

Visual Basic 6.0 基础教程

第一章 Visual Basic 6.0 概述

VB6.0 基础教程 1.1 Visual Basic 6.0 的特点

Visual Basic 是 Microsoft 公司推出的一个集成开发环境，具有简单易学、功能强大、软件费用支出低、见效快等特点。Visual Basic 继承了 Basic 语言易学易用的特点，特别适合初学者学习 Windows 系统编程。

Visual Basic 之所以受到广大编程爱好者以及专业程序员的青睐，是因为它具有以下一些特点：

1.可视化的集成开发环境

"Visual"指的是开发图形用户界面（GUI）的方法。在使用过去的一些语言如 C 语言、Basic 语言编写程序时，最令程序员烦恼的是编写友好的用户界面。使用 Visual Basic 编写应用程序，则不需编写大量代码去描述界面元素的外观和位置，而只要把预先建立的对象添加到屏幕上即可。

"Basic"指的是 BASIC（Beanner's All-Purpose Symbolic Instruction Code）语言--一种在计算技术发展历史上应用得最为广泛的语言。

Visual Basic 在原有 BASIC 语言的基础上进一步发展，至今已包含了数百条语句、函数及关键词，其中很多和 Windows GUI 有直接关系。专业人员可以用 Visual Basis 实现其它任何 Windows 编程语言的功能，而初学者只要掌握几个关键词就可以建立实用的应用程序。

可见,从 BASIC 语言发展到 Visual Basic,也就是将一们单纯的计算机语言发展成为一个集应用程序开发、测试、查错功能于一体的集成开发环境。

2.面向对象的程序设计思想

面向对象的程序设计是伴随 Windows 图形界面的诞生而产生的一种新的程序设计思想,与传统程序设计有着较大的区别,Visual Basic 就采用了面向对象的程序设计思想。所谓"对象"就是一个可操作的实体,如窗体,以及窗体中的按钮、文本框等控件。每个对象都能响应多个不同的事件,每个事件均能驱动一段代码(事件过程),该段代码决定了对象的功能。我们称这种机制为事件驱动。事件由用户的操作触发。例如,单击一个按钮,则触发按钮的 Click(单击)事件,处于该事件过程中的代码就会被执行。若用户未进行任何操作(未触发事件),则程序将处于等待状态。整个应用程序就是由彼此独立的事件过程构成,因此,使用 VB 创建应用程序,就是为各个对象编写事件过程。

3.交互式的开发环境

Visual Basic 集成开发环境是一个交互式的开发环境。传统的应用程序开发过程可以分为 3 个明显的步骤:编码、编译和测试代码。但是 Visual Basic 与传统的语言不同,它使用交互式方法开发应用程序,使 3 个步骤之间不再有明显的界限。

在大多数语言里,如果编写代码时发生了错误,则在开始编译应用程序时该错误就会被编译器捕获。此时必须查找并改正该错误,然

后再次进行编译。对每一个发现的错误都要重复这样的过程。而 Visual Basic 在编程者输入代码时便进行解释，即时捕获并突出显示人多数语法或拼写错误，看起来就像一位专家在检查代码的输入。

除即时捕获错误以外，Visual Basic 也在输入代码时部分地编译该代码。当准备运行和测试应用程序时，只需极短时间即可完成编译。如果编译器发现了错误，则将错误突出显示于代码中。这时可以更正错误并继续编译，而不需从头开始。

由于 Visual Basic 的交互特性，因此可以在开发应用程序时运行它。通过这种方式，代码运行的效果可以在开发时就进行测试，而不必等到编译完成以后。

4.高度的可扩充性

Visual Basic 是一种高度可扩充的语言，除自身强大的功能外，还为用户扩充其功能提供了各种途径，主要体现在以下 3 方面：

(1)支持第三方软件商为其开发的可视化控制对象：Visual Basic 除自带许多功能强大、实用的可视化控件以外，还支持第三方软件商为扩充其功能而开发的可视化控件，这些可视化控件对应的文件扩展名为 OCX.只要拥有控件的 ocx 文件，就可将其加入到 VB 系统中，从而增强 VB 的编程能力。

(2)支持访问动态链接库（ Dynamic Link Library,DLL）:Visual Basic 在对硬件的控制和低级操作等方面显得力不从心，为此 VB 提供了访问动态链接库的功能。可以利用其它语言，如 Visual C++语言，将需要实现的功能编译成动态链接库（DLL）,然后提供给 VB 调用。

(3) 支持访问应用程序接口 (API):应用程序接口 (Application ProgmrInterface,API)是 Windows 环境中可供任何 Windows 应用程序访问和调用的一组函数集合。在微软的 Windows 操作系统中, 包含了 1000 多个功能强大、经过严格测试的 API 函数, 供程序开发人员编程时直接调用。Visual Basic 提供了访问和调用这些 API 函数的能力, 充分利用这些 API 函数, 可大大增强 VB 的编程能力, 并可实现一些用 VB 语言本身不能实现的特殊功能。

VB6.0 基础教程 1.2.1 安装 VB 6.0 的系统需求

在安装 Visual Basic 之前, 必须确认计算机满足相应的硬件和软件配置。这些配置包括:

Pentium (R) 90MHz 或更高的微处理器。

Microsoft Windows 支持的 VGA 或分辨率更高的监视器。

对于 Windows 95 需要 24MB 的内存, 对于 Windows NT 则需要 32MB 的内存。

一个 CD-ROM 驱动器。

Microsoft Windows NT 3.51 或 Microsoft Windows95,或两者的更新版本。

Microsoft Internet Explorer 4.01 或更新的版本 (4.01 版的 Service Pack 1 或对 DHTML 应用程序开发者的更高版本, 以及对这些应用程序的最终用户的 4.x 版本)。

磁盘空间要求:

标准版本: 典型安装 48MB,完全安装 80MB.

专业版：典型安装 48MB,完全安装 80MB.

企业版：典型安装 128MB,完全安装 147MB.

附加部件（如果需要的话）:MSDN（用于文档）:67MB,Internet Explorer4.x:大约 66MB.

VB6.0 基础教程 1.2.2 安装 Visual Basic 6.0

在确认自己的电脑硬件配置和软件环境满足 Visual Basic 6.0 的安装与使用要求后,就可以开始安装 Visual Basic 6.0 了。启动计算机,进入到 Windows95/98 或 Windows NT 操作系统后,将 Visual Basic 6.0 安装盘放入光驱中,稍等片刻,屏幕上就会出现如图 1.1 所示的安装向导的界面。如果 Visual Basic 6.0 的安装盘放入光驱后没有出现该画面,用鼠标双击安装盘中的 Setup.exe 文件,也可显示如图 1.1 所示的画面。



单击【下一步】按钮就开始执行安装过程。这时将弹出关于最终用户许可协议对话框,如图 1.2 所示。



在该对话框中选中【接受协议】单选按钮，再单击【下一步】按钮。将弹出用来输入产品号和用户信息的对话框。在该对话框中输入 Microsoft 公司提供的产品 ID 号及用户姓名和公司名称，产品 ID 号同时也是安装密码。然后单击【下一步】按钮，弹出如图 1.3 所示的安装 Visual Basic 6.0 中文企业版对话框。



选中【安装 Visual Basic 6.0 中文企业版】单选按钮，单击【下一步】按钮。在弹出的欢迎界面中单击【继续】按钮，则弹出带有产品标识号的消息框，接着单击【确定】按钮则出现如图 1.4 所示的用于选择安装类型和安装路径的对话框。



在该对话框的【文件夹】设置区中显示的是系统默认的程序安装路径，用户可以自行指定程序的安装路径。单击【更改文件夹】按钮，则打开【改变目录】对话框，如图 1.5 所示。在该对话框的【驱动器】列表框中选择一个驱动器，然后在目录列表框中选择要安装 VB 的文件夹，在【路径】文本框中就会显示出所选的安装路径。用户也可以直接在【路径】文本框中输入路径，如 D:\VB6.0。路径选择完毕后，单击【确定】按钮，返回到如图 1.4 所示对话框，就会发现安装路径已被更改。



在指定了程序安装路径后，用户还需要选择一种安装类型。VB 提供了两种安装类型，分别是典型安装和自定义安装；不同的安装类型，安装的程序组件不同，需要的硬盘空间也不同，典型安装是指仅安装程序的一些最常用的选项，其中包括程序的主要内容，对于初学

者最好使用这一安装。自定义安装是指安装的组件完全由用户自己选择，但如果定制不当（如没有选择运行程序所必需的组件）可能导致程序无法正常运行。

在图 1.4 中，单击选择的安装类型前面的图标按钮，系统将检测系统是否具有所需要的硬盘空间，然后就开始正式复制文件。

文件复制完毕后，将弹出提示框，提示用户只有重新启动 Windows 才能完成程序的安装。

单击【重新启动 windows】按钮，则 Windows 系统将重新启动。系统启动成功后，屏幕上将弹出如图 1.6 所示的安装 MSDN 对话框。MSDN 中包含了 VB 的帮助文档，要在使用 VB 时能获得联机帮助，则必须安装 MSDN。用户也可以在此时不安装 MSDN，而在以后通过 MSDN CD 来单独安装 MSDN。

插入 MSDN CD，选中【安装 MSDN】复选框，单击【下一步】按钮，则开始安装 MSDN。安装完毕后，将弹出服务器安装对话框，与 MSDN 一样，服务器工具也可以单独安装。单击【下一步】按钮，则弹出如图 1.7 所示的注册对话框。不选中【现在注册】复选框，然后单击【完成】按钮即可完成对 VB 的安装。

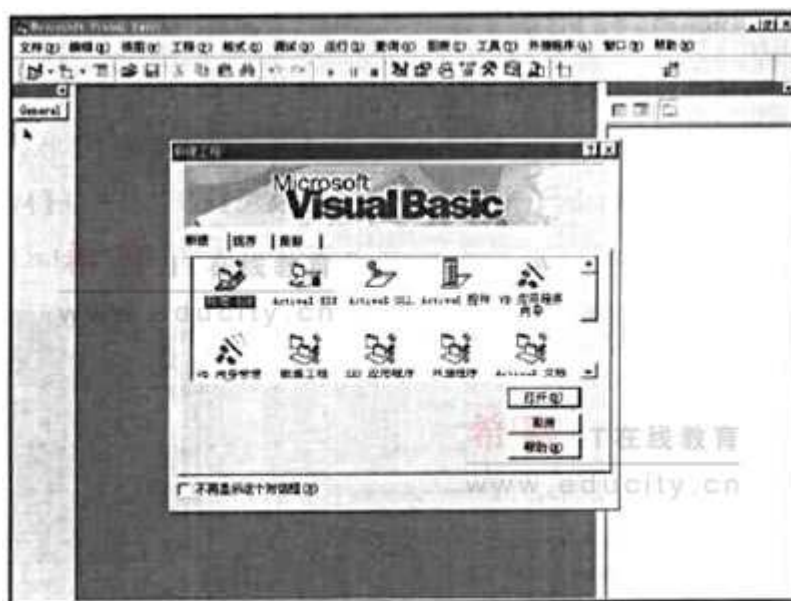
VB6.0 基础教程 1.3.1 VBasic 6.0 的启动

在 Visual Basic 安装成功后，安装程序自动在【开始】菜单中建立 VisualBasic 6.0 的程序组和程序项。单击屏幕左下角的【开始】按钮，指向【程序】选项，再指向【Microsoft Visual Basic 6.0 中文版】

