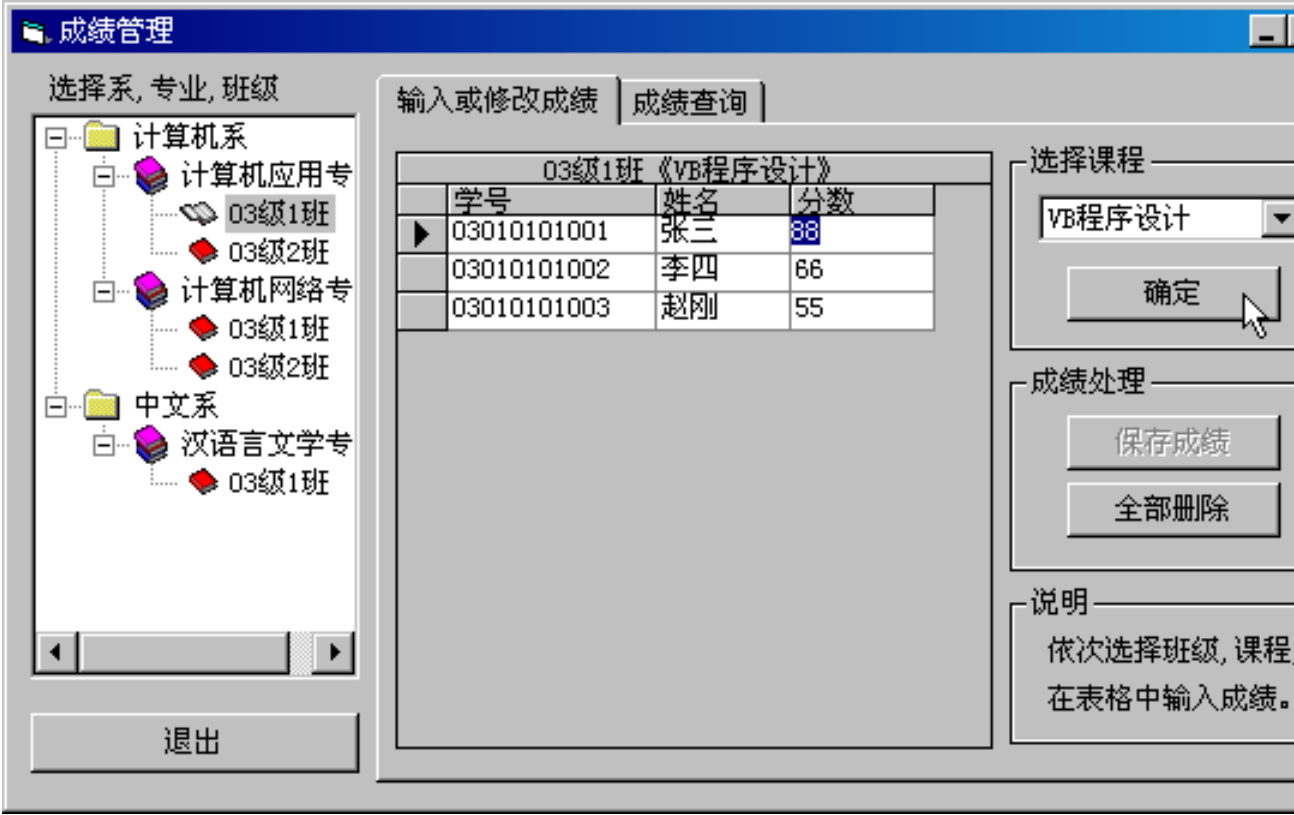


图 15.15 输入或修改成绩



图 15.16 保存成绩

在教学管理工作中，通常按班级和课程成批录入学生成绩。为了便于用户操作，避免在鼠标和键盘操作之间频繁切换，本例采用 DataGrid 控件输入成绩。用户输入成绩并按回车键确认后，即可通过上下箭头键移动记录指针继续录入。

(1) 设置 ADO 数据控件和 DataGrid 控件

在窗体上添加两个 ADO 数据控件，名称分别为 adoInGrid 和 adoGrade，设 Visible = False，LockType = 4（批更新模式）。设 adoInGrid 的记录源为 SQL 语句“SELECT * FROM 临时 WHERE 学号='’”，adoGrade 的记录源为空（运行时与成绩表链接）。

DataGrid 控件的名称为 DataGrid1，AllowUpdate=True，DataSource=adoInGrid，检索字段后在属性页中将课号字段隐藏，将学号和姓名字段锁定，将分数字段的格式设为数字。

(2) 设置其他控件

【选择课程】组合框名称为 cboGrade，窗体加载时调用 AddCourseItem(cboX As ComboBox) 过程（在标准模块中）填充课程列表。【确定】、【保存成绩】和【全部删除】按钮的名称分别为 cmdOK、cmdSave 和 cmdDelete。

2. 查询成绩

查询成绩的界面设计和功能实现均与 15.4.1 小节“查询学籍信息”相似，其中用于获取记录集的 ADO 数据控件名称为 adoQuery，记录源为 SQL 语句：

SELECT 成绩.学号, 学籍.姓名, 课程.课程名称, 成绩.分数, 学籍.班号 FROM 成绩, 学籍, 课程 WHERE 成绩.学号='’

界面如图 15.17 和图 15.18 所示。



图 15.17 显示查询结果



图 15.18 报表打印预览

3. 添加或修改课程

【课程管理】窗体 (frmCourse) 的运行界面如图 15.19 和图 15.20 所示。在窗体上添加一个 ADO 数据控件, 名称为 adoEdit, 设 Visible=False, CommandType=2-adCmdTable, RecordSource=课程, LockType=4 (批更新)。

DataGrid 控件名称为 DataGrid1，设 DataSource 为 adoEdit 并检索字段。用于输入课号和课程名称的文本框名称分别为 txtCourseNo 和 txtCourseName。5 个数据操作按钮的设置与学籍管理窗体相似。