程序组，单击【Microsoft Visual Basic 6.0 中文版】选项即可启动 Visual Basic 6.0 中文版。

这种方法虽然方便快捷，但有时也会遇到一些问题。一旦。安装 Visual Basic 6.0 文件夹的名称或位置改变了，利用【开始】菜单就不能启动 Visual Basic 6.0 了。因此，知道第 2 种启动 Visual Basic 6.0 的方法是必要的。通过"我的电脑"或"资源管理器"进入到 Visual Basic 6.0 所在的文件夹，在此文件夹中双击 Vb6.exe 文件，即可启动 Visual Basic 6.0。这种方法一般在使用前种方法启动失败后使用。

在 Visual Basic 6.0 启动后，屏幕上将显示如图 1.9 所示的【新建工程】对话框。在该对话框中选择希望创建的工程类型，单击【打开】按钮，即可开始使用 Visual Basic 6.0 工作了。



成功启动 Visual Basic 6.0 后，可以看到如图 1.10 所示的用户界面。熟悉 Windows 的用户可以发现，这是一个标准的 Windows 应用程序界面，其中包括标题栏、菜单栏、工具栏等基本组件。还包括工具箱、浮动窗口等组件。

提示：初次启动了 Visual Basic 6.0 时，在用户界面中可能并不会显示某些窗口，如【代码】窗口，其实，这些窗口的显示受【视图】菜单中的相关命令控制。如执行【视图】菜单中的【代码窗口】命令，即可打开【代码】窗口。

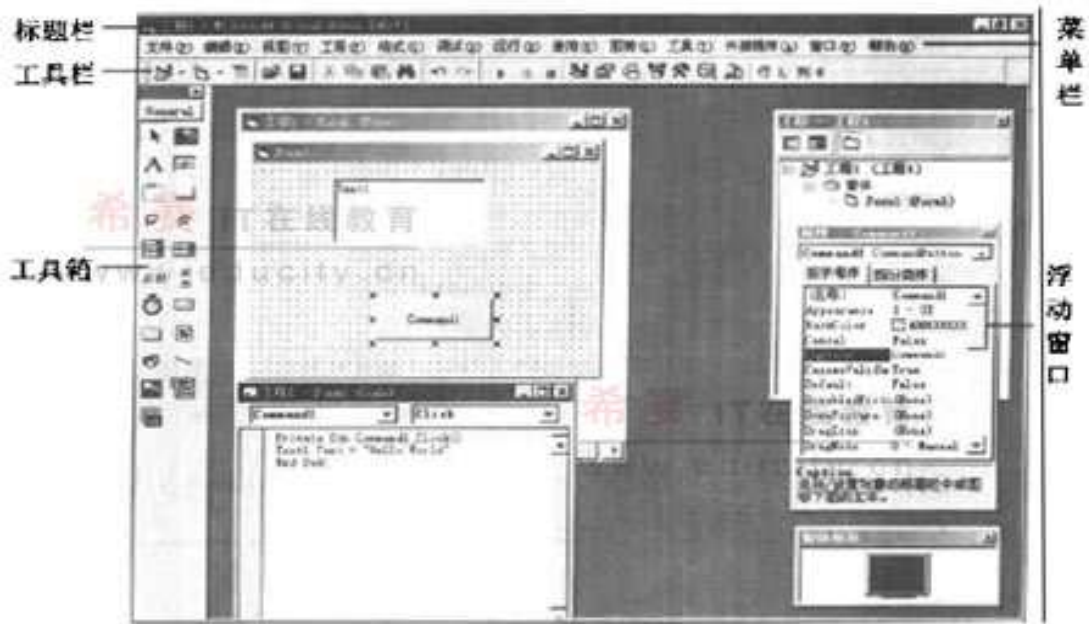


图 1.10 Visual Basic 6.0 的用户界面

VB6.0 基础教程 1.3.2 菜单栏

Visual Basic 6.0 的菜单栏包含 13 个菜单，如图 1.11 所示。除了提供了标准的【文件】、【编辑】、【视图】、【窗口】和【帮助】菜单之外，还提供了编程专用的功能菜单，例如【工程】，【格式】、【调试】等。各个菜单的功能如表 1.1 所示。

图 1.11 菜单栏

希赛 IT 在线教 表 1.1 菜单功能表

菜单	功能
【文件】	用于建立和处理文件。该菜单包括工程管理、保存文件、加入文件、打印文件及退出系统等命令
【编辑】	包含一般文本的各种编辑功能，如剪切、复制和粘贴等
【视图】	用于切换 Visual Basic 窗口的视图格式，便于用户使用 VB 的集成开发环境。包括显示和隐藏集成开发环境的各种特征，以及操作构成用户应用程序的各种对象和控件
【工程】	用于管理当前工程。主要包括在工程中添加和删除各种工程组件，显示当前工程的结构和内容等命令
菜单	功能
【格式】	主要用于编排窗体上可视控件的格式，包括自动排齐、对格线排齐等
【调试】	用于调试程序，包括设置断点、监视器、步进等
【运行】	用于在集成开发环境中运行程序。包括运行程序、编译后运行程序、中断程序、结束运行和重新开始等命令
【查询】	用于与数据库有关的查询操作
【图表】	用于完成与图表有关的操作
【工具】	用于添加菜单或各种工具栏，如过程控制、菜单设计器、工程和环境等
【外接程序】	和 VB 协调工作的内置工具选择菜单。如，可加入数据库管理器、报表设计器等工具与 VB 协调工作
【窗口】	用于调整各种类型的 VB 子窗口在主窗口的排列方式
【帮助】	用于启动 Visual Basic 的联机帮助系统

VB6.0 基础教程 1.3.3 工具栏

工具栏也是 Windows 应用程序的一个常见的组件，它为菜单栏中的常用命令提供了快捷方式。将鼠标指向某个工具按钮时，会自动显示出该按钮的名称。

表 1.2 中列出了各工具按钮的名称和功能。

VB6.0 基础教程 1.3.4 工具箱

在图 1.10 中可以看到，用户界面的左边是工具箱。工具箱中提供一些控件，它们是构造 Windows 应用程序用户界面的图形化工具。在设计时，通过将这些控件添加到窗体中，即可轻松创建出标准的

Windows 应用程序的用户界面。有又在窗体上布置控件的内容将在第 4 章中详细讲述, 本节只是介绍上工具箱本身的一些操作和各控件功能。

表 1.3 详细列出了各工具的名称及其对应的按钮形式, 并给出了各工具的功能简介。

表 1.3 标准控件功能一览表

图标	名称	功能简介
	指针	这是工具箱中唯一不绘制控件的选项。在选定指针后只能改变窗体中绘制的控件的大小, 或移动这些控件
	图片框	显示图形图像 (装饰或者活动图片), 该控件也可作为接受来自图形方法的输出容器, 或作为其他控件的容器
	标签	用来显示一些不被修改的文本, 例如一个图形下的标题
	文本框	用来输入或显示文本
	框架	用来从界面方面或在功能上对控件分组。为了将控件分组, 首先要绘制框架, 然后在框架中画出控件
	命令按钮	创建按钮, 选择它来执行某项命令
	复选框	用来接收用户所做的选择, 常常成组使用
	单选项	与复选框类似, 但在一组中只能选择其中的一项, 可以使用框架控件对单选项进行分组
	组合框	组合框是列表框和文本框的组合, 使用时可从下拉列表中选择一项, 也可在文本框中输入值
	列表框	用来显示一个列表, 列表中提供多个选项可供用户选择
	水平滚动条	可快速在水平方向上移动很长的列表或大量信息, 并在标尺上指示当前位置, 也可以作为输入设备, 或作为速度和数量的指示器
	垂直滚动条	可以快速在垂直方向上引导一个很长的列表或大量信息, 并在标尺上指示当前位置, 也可以作为输入设备, 或作为速度和数量的指示器
	计时器	在指定的时间间隔内产生 Timer 事件, 该控件在运行时不可见
	驱动器列表框	显示有效的磁盘驱动器
	目录列表框	显示目录和路径
	文件列表框	显示文件列表
	形状	使用该控件可在窗体上绘制多种形状, 这些形状包括矩形、圆角矩形、正方形、圆角正方形、椭圆形或圆形
	直线	用来在窗体上绘制各种样式的直线

(续表)

图标	名称	功能简介
	图像框	与图片框控件类似, 用来显示图形图像, 但不能作为其他控件的容器
	数据	通过窗体上被绑定的控件来访问数据库中的数据
	OLE	允许把其他应用程序创建的对象链接和嵌入到 Visual Basic 应用程序中

表 1.2 工具栏各工具功能

图标	名称	功能
	添加工程	添加一个新工程。在 VB 中可以同时编辑多个工程。可以通过其下拉菜单来选择工程的类型
	添加窗体	向当前工程中添加新部件。可以通过其下拉菜单来选择部件的类型
	菜单编辑器	打开【菜单编辑器】对话框, 用来设计菜单
	打开工程	打开一个已存在的工程, 同时关闭正在编辑的工程
	保存工程	保存所编辑的工程
	剪切	删除当前选定的控件、文本等内容, 并将它们复制到剪贴板中
	复制	将当前选定的控件、文本等内容复制到剪贴板中
	粘贴	将剪贴板中的内容复制到指定的位置, 只有剪贴板中有内容时, 该工具才有效
	查找	查找文本, 只有在代码窗口被激活时该命令才有效
	撤消	撤消上次的操作, 如添加控件、输入文本等操作
	恢复	重复上述操作, 仅当执行了撤消操作时该工具才有效
	启动	运行当前工程
	中断	中断正在运行的工程。在调试程序时常常用到中断
	结束	终止正在运行的工程, 返回到主窗口
	工程资源管理器	显示【工程资源管理器】窗口
	属性窗口	显示【属性】窗口
	窗体布局窗口	显示【窗体布局】窗口
	对象浏览器	显示【对象浏览器】窗口
	工具箱	显示工具箱
	数据视图窗口	显示数据视图窗口
	Visual Component Manager	显示 Visual Component Manager 对话框

VB6.0 基础教程 1.3.5 各种窗口简介

在 Visual Basic 6.0 的用户界面上，还分布着各种窗口，用户可以通过视图菜单中的相关命令打开需要的窗口。每个窗口都有一个特定的功能，如【代码】窗口专门用于编写代码。本节只是简单概述几个常用窗口的功能，使读者对它们有一个初步的认识。

【窗体设计】窗口。

在启动 Visual Basic 后，【窗体设计】窗口就会出现在用户界面的中央，如图 1.13 所示。如果在界面中没有出现该窗口，则可以通过执行【视图】菜单中的【对象窗口】命令来打开它。【窗体设计】窗口是设计应用程序界面的地方，它也是 Visual Basic 中最重要的一个窗口。

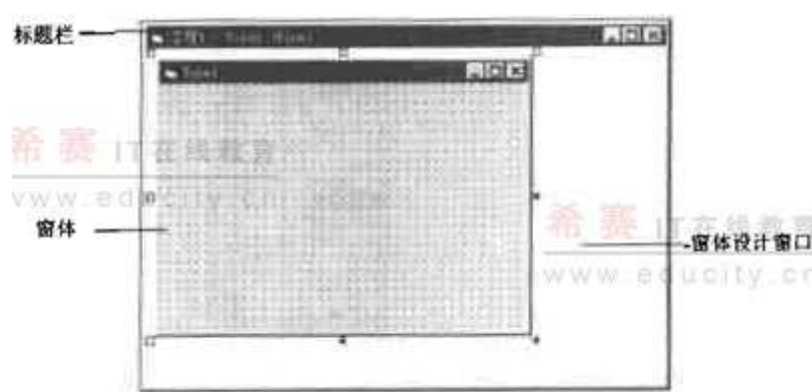


图 1.13 【窗体设计】窗口

在【窗体设计】窗口的左上角有一个窗体，这个窗体就是应用程序最终面向用户的窗体。在设计应用程序时，窗体就像一块画布，用户可在其中添加控件、图片以及菜单等组件来设计用户界面。

在窗口的标题栏上还显示了当前工程的名称以及其中窗体的名称。在图 1.13 所示的【窗体设计】窗口中，“工程 1”是工程的名，Form 1 是窗体名。如果应用程序包含有几个窗体，则每个窗体都有自己的设计窗口。

可以看出，【窗体设计】窗口中的窗体与【窗体设计】窗口本身很相似，都有标题栏、控制图标与控制按钮等。在设计阶段，窗体上的控制图标与控制按钮（除最大化按钮）不可用，它们此时只是一个外观，用户可以更改这些外观，如控制图标、窗体标题等。在程序运行阶段，它们都是可用的，不需要用户编写任何代码。窗体的大小、背景色等都是用户可以自行定制的，关于窗体的设计将在第 4 章中详细介绍。

【属性】窗口

属性是指窗体和控件等对象的特征，如大小、标题、颜色、位置等。通过【属性】窗口，用户可快速地设置对象的属性。在属性列表中设置了窗体或控件的属性后，在【窗体设计】窗口中即可看到效果。执行【视图】菜单中的【属性窗口】命令或单击工具栏中的【属性窗口】按钮均可打开【属性】窗口。

与工具箱一样，【属性】窗口通常浮动在主窗口中，双击它的标题栏，或将鼠标指针移动到它的标题栏下，向主窗口的边界拖动窗口，可以使【属性】窗口横向连接到主窗口上，如图 1.15 所示。再次双击【属性】窗口的标题栏，或使用鼠标向主窗口中心方向拖动窗口，可使窗口恢复到浮动状态。

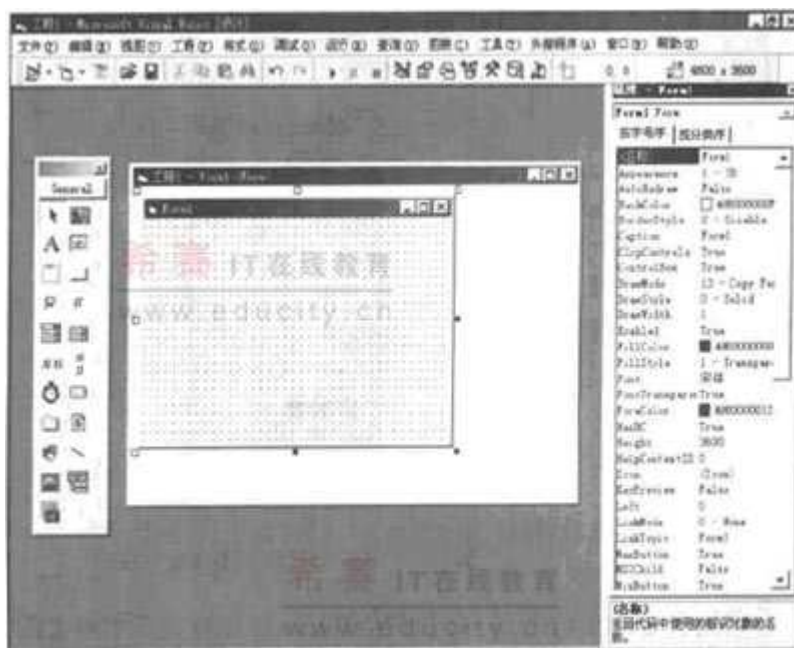


图 1.15 将【属性】窗口连接到主窗口的右方

由于不同的对象所具有的属性是不同的，因此，对于不同的对象，【属性】窗口中所显示的属性也是不同的。在【属性】窗口的标题栏显示有对象的名称，用户可从中看出当前【属性】窗口中列出的是哪个对象的属性。

在【属性】窗口中列出指定对象属性的方法有两种：

在【窗体设计】窗口中选中某对象，则在【属性】窗口中就列出该对象的属性。

位于【属性】窗口上方的列表框叫做对象框，其中列出了当前的窗体及其上所放置的控件。单击该列表框，可从中选择要设置属性的窗体或控件。

位于【属性】窗口中部的是属性列表，该列表分为两栏，左栏显示的是属性的名称，右栏显示的是属性的值。例如，在如图 1.16 所示的【属性】窗口中，Caption 是属性名，Form1 则是它的值。

在默认情况下，属性是按字母顺序出现在属性列表中，单击【按分类序】选项卡，则各属性按外观、位置等逻辑类出现在属性列表中，如图 1.16 所示。单击【按字母序】选项卡，则属性又恢复为按字母顺序排列。

位于【属性】窗口下方的是信息栏，在属性列表中单击选中某一属性（属性名以蓝色亮条显示）后，在信息栏中就会显示出该属性的名称以及功能。

【代码】窗口。

【代码】窗口（如图 1.17 所示）是输入应用程序代码的编辑器，应用程序的每个窗体或代码模块都有一个单独的【代码】窗口。在标题栏上显示有工程的名称和窗体的名称，从中可以看出该【代码】窗口属于哪个工程的哪个窗体。

只有在编写程序代码时，才需要使用到【代码】窗口。在 VB 启动后，【代码】窗口并不出现在界面中，可以通过以下几种方法来打开它：

双击窗体或窗体上的控件。

执行【视图】菜单中的【代码窗口】命令。

单击工程资源管理器中的【查看代码】按钮。



图 1.17 【代码】窗口

【代码】窗口的标题栏下面有两个列表框，左边的列表框是对象框，右边的列表框是事件框。这里的对象框与【属性】窗口中的对象框是一样的，单击该列表框，则在弹出的下拉列表中列出了当前窗口中所有对象的名称，用户可以在【属性】窗口中的"名称"属性中为对象指定名称。对于窗体对象，由于每个【代码】窗口对应唯一的窗体，因此，在对象框中，"窗体"这个对象用 **Form** 标识，而不是使用用户在【属性】窗口中为窗体指定的名称来标识。

在事件框中列出了所选对象所能响应的事件。关于对象与事件的概念将在第三章中讲述。

在默认情况下，代码编辑区中显示出用户编写的所有过程。在查看某过程中的代码时，为了避免其他过程的干扰，可单击【代码】窗口左下角的【过程查看】按钮圈，则只显示当前插入点所在的过程。单击【全模块查看】按钮则代码编辑区恢复显示出用户编写的所有过程。

第二章 Visual Basic 6.0 中的基本概念与操作

VB6.0 基础教程 2.1.1

属性是指对象的各种性质，如对象的位置、颜色、大小等。不同的对象所具有的属性有的是相同的，有的是不同的。例如，收音机有个"音量"属性，水杯就没有"音量"属性，但它有个"容量"属性，而收音机却没有。此外，收音机和水杯都有个"颜色"属性。

1. 设置属性的值。

改变对象的属性就可改变对象的特性。例如，改变收音机"音量"属性的值就可调节收音机音量的大小。可以通过两种方法来设置对象的属性：

在设计阶段，通过【属性】窗口设置对象属性的值。对不同的属性，设置方式有所差异，在第 4 章中，将结合窗体属性的设置来详细介绍。在运行阶段，在程序中由代码设置对象属性的值。其一般形式为：对象名。属性名=属性值

例如，假定收音机的音量值可设置在 0~10 之间。如果能够通过 Visual Basic 控制收音机，则可在程序代码中使用下列语句将收音机的音量调节到中等音量：

```
Radio.Volume=5
```

提示：在代码中使用的属性名称与在【属性】窗口中列出的属性名称是相同的，但 Font 属性例外。在【属性】窗口中，通过 Font 属性可以同时设置对象上所显示文本的字体、字号以及下划线等属性。在代码中，字体、字号等属性分别对应一个属性名。在本节的最后将详细介绍。

上述两种属性设置方法的特点是：

在设计阶段，通过【属性】窗口设置对象的属性值，不需要编写任何代码，且对于对象的一些外观属性，在【属性】窗口设置了相应的位后，在【窗体设计】窗口中即可预览到设置的效果。【属性】窗口主要用来设置对象属性的初始值和一些在整个程序运行过程中不改变的属性。

在运行阶段，在程序中由代码设置对象属性的值，可以在程序运行时随时改变对象属性的值。例如，在程序运行时，用户调整收音机音量的大小，其实就是通过在代码中重新设置"音量"属性的值来实现的。此外，有的属性在设计时是不可用的，因此，这些属性只有通过代码在运行时设置。

在 VB 中，每个对象的各个属性都有一个默认值，在实际应用中，大多数属性都采用系统提供的默认值。因此。用户一般不必一一设置对象各属性的值，只有在默认值不满足要求时，才需要用户指定所需的值。

2.读取属性的值

在代码中不仅能设置属性的值，还能读取属性的值。在运行时可以设置并可获得其值的属性叫做读写属性；在运行时只能读取的属性叫做只读属性。

有时，要在执行某操作之前得知对象的状态，这时就要读取属性值。例如，想要将收音机的音量增大一点，在执行该操作前就需要得到当前音量的大小，以确定将"音量"属性的值设置为多少。

在大多数情况下可以用以下语法读取属性的值：

变量=对象。属性。

例如，下列语句就是将当前音量的值赋给变量 Col:

Col=Radio.Volume.