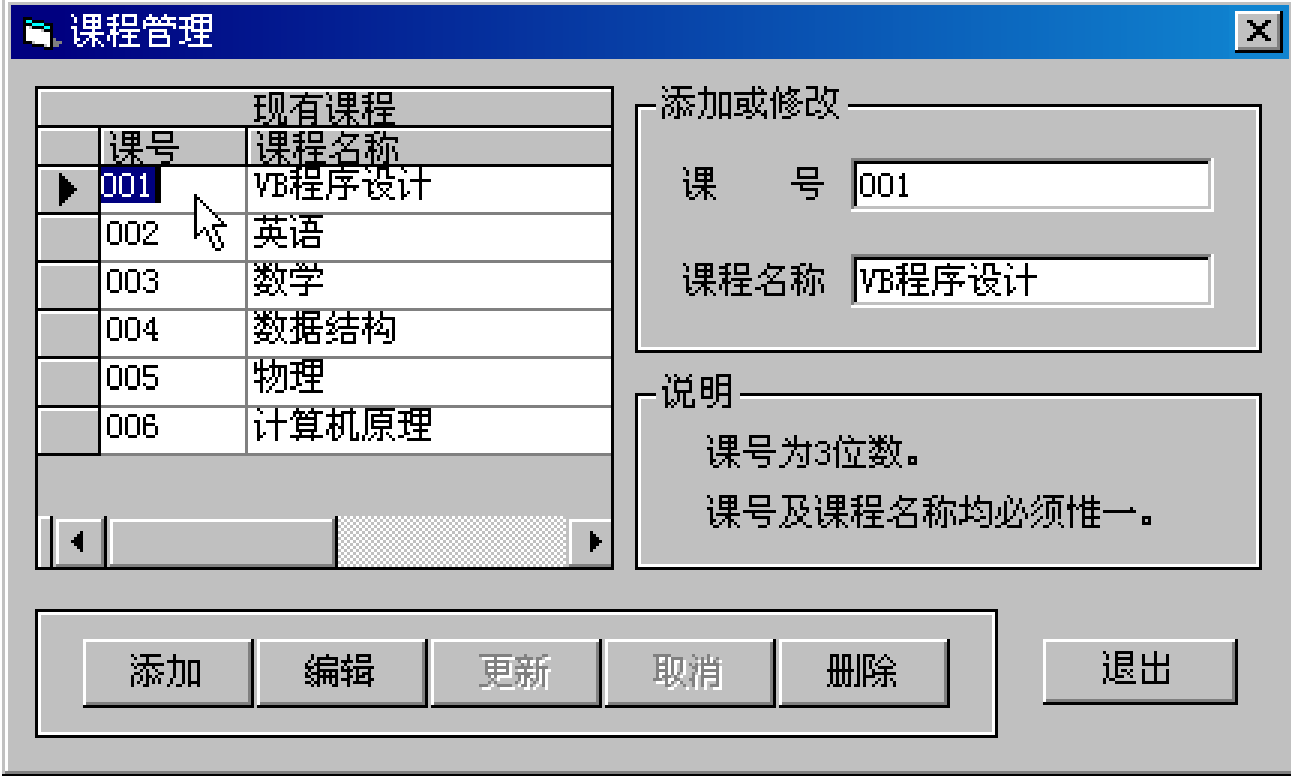


图 15.19 浏览状态

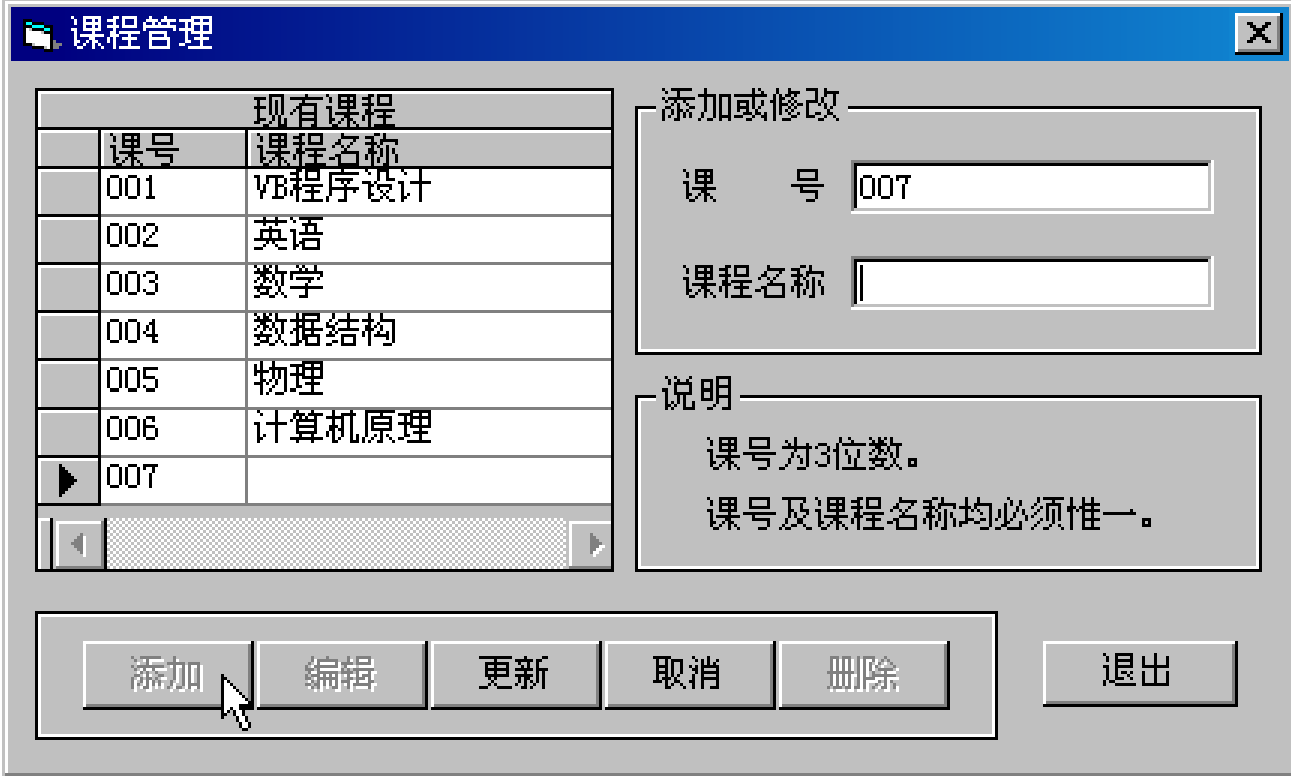


图 15.20 添加或修改状态

15.4.4 系统模块

【系统】菜单下有 5 个菜单项：添加用户、删除用户、更改权限、修改密码和退出。

1. 添加用户

【添加用户】窗体（frmUser）的运行界面如图 15.21 所示。



图 15.21 添加用户

在窗体上添加一个 ADO 数据控件，名称为 adoUser，设 Visible=False，记录源为空。3 个文本框用于输入用户名和密码，名称分别为 txtUserName、txtPassword1 和 txtPassword2，设输入和确认密码的文本框的 PasswordChar 属性为“*”。单击“确认”按钮后，通过 adoUser 用 SQL 语句查询数据库“用户”表中是否有相同的用户名和密码，若有，提示该用户已存在，重新输入，焦点返回用户名文本框。如果无同名用户，将用户名和密码添加到数据库“用户”表中，并设默认权限为“普通”，用 MsgBox 语句提示添加用户成功。