属性值也可以作为较复杂的表达式的一部分，而不必将属性值赋予变量。下面的代码是将收音机的音量在原来的基础上调大一点：

```
Radio.Volume=Radio.Volume+1.
```

3.常用的属性。

在使用 VB 创建一个应用程序时，很重要的一步就是设置窗体以及控件等对象的属性，表 2.1 中列出了几个常用的属性，这些属性也是大多数对象所共有的。

表 2.1 几个常用的属性

属性	说明
名称	每个对象都有一个名称属性，在代码中正是通过名称来访问对象的。如收音机的名称是 Radio，在代码中，总是使用 Radio 来表示对象收音机
Appearance	决定控件和窗体的外观。值为 0 时表示平面外观，值为 1 时表示三维外观。系统默认值为 1。现在多数 Windows 应用程序均采用三维外观，因此，建议读者在设计应用程序时就使用该属性的默认值
BackColor	设置对象的背景颜色
ForeColor	设置对象的前景颜色
Font	设置对象上文本的字体、字号等属性
Caption	设置对象上显示的文本。如窗体的标题、按钮上的提示文字、复选框旁边的文字等
Width	设置对象的宽度
Height	设置对象的高度
Left	指定控件左上角的横坐标
Top	指定控件左上角的纵坐标
Enabled	决定对象是否可用。值为 True 时，对象可用，值为 False 时，对象不可用
Visible	决定对象是否可见。值为 True 时，对象可见，值为 False 时，对象不可见

VB6.0 基础教程 2.1.2 方法

除了属性以外，对象还有方法，属性是指对象的特性，而方法则是对象要执行的动作。不同的对象所具有的方法也是不同的。以拨号打电话为例，可以说电话（Phone）对象有一个“拨号”（Dial）方法，拨一个 7 位电话号码的语法就是：

```
Phone.Dial 5551111.
```

在代码中使用方法时如何书写语句，取决于该方法要求多少参数，以及是否返回同一个值，如果方法不要求参数，则用以下语法编写代码：

对象名。方法名。

例如，窗体对象有一个 `Cls` 方法，该方法的功能是清除窗体上显示的文本或图形等内容。调用该方法的语句如下：

窗体名。`Cls`。

有些方法还带有参数，参数是对方法所执行动作的进一步描述。在调用这类方法时要在方法名的后面写上参数。如电话的“拨号”方法就有一个参数，该参数用来说明拨什么号。如果方法有多个参数，就用逗号将它们分开。

例如，窗体对象的 `Circle` 方法就有多个参数，该方法的功能是在窗体上画圆。

使用该方法需要指定圆的位置、半径和颜色等参数：

`Form1.Circle (1600, 1800), 1200.vbBlue` 有的方法还有返回值，如果保存方法的返回值，就必须把参数用括号括起来：例如，剪贴板的 `GetData` 方法是返回一张图片：

`Picture=Clipboard.GetData (vbCfBitmap) .`

如果没有返回值，则参数不出现在括号中。

使用对象的方法与属性的语法格式有些类似，属性和方法与它们的拥有者——对象都是以一个点来连结。在实际操作中，可以通过词

性来判断是属性还是方法。属性名一般是名词(如 Appearance, Caption, Width 等),方法名一般是动词。

另外,在程序代码中,"对象名.方法名"可以是一个完整的语句,但"对象名.属性名"不是一个完整的语句。在代码中,涉及到对象属性的语句总是一个赋值语句,要么是给对象的属性赋值,要么是将对象的属性值返回给一个变量。

VB6.0 基础教程 2.1.3 事件

事件是指由系统事先设定的、能被对象识别和响应的动作。例如,在应用程序中单击一个按钮,则程序会执行相应的操作。在 VB 中,就称按钮响应了鼠标的单击事件。

传统的高级语言程序由一个主程序和若干个过程和函数组成,程序运行时总是从主程序开始,由主程序调用各过程和函数。程序设计者在编写程序时必须将整个程序的执行顺序十分精确地设计好。程序运行后,将按指定的过程执行,用户不能改变程序的执行顺序。因此,这种语言称为面向过程的语言。

VB 程序没有传统意义上的主程序,在 VB 中,子程序称为过程。VB 中有两类过程:事件过程和通用过程。程序的运行并不要求从主程序开始,每个事件过程也不是由所谓的"主程序"来调用,而是由相应的"事件"触发执行,通用过程则是由各事件过程来调用。例如,单击鼠标按钮,系统将跟踪指针所指的对象,如果对象是一个按钮控件,则用户的单击动作就触发了按钮的 Click 事件,该事件过程中的代码就会被执行。执行结束后,又把控制权交给系统,等待下一个事件发

生。各事件的发生顺序完全由用户的操作决定，这样就使编程的工作变得比较简单了，人们不再需要考虑程序的执行顺序，只需针对对象的事件编写出相应的事件过程即可。我们称这些应用程序为事件驱动应用程序。

在事件驱动应用程序中，由对象来识别事件。事件可以由一个用户动作产生，如单击鼠标或按下一个键；也可以由程序代码或系统产生，如计时器。使用、由创建应用程序，其实就是为每个对象，如窗体、控件、菜单等编写事件代码。因此，VB 是面向对象的编程语言。

触发对象事件的最常见的方式是通过鼠标或键盘的操作。我们将通过鼠标触发的事件称为鼠标事件，将通过键盘触发的事件称为键盘事件。

1.鼠标事件。

现在，多数应用程序是通过鼠标来操作的，如单击按钮、选择菜单等。鼠标的操作有单击、双击、移动等几种，它们分别能触发一个事件，表 2.2 中列出了鼠标的操作及其所触发的事件。

表 2.2 鼠标事件

事件	操作
Click	单击鼠标左键
DbClick	双击鼠标左键
MouseDown	按下鼠标键
MouseUp	释放鼠标键
MouseMove	移动鼠标。移动鼠标时，连续触发 MouseMove 事件

从表中可以看出，鼠标的单击操作实际上会触发 3 个事件。当按下左键后，触发 MouseDown 事件，释放左键后，触发 MouseUp 事件，同时又触发了 Click 事件。

2.键盘事件。

有些用户则更习惯使用键盘来操作，表 2.3 中列出了鼠标的操作及其所触发的事件。

表 2.3 键盘事件

事件	操作
KeyDown	按下键
KeyUp	键弹起
KeyPress	按键

同样，键盘的按键操作实际上也会触发 3 个事件。当按下某键后，触发 KeyDown 事件，键被弹起后，触发 KeyUp 事件，同时又触发了 KeyPress 事件。

每一种对象所能识别的事件是不同的。例如，窗体能响应 Click（单击）和 DBClick（双击）事件，而命令按钮能响应 Click 却不能响应 DBClick 事件。每一种对象所能响应的事件在设计阶段可以从该对象的【代码】窗口右边过程框中的下拉列表中看出（参见第 1 章）。一个对象通常能响应多个事件，但没有必要编写每一个事件过程（或为每一个事件编写代码）。例如，按钮控件可以响应 Click,MouseMove（鼠标移动）等事件，但通常只编写 Click 事件过程。因此，在多数应用程序中，单击按钮，则程序会做出相应的操作，而在按钮上移动鼠标，则程序不会有任何反应。

VB 的事件过程的一般形式为：

Private::对象名_事件名（ ）

End sub

其中"Private Sub 对象名_事件名（）"是事件过程的开头，End Sub 是事件过程的结尾。"对象名"就是用户在【属性】窗口中为对象的"名称"属性设置的值，事件名则是该对象所能响应的事件的名称。

VB6.0 基础教程 2.2.1 在窗体中布置控件

设计应用程序界面的最重要的一项内容就是在窗体中布置控件。向窗体中添加控件的方法很简单，首先，在工具箱中单击所要添加的控件，然后，移动鼠标到窗体上，光标的形状变为十字形，在要放置控件的位置处按下鼠标左键并拖动鼠标，就出现一个短形象区域，拖动到一定大小后释放鼠标，则所选控件就被放置在窗体上指定的位置。鼠标拖动出的方框的大小决定了控件的大小。如图 2.1 所示是在窗体中添加了两个文本框和两个按钮控件。

还可以随时改变放置在窗体中的控件的大小和位置。单击窗体中的某控件，则在控件的周围会出现 8 个小方块，如图 2.2 所示，这表明该对象被选定。将鼠标指针移动到某个小方块上，则指针形状变成双向箭头状，拖动鼠标，则会出现一个黑色的方框随鼠标一起移动，拖动到合适的大小后释放鼠标，控件的大小就被改变了。将鼠标移动到控件上，拖动鼠标，就出现一个和控件一样大小的黑色方框随着鼠标一起移动，到合适的位置后释放鼠标，控件就被移动到了新的位置。



图 2.1 放置了控件的窗体

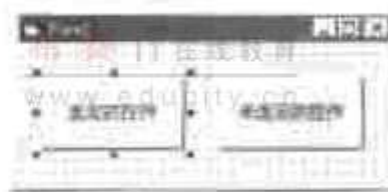


图 2.2 控件被选定时的情形

双击工具箱中的控件，也可以将它添加到窗体中。并且控件总是以特定的大小添加到窗体的中心位置。

窗体中的网格是为方便布置控件而提供的参考线，在程序运行时窗体上并不显示这些网格。用户也可以自行设置有关网格的显示形状。

单击【工具】菜单，执行【选项】命令，弹出【选项】对话框，单击该对话框中的【通用】选项卡，弹出如图 2.3 所示的对话框。该对话框的【窗体网格设置】设置区用来设置有关网格的各种选项。不选中【显示网格】复选框，则在设计时窗体中将不出现网格。【宽度】与【高度】文本框用来设置网格的大小。在默认情况下，控件总是与网格对齐的，如果不选中【对齐控件到网格】复选框，则网格将不再限制控件的大小与位置。

为了创建出友好的用户界面，往往要求各控件的大小完全相等、各控件严格对齐或等间距分布。用户除了可以借助网格通过手动调整来达到上述要求之外，还可以使用【格式】菜单中提供的各种命令来实现。

在使用【格式】菜单中命令时要事先选定要执行操作的控件。按住 Shift 键，然后依次单击要选定的控件，则这些控件就会被同时选定。在所有被选定的控件中，只有一个控件周围的小方块是实心的，其他都是空心的，如图 2.4 所示。在使用【格式】菜单中的命令操作这些控件时，将以周围是实心方块的控件为基准。例如，执行【格式】菜单中的统一尺寸操作，则所有选中控件的大小都将与周围是实心方块的控件的大小相同。



图 2.3 【选项】对话框的【通用】选项卡

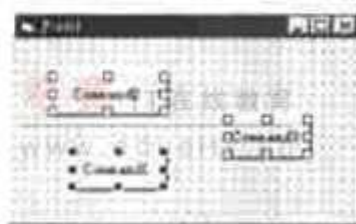


图 2.4 同时选定多个控件的情形

在控件的大小以及位置调整完成后,为了避免遭到以后操作的破坏,可以通过执行【格式】菜单中的【锁定控件】命令将所有的控件锁定。

选定控件后,按 Delete 键即可将所选的控件删除。也可以执行【编辑】菜单中的【剪切】与【删除】命令来删除控件。

VB6.0 基础教程 2.2.2 设置对象属性的方式

单击工具栏中的【属性】按钮打开【属性】窗口,在属性窗口中显示的是当前选中对象的属性。要使属性窗口中显示某一对象(窗体或控件)的属性,只需选中该对象就可以了。

在通过【属性】窗口设置窗体或控件的属性时,设置方式大致可分为如下 3 种情况:

直接输入属性的值。有些属性,当使用鼠标单击它的值时,则插入点就出现在文本框中,即可在其中输入属性的值。按回车键或选择其他属性,表明对新属性值的输入结束与确认。如 Caption, Height, Width 等属性就属于这种类型。在前面我们曾以这种方式设置过 Caption 属性的值。

在下拉列表中选择属性的值。有些属性，当使用鼠标单击它时，则在右边出现一个下三角形按钮，单击该按钮则出现一个包含可选属性值的下拉列表，用户可从该列表中选择所要的值。

在对话框中选择属性的值。还有一些属性，当使用鼠标单击它时，则在右边出现一个省略号按钮，单击该按钮，会出现一个对话框，如图 2.6 所示。用户可从该对话框中选择属性的值。如 Picture、Font 属性就属于这种类型。



图 2.6 在对话框中选择属性的值

VB6.0 基础教程 2.2.3 编写对象的事件过程

将控件放置在窗体上，并且在【属性】窗口中设置了它们的属性后，单击【运行】按钮，就会出现这个设计好的用户界面。用户可以对其执行一些操作，如在文本框中输入内容、单击按钮等。但这个用户界面仅是一个“外壳”，它不会响应用户的任何操作。

用户的操作，对对象（窗体或控件）来讲就是触发它的某个事件。如用户单击一个按钮，就激发了这个按钮的 Click 事件。因此，要使对象能够响应用户的操作，就必须编写控件的事件过程。

编写对象事件过程的流程如下：

- (1) 打开对象所在窗体的【代码】窗口。

(2) 选择对象的事件。

(3) 为事件添加代码。

双击窗体或窗体上的控件打开【代码】窗口后，窗口的代码编辑区中会自动出现与用户双击对象相对应的某一事件过程的框架。例如，双击窗体，则打开的代码窗口如图 2.7 所示，其中显示有窗体对象的 Load（装载）事件过程。再次双击窗体上的其他控件，则窗口的代码编辑区又会出现双击对象的某一事件过程的框架。图 2.8 所示的是在双击窗体后，又双击窗体中按钮控件后的【代码】窗口。



图 2.7 双击窗体打开的【代码】窗口



图 2.8 双击按钮控件后的【代码】窗体

如果【代码】窗口中所出现事件过程不是用户所要编写的事件过程，用户还可以在【代码】窗口中自行选择所要编写的事件过程，操作步骤如下：

(1) 单击【代码】窗口的对象框，在对象列表中选择要编写事件过程的对象。

(2) 单击【代码】窗口的事件框，在事件列表中为对象选择所要响应的事件。

选择完毕后，在【代码】窗口的编辑区中就出现了一个事件过程。如图 2.9 所示的【代码】窗口是选择了 Form 对象和 Click 事件后的情形；用户只需向该事件过程中添加代码即可，省去了输入过程框架的麻烦。

对于一些出现在代码窗口中但不使用的事件过程，建议用用户将其删除，避免由于它们的存在而使程序代码显得杂乱。

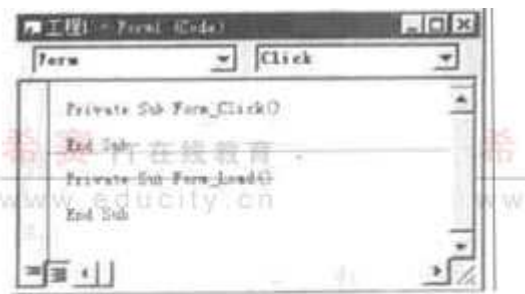


图 2.9 在【代码】窗口中显示事件过程

VB6.0 基础教程 2.3 焦点概述

焦点是对象接收用户鼠标或键盘操作的能力。当对象具有焦点时，才可接收用户的操作。例如，在有几个文本框的窗体中，只有具有焦点的文本框才能接受用户的输入。

下列方法可以将焦点赋给对象：

运行时选择对象。

运行时用快捷键选择对象。

在代码中使用 **SetFocus** 方法。

焦点只会出现在活动窗口中，并且，在活动窗口中每一时刻只能有一个对象具有焦点。有些对象，它是否具有焦点是可以看出来的。例如，当按钮具有焦点时，按钮标题周围的边框将突出显示，如图 2.10 所示。当文本框具有焦点时，文本框中将显示闪动的插入点。



图 2.10 具有焦点的按钮

并非所有的对象都能接受焦点。例如，标签、框架、定时器等控件就不能接受焦点。对于可以接受焦点的对象，只有当对象的 **Enabled** 和 **Visible** 属性为 **True** 时，它才能接收焦点。**Enabled** 属性允许对象响应由用户产生的事件，如键盘和鼠标事件。**Visible** 属性决定了对象在屏幕上是否可见。对于窗体来讲，只有窗体中不包含任何可接收焦点的控件，它才能接收焦点。

当对象得到或失去焦点时，会产生 **GotFocus** 或 **LostFocus** 事件。窗体和多数控件能响应这些事件。**GotFocus** 事件在对象得到焦点时发生；**LostFocus** 事件在对象失去焦点时发生。

使用鼠标来使对象具有焦点有时很不方便。例如，若窗体中有多个文本框，在输入数据时，如果每次总是使用鼠标来切换文本框的焦点，是一件很烦人的事。人们通常习惯使用 **Tab** 键来使对象按指定的顺序获得焦点，这就是所谓的 **Tab** 键顺序。

在使用 **VB** 创建程序时，可以使用 **TabIndex** 和 **TabStop** 两个属性来指定对象的 **Tab** 键顺序。通常情况下，**Tab** 键顺序与窗体所安放对象的顺序相一致。

1. **TabIndex** 属性。

该属性用来设置对象的 Tab 键顺序。在默认情况下，第一个被安放的控件，其 TabIndex 取值为 0;第二个被安放的控件，其 TabIndex 取值为 1,依次类推。在程序运行时，焦点默认位于 TabIndex 取值最小的控件上。当按 Tab 时，焦点按对象 TabIndex 属性的值顺序切换。

如果需要改变某个控件的 TabIndex 取值或在窗体上插入、删除控件时，VB 会自动地将其他控件的 Tab 键顺序重新编号，以反映变化了的情况。

2.TabStop 属性。

该属性的作用是决定用户是否可以使用 Tab 键来使对象具有焦点。当一个对象的 TabStop 属性取值为 True（默认）时，使用 Tab 键可以使该对象具有焦点：若它取值为 False,则 Tab 操作时将跳过该对象，即不能使用 Tab 键使该对象具有焦点。

实例 2.1 Tab 键顺序

在窗体中放置 4 个文本框控件和两个按钮控件，如图 2.11 所示。控件上所显示的标号是控件的放置顺序。例如，显示有 1 的文本框是第一个放置到窗体中的，显示有"6"的按钮是最后一个放置到窗体中的。

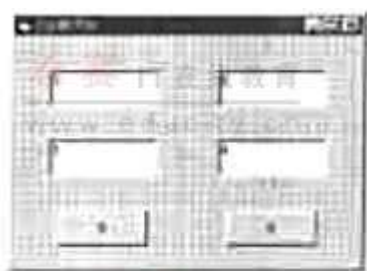


图 2.11 对象的 Tab 键顺序

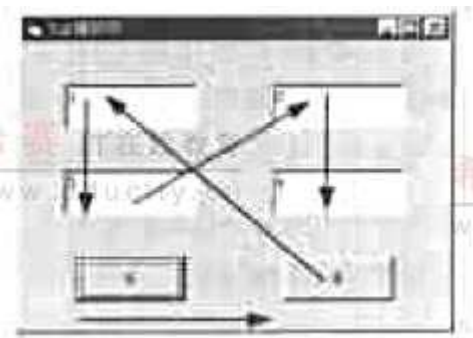
运行该程序，可见，默认时文本框 1 具有焦点，按 Tab 键，则焦点按控件的放置顺序切换，即 1→2→3→4→5→6。

停止程序的运行，返回到设计界面中，设置各控件的 TabIndex 属性如表 2.4 所示。

表 2.4 实例 2.1 中各控件的 TabIndex 属性设置

对象	TabIndex 属性值
文本框 1	2
文本框 2	4
文本框 3	3
文本框 4	5
按钮 5	0
按钮 6	1

再次运行程序，可见，默认时按钮 5 具有焦点，按 Tab 键，则焦点的切换顺序为 5→6→1→3→2→4。如图 2.12 所示。



如果将文本框 2 的 TabStop 属性设置为 False,则按 Tab 键时，焦点不会在文本框 2 上停留，焦点的切换顺序为 5→6→1→3→4。

VB6.0 基础教程 2.4 第一个 VB 应用程序

在前面已经详细介绍了 Visual Basic 6.0 的集成开发环境以及一些基本的概念，在本节中，我们以一个实例来进一步理解对象的有关概念，尽管我们还没有学习 VB 语言的语法规则，相信读者通过创建这一个简单的程序，会对使用 VB 编程有一个总的认识，同时也会领略到 VB 的简单易用。