2. 删除用户

【删除用户】窗体（frmDelUser）的运行界面如图 15.22 所示。在窗体上添加一个 ADO 数据控件，名称为 adoUser，设 Visible=False，LockType=4，记录源为空（运行时设为 SQL 语句）。DataGrid 控件的名称为 dgdUser，设 AllowUpdate=False，DataSource=adoUser。在窗体上添加一个框架，名称为 fraDel，内含三个命令按钮，名称分别为 cmdDelete、cmdCancel 和 cmdExit，Caption 分别为“删除用户”、“取消删除”和“退出”。

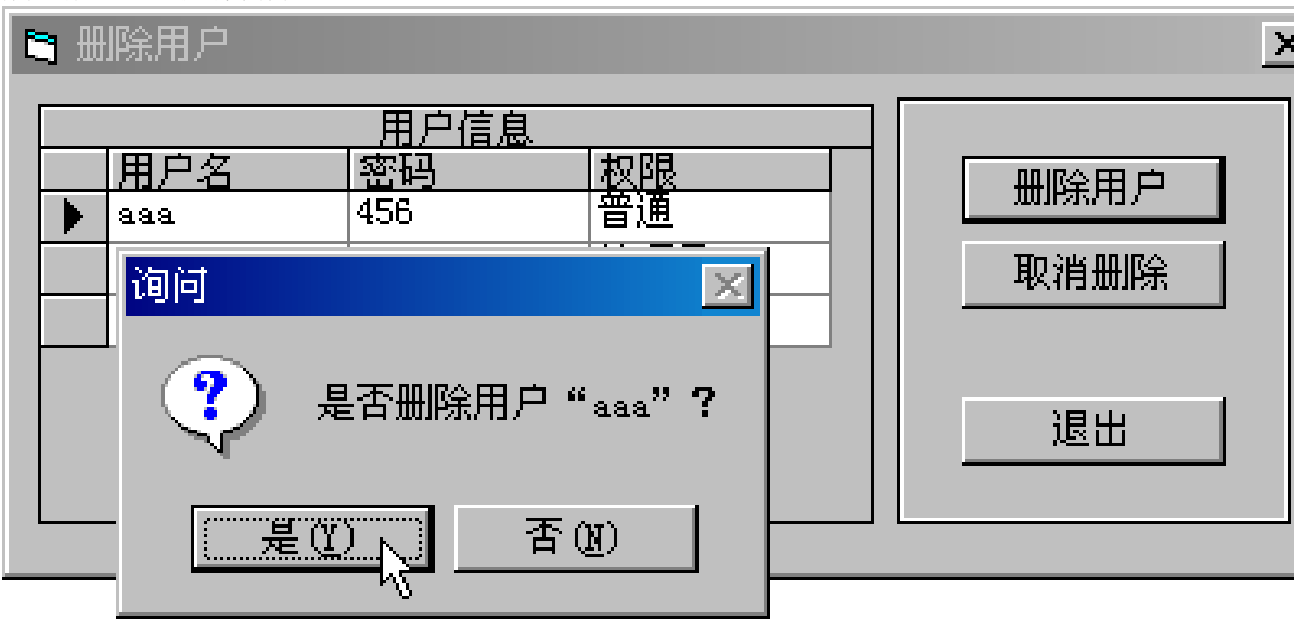


图 15.22 删除用户

3. 更改权限

【更改权限】与【删除用户】的窗体界面相似, 为了减少程序中的窗体数目, 复用代码, 将两项功能设计为共用一个窗体 (frmDelUser)。设计时和运行时界面如图 15.23 和图 15.24 所示。在“删除用户”窗体上添加一个框架, 名称为 fraModi, 设置其宽度和高度与框架 fraDel 相同。在框架中添加一个用于选择权限的组合框 (含有“普通”和“管理员”两个列表项) 和两个命令按钮。