这里创建的是这样一个程序，该程序只有一个窗体，在窗体中有一个文本框和一个按钮，在程序运行后，文本框中显示"第一次接触VB"几个字，如图 2.13 所示。单击按钮，则文本框中的内容变成"VB原来如此简单易学",如图 2.14 所示。



图 2.13 启动应用程序后的情形



图 2.14 单击按钮后的情形

在 VB 中创建程序首先是设计程序的用户界面，然后是编写事件过程。该应用程序的用户界面很简单，只有一个文本框和一个按钮，可以将工具箱中的相应控件直接添加到窗体上。该程序中只有按钮响应用户的一个操作，即鼠标的单击。因此，只需编写按钮的 Click 事件过程即可。

创建该应用程序的步骤如下：

(1) 首先，启动 Visual Basic 6.0,在【新建工程】对话框中选择创建的工程类型为标准 EXE,单击【打开】按钮。则出现一个名称为"工程 1"的设计窗口。在这个窗口中只有一个名称为 Form1 的窗体。

(2) 单击工具箱中的文本框控件，将鼠标移动到窗体 Form1,从窗体某位置开始拖动鼠标，绘制出一个人小合适的方框后释放鼠标，则窗体上就出现了一个文本框。

(3) 再单击工具箱中的按钮控件，以同样的方法在该窗体上放置一个按钮控件，如图 2.15 所示。

(4) 单击工具栏中的【属性窗口】按钮打开[属性]窗口。

(5) 在窗体上单击文本框控件，则文本框的四周出现了一些小蓝点，这表明文本框已被选中。此时，在【属性】窗口中所列出的就是该文本框的属性，如图 2.16 所示。



图 2.15 应用程序界面



图 2.16 设置文本框的 Text 属性

(6) 文本框的 Text 属性决定文本框中显示的内容，在默认情况下，Text 属性的值为 Text1.单击 Text1 所在的区域，则在该区域中就会出现一个插入点，这时，就可以重新编辑其中的内容了。在其中输入"第一次接触 VB"几个字。在输入的同时，窗体上文本框中的内容也随着改变。

(7) 同理，在窗体中选中按钮控件，在【属性】窗口中设置 Caption 属性的值为"欢 迎".

(8) 再选中窗体，在【属性】窗口中设置 Caption 属性的值为"第一个 VB 程序".自此，程序的用户界面设计完毕，如图 2.17 所示。可见，通过设置对象的属性，改变了对象的外观。

(9) 双击按钮控件，打开【代码】窗口，在代码窗口中自动出现了按钮 Click 事件过程的框架，如图 2.18 所示，用户只需为该过程添加代码即可。



图 2.17 设置属性后的窗体



图 2.18 【代码】窗口

(10) 将下列代码添加到 Command1_Click 事件过程中

```
Private Sub Command1_Click ()
```

```
Text1.Text="VB 原来如此简单易学"
```

```
End Sub
```

在 Command1_Click 事件过程中只有一行语句，该语句的含义是将文本框的 Text 属性的值设置为"VB 原来如此简单易学".在【属性】窗口中，我们曾将文本框的 Text 属性的值设置为"第一次接触 VB"，在程序启动后，文本框中将显示"第一次接触 VB",而不会显示"VB 原来如此简单易学".只有当用户单击按钮后，Command1_Click 事件过程被触发，过程中的语句才会被执行，执行的结果是文本框中的内容被替换为"VB 原来如此简单易学".

自此，一个简单的应用程序就创建完毕了。单击工具栏中的【运行】按钮来运行该程序，则出现如图 2.13 所示的窗体，单击【欢迎】按钮，则窗体如图 2.14 所示。

上述小程序是在 VB 环境中运行的，为使程序能脱离 VB 环境而独立运行，还需要将它编译成可执行的 EXE 文件。VB 中提供了专门用于生成可执行文件的命令，打开【文件】菜单，执行【生成 EXE】

命令，则弹出如图 2.19 所示的【生成工程】对话框，选择要保存文件的位置，输入可执行文件的文件名，然后单击【确定】按钮即可生成程序的可执行文件。



图 2.19 【生成工程】对话框

通过"我的电脑"或"资源管理器"在硬盘中找到生成了的可执行文件，双击该文件，即可启动应用程序了，而不再需要 VB 环境的支持。

从上述的一个简单应用程序的创建过程中，可以得出使用 V\$ 创建应用程序的基本步骤如下：

设计用户界面。

设置对象属性。

编写代码。

保存程序。

调试并运行程序。

发布应用程序。

VB6.0 基础教程 2.5 工程的管理

使用 Visual Basic 6.0 可以创建标准的 Windows 应用程序、ActiveX 与 Active 文档等，在设计阶段，VB 通称它们为一个工程。上面所创

建的简单应用程序，就是一个工程，该工程只包含一个窗体。但对于一些较复杂的应用程序，工程中一般都包括一个或若干个窗体、模块等文件。这就涉及到工程与文件的新建、保存、移除等多种操作。这些操作贯穿到创建应用程序的整个过程。

VB6.0 基础教程 2.5.1 工程资源管理器

工程资源管理器是用来管理工程的，它的功能就像 Windows 中的资源管理器一样。执行【视图】菜单中【工程资源管理器】命令或单击工具栏中的【工程资源管理器】按钮均可打开工程资源管理器。

在工程资源管理器中，列出了当前用户所创建的所有工程，并且以树状的形式显示了每个工程的组成。在启动 VB 后，工程资源管理器如图 2.20 所示，从中可以看出当前只有一个过程，工程中包含一个窗体。单击工程名前的一号节点或+号节点可以折叠或展开工程。如图 2-21 所示的是将工程折叠起来后的效果。

如图 2.22 所示的是在 VB 中同时创建多个工程，并且工程中有多个文件时的工程资源管理器，可以看出，当前新建（或打开）了两个工程，其中工程 1 包含有两个窗体，工程 2 只包含一个窗体。两个工程又组成一个工程组。



图 2.20 工程资源管理器 图 2.21 工程折叠起来后的效果 图 2.22 工程资源管理器

在工程资源管理器中，显示有工程名、工程文件名、窗体名和窗体文件名，如图 2.23 所示。请读者注意区别它们。



图 2.23 工程资源管理器中的名称

工程文件名与窗体文件名是用户在保存工程时为工程与窗体指定的名称。窗体名是指用户在【属性】窗口中为窗体设置的"名称"属性。工程名为 VB 对用户所创建的应用程序的标识。

在默认情况下，工程名为工程 1、工程 2 等。用户也可以指定工程的名称。执行【工程】菜单中的【工程 x 属性】命令，打开如图 2.24 所示【工程属性】对话框，在其中的【工程名称】文本框中输入工程的新名称，单击【确定】按钮即可。



图 2.24 【工程属性】对话框

资源管理器左上角的按钮是【查看代码】按钮，按钮是【查看对象】按钮，单击它们可以分别打开所选窗体的【代码】窗口和【窗体设计】窗口，双击窗体的名称也可以打开其对应的【窗体设计】窗口。

VB6.0 基础教程 2.5.2 创建和打开工程

在启动 visual Basic 6.0 时，系统会弹出【新建工程】对话框，在其中选择一种工程类型，单击【打开】按钮后，就创建了一个新的工程，且该工程中只包括一个窗体。该工程的默认名为"工程 1"窗体默认名为"Form1"。在保存工程时，用户可为工程以及工程中的窗体、模块等指定名称。

使用【文件】菜单中的【新建工程】命令也可以新建工程，执行该命令后，弹出如图 2.25 所示的【新建工程】对话框，在其中选择一种工程类型，单击【确定】按钮即可。使用该命令来新建工程，原有的工程将被移除，如果原有的工程没有保存，则系统还会弹出对话框提示用户保存工程。

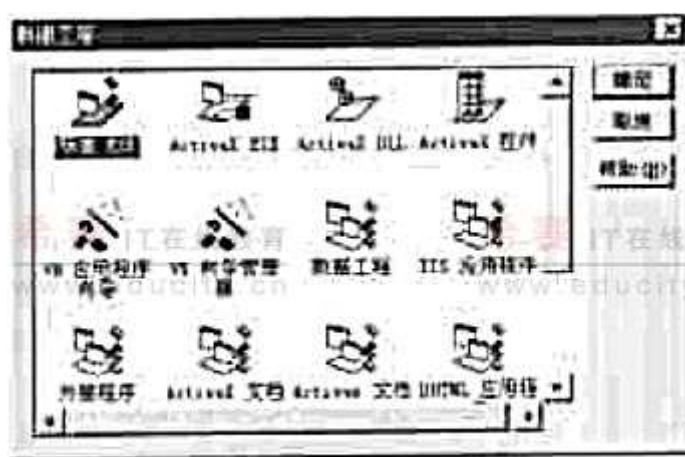


图 2.25 【新建工程】对话框

在 Visual Basic 6.0 中也可以同时建立多个工程，执行【文件】菜单中的【添加工程】命令，打开如图 2.25 所示的【添加工程】对话框，该对话框有 3 个选项卡，【新建】选项卡用来添加一个新的工程，【现存】与【最新】选项卡则用来添加一个已存在的工程。打开【现存】选项卡，如图 2.27 所示，该选项卡与打开文件对话框类似，用户可从中选择要添加的工程。打开【最新】选项卡，如图 2.28 所示，

该选项卡中列出了若干个最近编辑过的工程。在工程列表中选中一个工程，单击【打开】按钮即可将其添加到 VB 中。

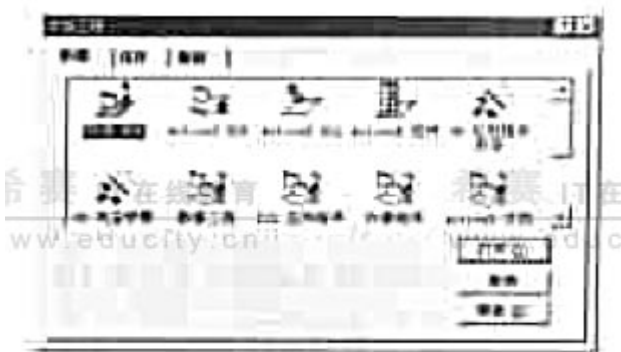


图 2.26 【添加工程】对话框



图 2.27 【最近】选项卡



图 2.28 【最新】选项卡

VB6.0 基础教程 2.5.3 保存和移除工程

在创建工程后，只有将其保存到硬盘上，才能在下次继续使用该工程。我们已经知道，一个工程包含一个或多个窗体与模块等文件。在保存工程时，这些窗体或模块文件也将一同保存。

执行【文件】菜单中的保存工程]命令，则打开如图 2.29 所示的【文件另存为】对话框。在【文件名】文本框中输入窗体或模块的名称，单击【保存】按钮。如果工程中包含有多个窗体或模块等，则【文件另存为】对话框仍然存在，要求用户保存下一个文件，直到工程中所有的文件保存完毕。最后，出现如图 2.30 所示的【工程另存为】对话框，在【文件名】文本框中输入工程的名称，单击【保存】按钮。这样，就完成了对一个工程的保存。



图 2.29 【文件另存为】对话框

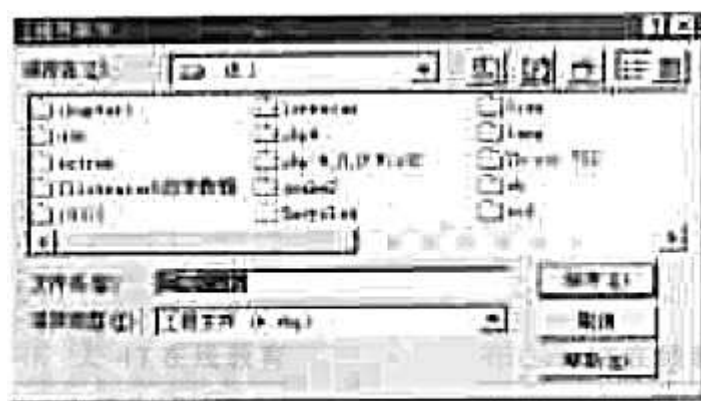


图 2.30 【工程另存为】对话框

工程中的各文件以及工程本身既可以保存在同一位置，也可以保存在不同的位置，在工程本身的文件中包含有它的窗体等文件的路径、名称等信息，在打开工程时，工程文件会以这些信息去寻找它的

窗体等文件。如果窗体等文件的路径或名称改变了，则会出现加载错误。

提示：由于一个工程的各个部分是紧密联系的。任何一个部分出现错误都会导致整个工程发生错误。因此，为了便于管理，建议读者尽量将一个工程中的各文件保存在同一位置。

一个工程创建完成后，如果想关闭该工程，而不退出 VB 环境，则可以使用【文件】菜单中的【移除工程】命令将该工程移除。如果当前只有一个工程，直接执行【文件】菜单中的【移除工程】命令即可将其移除；如果当前有多个工程，需要事先在工程资源管理器中选择要移除的工程，然后执行【移除工程】命令即可将指定的工程移除。

也可通过工程资源管理器来保存或移除工程。将鼠标移动到要保存或移除的工程名上，单击右键，在弹出的快捷菜单中执行【移除工程】命令即可。

注意：移除工程只是将工程从 VB 开发环境中移除出去，而不是将其从硬盘中删除。

VB6.0 基础教程 2.5.4 添加、删除和保存文件

新建一个工程后，在工程资源管理器中可以看出它只包含一个窗体。上面提到，一个工程一般包含有多个窗体或模块，如常用的 Word 等软件，都是包含有多个窗体的应用程序。在工程新建后，可以根据需要为工程添加多个窗体以及模块等文件。

打开【工程】菜单，或单击工具栏中【添加窗体】按钮右边的下三角形，在其下拉菜单中选择要添加的文件类型即可：



图 2.31 【添加文件】对话框



图 2.32 添加新窗体后的工程资源管理器

不同的工程可以共享同一个窗体、模块等文件。可以将在其他上程中创建的文件添加到新的工程中。执行【工程】菜单中的【添加文件】命令，出现如图 2.31 所示的【添加文件】对话框，在其中选择一个文件，单击【打开】按钮即可将其添加到工程中。

这里，我们来做做一个为工程添加窗体文件的实例。打开【工程】菜单，执行【添加窗体】命令；再次打开【工程】菜单，执行【添加文件】命令，在打开的【添加文件 1 对话框中选择一个已存在的窗体文件。此时的工程资源管理器如图 2.32 所示。可以看出，我们为工程 1 新建了一个名称为 form2 的新窗体。同时，窗体文件的"封面"也被加载到了工程 1 中。

对于工程中不再需要的文件，用户还可以将其从工程中移除出去。在工程资源管理器中选中要移除的文件，然后打开【工程】菜单，执行其中的【移除】命令即可将其从工程中移除掉。也可以在资源管理器中将鼠标移动到要移除的文件上，单击右键，在弹出的快捷菜单中执行【移除】命令即可。

在保存工程时，会同时保存工程中的窗体、模块等文件，还可以单独保存这些文件。在工程资源管理器中选中要保存的文件，然后打开【文件】菜单，执行其中的【保存】命令即可将该文件保存。

第三章 Visual Basic 语言基础

VB6.0 基础教程 3.1.1 注释

注释是指在编写代码时，编写者在代码中添加的一些说明性语句。注释是非执行语句，只是对有关的内容加以说明。例如，说明某个过程的功能，定义某个变量的目的等。

在程序中添加注释是个良好的编程习惯。每个程序员都有这样的体会：自己编写的一些代码，过一段时间后再去阅读，会感到很费劲甚至难以读懂。阅读别人编写的代码更是一可想而知。因此，在编写程序代码时，最好为代码添加注释，以便于自己或别人能较轻松地读懂代码。

在 VB 中，注释以 **Rem** 关键字开头，并且 **Rem** 关键字与注释内容之间要加一个空格。注释可以是单独的一行，也可以写在其他语句行的后面。如果在其他语句行后使用 **Rem** 关键字，则必须使用冒号 (:) 与语句隔开。

也可以使用一个撇号 (') 来代替 **Rem** 关键字。若使用撇号，则在其他语句行使用时不必加冒号。如图 3.1 所示的是注释的写法。在【代码】窗口中，注释添加完成后，自动以绿色显示。



图 3.1 注释语句

在 VB 的【编辑】工具栏中还提供了专门用于设置注释块的按钮，使得将多行语句设置为注释或取消注释十分方便。在默认情况下，【编辑】工具栏不出现在界面中，在【视图】菜单的【工具栏】子菜单中选择【编辑】选项即可打开【编辑】工具栏。

设置注释块的操作是：在【代码】窗口中选中要设置为注释的单行或多行语句，单击【编辑】工具栏中的【设置注释块】按钮，即可为所选的每行语句前添加一个撇号（'），将代码设置为注释。单击【解除注释块】按钮，则情况相反。

VB6.0 基础教程 3.1.2 分行与续行

有时候，一条语句可能会很长，如果将它写在一行上，将给阅读或打印代码带来不便。在 VB 中，可以使用分行符（_）（一个空格和一条下划线）将一条语句写在两行或多行上。

例如：

```
Form1.Caption="分行".
```

语句可以分为两行写成：

```
Form1.Caption=_ "分行".
```

在同一行内，分行符后面不能添加注释，例如下面的语句是错误的：

```
Fonm1.Caption=_ '设置窗体标题。
```

```
"分行".
```

分行符一般添加在运算符的前后。不能使用分行符将一个变量名或属性名分隔成两行，例如下面的语句是错误的：

Form1.Capt_.

ion="分行".

在通常情况下，一条语句占用一行，并且在语句末尾没有表示语句结束的符号。但也可以将多行语句写在同一行上，相邻的两条语句中间使用冒一号（:）作为续行符。

例如：

Form1.Caption="断行":Form1.FontSize=14: Print"你好".

在打印程序时，将多行语句写在同一行上可以节省纸张。但为了便于程序的阅读，最好在一行上只写一条语句。

VB6.0 基础教程 3.2.1 变量的命名规则

不同的变量是通过变量名标识的。在命名变量时，有很大的灵活性，例如，可以将用来保存产品价格的变量命名为 X,也可以将其命名为 Price 或其他名称。

在较大型的程序中，最好用带有一定描述性的名称来命名对象，如将表示价格的变量命名为 Price ,将表示年龄的变量命名为 Age 等，这样会使得程序易于阅读与维护。

在 VB 中，变量的命名还需要遵循以下几条规则：

变量名必须以字母或汉字开头。例如，abc.姓名、年 n3 和 ff28 等变量名都是合法的，而 3abc.#xy 和+uu 等变量名是非法的。

不能在变量名中出现句号、空格或者嵌入！、@、#、\$、%、& 等字符。例如，#, d%等变量名是合法的，而 r%R, a#bc 和 a be 等变量名是非法的。

不能使用 VB 的关键字作为变量的名字。关键字是 VB 内部使用的词，是该语言的组成部分。例如，`print`, `dim` 和 `For` 等都是非法变量名。

变量名不得超过 255 个字符。

变量名在变量的有效范围内必须是唯一的。

变量名不区分大小写。例如，变量 `ABC`, `Abc` 和 `aBc` 表示同一变量。

VB6.0 基础教程 3.2.2 变量的数据类型

除名称外，变量还有数据类型。变量的数据类型决定了如何将变量的值存储到计算机的内存中。所有的变量都具有数据类型，以决定它能够存储哪种类型的数据。例如，某个变量的数据类型为整型（存放整数），但是如果在代码中将一个字符串赋给它，则运行程序时会出现类型不匹配的错误，并弹出消息框，提示用户类型不匹配。在声明变量时可指定它的数据类型。

表 3.2 中列出了 VB 所支持的基本数据类型。

表 3.1 VB 的基本数据类型

数据类型		占用字节	类型符	取值范围说明
Numeric (数值型)	Byte (字节型)	1		0~255
	Integer (整型)	2	%	-32768~32767
	Long (长整型)	4	&	-2147483648~2147483647
	Single (单精度浮点型)	4	!	-3.402823E38~3.402823E38
	Double (双精度浮点型)	8	#	-1.79769313486232D308~1.79769313486232D308
	Currency (货币型)	8	@	-9222337203685477.58078~9222337203685477.5807
String (字符串型)	String	1/字符	\$	0~65535 个字符
Date (日期型)		8		公元 100 年 1 月 1 日到公元 9999 年 12 月 31 日

(续表)

数据类型	占用字节	类型符	取值范围说明
Boolean (布尔型)		2	True 或 False
Variant (变体型)			根据需要分配

1. 数值型数据类型

VB 支持 6 种数值型数据类型，分别是 Byte (字节型)、Integer (整型)、Long (长整型)、Single (单精度浮点型)、Double (双精度浮点型) 和 Currency (货币型)。

如果知道变量总是存放整数 (如 12) 而不是带小数点的数字 (如 3.57), 就应当将它声明为 Integer 类型或 Long 类型。整数的运算速度较快, 而且比其他数据类型占据的内存要少。在 For...Next 循环内作为计数器变量使用时, 整数类型尤为高效。

如果变量包含小数, 则可将它们声明为 Single, Double 或 Currency 变量。

浮点数值可表示为 mmmEeee 或 mmmDeee 形式。其中 mmm 是假数, 而 eee 是指数 (以 10 为底的幂)。Single 数据类型的最大正数值为 3.402823E+38, 即 3.4 乘以 10 的 38 次方; Double 数据类型的最