图 15.23 设计时界面

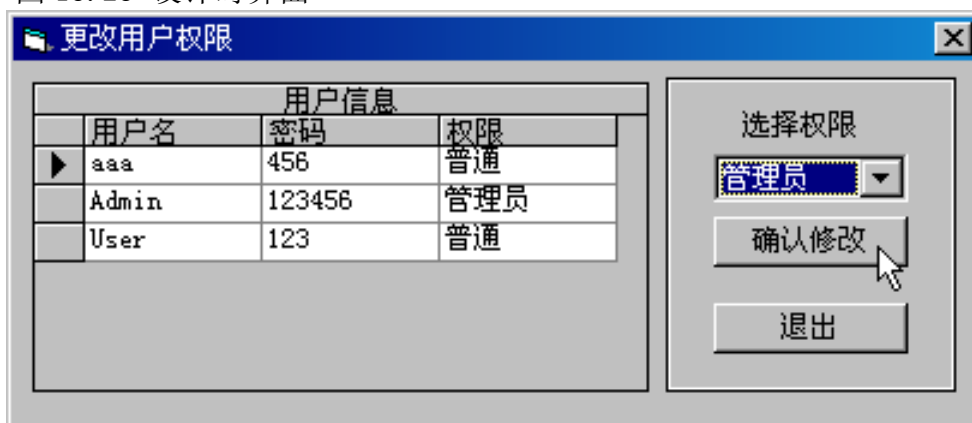
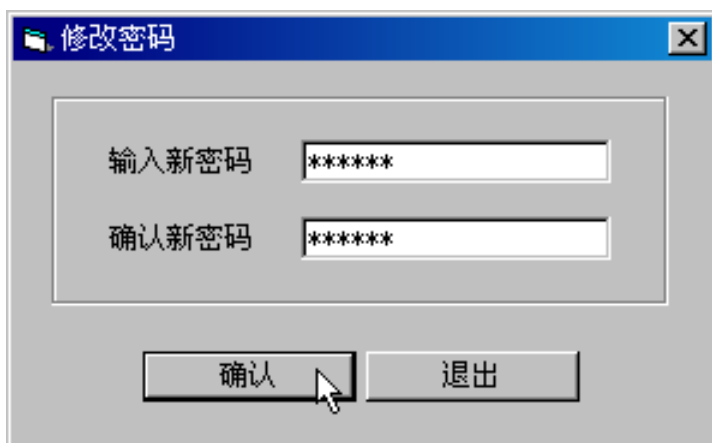


图 15.24 运行时界面

窗体加载时使两个框架的位置重合。在主窗体单击“删除用户”或“更改权限”菜单项时, 根据被选中的菜单项修改 frmDelUser 窗体的标题。在 frmDelUser 窗体的激活事件 (Activate) 中, 根据窗体标题, 显示对应的框架, 隐藏另一个框架。单击“确认修改”按钮后, 将组合框中被选中的权限赋予用户表中当前记录的“权限”字段。

4. 修改密码

【修改密码】窗体 (frmModiPass) 的运行界面如图 15.25 所示。



在窗体上添加一个 ADO 数据控件，名称为 Adodc1，设 Visible=False，记录源为空（运行时设为 SQL 语句）。单击“确认”按钮后，根据保存在全局变量中的用户名查询用户表，将当前用户的新密码存入“用户”数据库。

15.4.5 帮助菜单

主窗体【帮助】菜单下仅含一个菜单项【关于】，在其单击事件中用 MsgBox 语句显示本程序的版本信息，如图 15.26 所示。

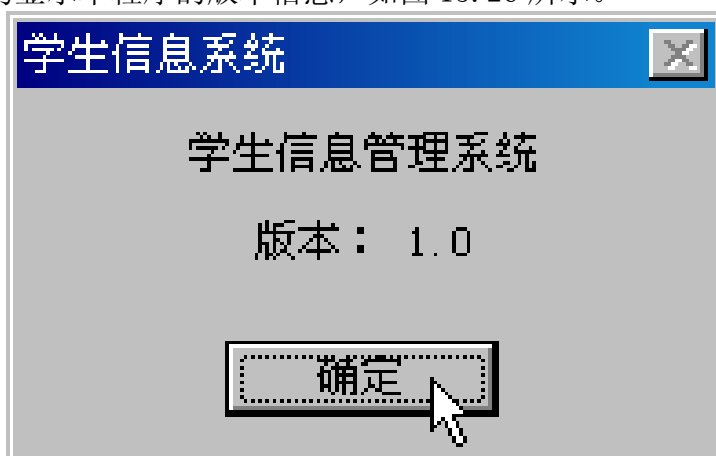


图 15.26 版本信息

15.4.6 关于标准模块

在标准模块中主要是声明一些供各模块使用的全局变量（如用于保存用户名和用户权限的变量）。对于各模块经常用到的一些程序段，亦可将其编制成子过程或函数，用 Public 关键字在标准模块中声明为公用过程，实现代码的复用。

本例中主要的公用过程如下：

CreateConnection()：建立与数据库的连接。

FocusBack(txtX As TextBox)：焦点返回文本框并反显内容。参数为文本框对象变量。

MakeTree(tvwX As TreeView)：生成 TreeView 控件中的系-专业-班级目录树。

AddDeptItem(cboX As ComboBox)：填充系组合框。

AddSpecItem(cboX As ComboBox, sDeptNo As String)：填充专业组合框。

AddCourseItem(cboX As ComboBox)：填充课程组合框。