大正数值是 1.797b9313486232D+308 即 1.8 乘以 10 的 308 次方。用 D 将数值文字中的假数部分和指数部分隔开，就会导致将该值作为 Double 数据类型来处理。同样，用这种方式使用 E,也会导致将该值作为 Sings.数据类型来处理。

Currency 数据类型的定点实数保留小数点右面 4 位和小数点左面 15 位，适用于货币计算。浮点（Single 和 Double）数比 Currency 的有效范围人得多，但有可能产生小的进位误差。

Byte 数据类型主要用于存储二进制数。

所有数值变量都可相互赋值。在将浮点数赋予整数之前，VB 会将浮点数的小数部分四舍五入，而不是将小数部分去掉。

2. 字符串型数据类型

如果变量总是包含字符串而从不包含数值，就可将其声明为字符串型。字符串是用双引一号括起来的若干个字符。字符串中的字符可以是计算机系统允许使用的任意字符。例如："Visual Basic 6.0"、"***计算机%%%"和"xxx@xxx.net"

等都是合法的字符串。字符串的长度是指字符串中字符的个数，不含任柯字符的字符串为空字符串。字符串在内存中是按字符连续存储的，每个字符占用一个字节。例如字符串"1234"占用 4 个字节的内存。而 1234 是一个整型数据，它占用两个字节。

如果字符串表示数值，则可将字符串赋予数值变量。也可将数值赋予字符串变量。

3. 日期型数据类型

日期型变量用来存储日期或时间。可以表示的日期范围为从公元 100 年 1 月 1 日到公元 9999 年 12 月 31 日，时间则是从 0:00:00 到 23:59:59。日期常数必须用"#"号括起来。例如，如果变量 **Mydate** 是一个日期型变量，可以使用下面几种方式为该变量赋值：

Mydate=#3/19/1977

Mydate=1977-04-19#

Mydate=#77.3,19#

Mydate=#April 19,1944#

Mydate=#19 Apr 77#

上面几个语句的作用完全相同，都是将日期常数 1977 年 4 月 19 日赋给日期变量 **Mydate**。并且，在【代码】窗口中输入上述任一语句，VB 都将自动将其转换为第一条语句的格式，即 **Mydate=#4/19/1977#**。

如果将一个数值型数据赋给日期变量，则小数点左边的数字代表日期，右边的数字代表时间；0 为午夜，0.5 为中午 12 点；负数代表的是 1899 年 12 月 31 日之前的日期或时间。例如，从现在起过 6 小时的日期和时间可以表示为：

now+0.25

4. 布尔型数据类型

若变量的值只是"true/false"、"yes/no"、"on/off"等信息，则可将它声明为布尔类型。布尔类型的缺省值为 **False**。

5. 变体型数据类型

数据类型为变体型的变量可以存储所有系统定义类型的数据。变体型数据类型可在不同场合代表不同的数据类型。如果把数据赋予变体变量，则不必在这些数据的类型间进行转换：Visual Basic 会自动完成任何必要的转换，例如，如果变量 `Myvariant` 为变体型变量，下面是该变量的几条赋值语句以及说明：

`Myvariant="15"` 变量值包含两个字符的字符串

`Myvariant=17-Myvariant` 此时变量值为数值 2

`Myvariant="A"& Myvariant` 此时变量值为字符串 "A2"

VB6.0 基础教程 3.2.3 变量的声明

在使用变量前，一般要先声明变量名及其类型，以决定系统为变量分配的存储单元。在 VB 中可以通过以下几种方式来声明变量及其类型；

1.使用 Dim 语句

使用 Dim 语句声明变量的一般形式如下：

`Dim 变量名 AS 数据类型。`

例如：

`Dim Nuber As Integer`

`Dim Count As Single`

`Dim Name as String`

也可以使用数据类型的类型符来替代 `As` 子句。例如，上述 3 个声明语句也可写成：

`Dim Number%`

Dim Count!

Dim Name\$

注意：变量名与类型符之间不能有空格

一条 **Dim** 语句也可以声明多个变量，每个变量都需要有自己的声明类型，并且各变量之间以逗号隔开。例如，可以将上面的 3 条语句改写成一条语句：

Dim Number As Integer, Count As Single, Name As String

如果忽略了 **Dim** 语句中的 **As** 子句，则 **VB** 将变量的类型认为是变体型。

例如下面语句声明的 **Myv** 变量的数据类型是变体型：

Dim Myv

在默认情况下，字符串变量是不定长的，随着对字符串变量赋予新的数据，它的长度可增可减，也可以将字符串变量声明为定长的。声明一个定长字符串变量的语法如下：

Dim 变量名 As String*长度

例如，声明一个长度为 50 个字符的字符串变量，可用下列语句：

Dim Name As String*50

如果赋给该定长字符串变量的字符少于 50 个，则用空格将 **Name** 变量的不足部分填满。因为定长字符串用空格填充尾部多余的空间，所以在处理定长字符串时可发现，删除空格的 **Trim** 和 **Rtrim** 函数是很有用的。如果赋给的字符串的长度大于 50，则 **VB** 会自动截去超出部分的字符。

例如，编写窗体的 Click 事件过程如下：

```
Private Sub From_Click ()  
  
Dim str1 As String  
  
Dim str2 As String * 4  
  
Dim str3 As String * 2  
  
Str1="中华人民共和国"  
  
Str2="中华人民共和国"  
  
Str3="中华人民共和国"  
  
Print str1  
  
Print str2  
  
Print str3  
  
End Sub
```

在该段代码中，声明了 3 个字符串变量。其中 Str1 为不定长字符串变量，Str2 和 Str3 为定长字符串变量，并且长度分别为 4 和 2。为这 3 个字符串变量赋予相同的值"中华人民共和国"。然后使用 Print 语句在窗体上分别打印出各字符串变量。

运行该程序，单击窗体，则窗体上打印出 3 行文字，如图 3.3 所示。由于字符串变量 Str2 最多能存储 4 个字符，因此，字符串"中华人民共和国"的后 3 个字符被截去。同理，Str3 中只存储了两个字符，其他字符被截去。



图 3.3 在窗体上显示字符串

2. 隐式声明

在 VB 中，也可以不事先声明而直接使用变量，这种方式称为隐式声明。上述使用 Dim 语句声明变量的方式称为显式声明。所有隐式声明的变量都是变体型数据类型。

在使用一个变量之前并不必先声明这个变量。例如，不必在使用变量 TempVal 之前先声明它：

```
Function safeSqr (num)
```

```
TempVal=Abs (num)
```

```
SsfeSqr=Sqr (TempVal)
```

```
End Function
```

VB 用这个名字自动创建一个变量，使用这个变量时，可以认为它就是显式声明的。虽然这种方法很方便，但是如果把变量名拼错了的话，系统会认为它是另一个新的变量，从而会导致一个难以查找的错误。

如果知道变量确实总是存储特定类型的数据，最好还是先声明变量的数据类型，这样 VB 会以更高的效率处理这个数据。例如，存储人名的变量最好声明成字符串数据类型，因为名字总是由字符组成的。

为了避免写错变量名引起的麻烦，可以在【代码】窗口的声明段中加入语句：

Option Explicit

这样，在代码中只要遇到一个未经显式声明就当成变量的名字，Visual Basic 都会弹出错误警告。

例如，编写一段代码如图 3.4 所示，其中声明了变量 s1,而没有声明变量 s2,运行该程序，单击窗体，则弹出编译错误消息框，提示用户变量未定义。如果删除 option Explicit 语句，再次执行程序，则不会再出现变量未定义的错误。

也可以将系统定制为总要求显式声明变量。执行【工具】菜单中的【选项】命令，打开如图 3.5 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。这样就在任何新建的模块中自动插入如 Option Explicit 语句，但不会在已经建立起来的模块中自动插入。所以在工程内部，只能用手工方法向现有模块添加 Option Explicit 语句。



图 3.4 【代码】窗口



图 3.5 【选项】对话框

注意：Option Explicit 语句的作用范围仅限于语句所在模块，所以，对每个需要强制式变量声明的窗体模块和标准模块，都必须将 Option Explicit 语句放在它们各自的声明段中。

VB6.0 基础教程 3.2.4 变量的作用域

一个变量声明后，并不是在任何地方都能使用它。每个变量都有它的作用域。变量的作用域决定了哪些子过程和函数过程可使用该变量。变量的声明方式和声明位置决定了它的作用域。

在理解变量的作用域之前，首先需要了解一个应用程序的组成。一般应用程序的组成如图 3.6 所示。



图 3.6 应用程序的组成

关于在工程中添加窗体与模块的操作在第 2 章中已经介绍过了。本书只涉及到窗体模块和标准模块。标准模块主要用来定义一些公用的变量和过程，以供各窗体模块中的事件过程引用。

变量的作用域可分为 3 个层次：局部变量、模块级变量和全局变量。表 3.2 中列出了变量的作用范围及使用规则。

注意：如不加特别说明，模块是指对窗体模块与标准模块的统称。

表 3.2 变量的作用范围

变量作用域	声明方式	声明位置	被本模块访问	被其他模块访问
局部变量	Dim 或 Static	在过程中	不能	不能
模块级变量	Dim 或 Private	模块的通用声明段	能	不能
全局变量	Public	模块的通用声明段	能	能，如果是在窗体模块中定义的，则需要加窗体名

1.局部变量。

局部变量是指在过程内部使用 Dim 语句或 Static 语句声明的变量。在过程内不加声明而直接使用的变量也是局部变量。我们知道，一个应用程序包含若干个模块，模块中又包含若干个过程。对于局部变量，只能在声明它的过程中使用，本模块的其他过程以及其他模块均不可访问。

在不同的过程中可以声明相同名称的变量，它们相互独立，互不干扰。

2.模块级变量。

模块级变量是指在模块的任何过程之外，即在模块的声明部分使用 Dim 语句或 Private 语句声明的变量。可被本模块的任何过程访问。

3.全局变量。

全局变量是指在模块的任何过程之外，即在模块的"通用声明"段使用 Public 语句声明的变量。可被本模块的任何过程访问。需要注意的是。在窗体模块声明的全局变量，在访问时需要在变量名前加窗体名。而在标准模块中声明的全局变量可以直接访问。

例如，在窗体模块中定义了四个变量 A、B、C 和 D,如图 3.7 示。
则 A 为全局变量，B 为模块级变量，C 和 D 为局部变量。

需要说明的是，在 VB 中作用域不同的变量的名称可以相同，并且作用域小的变量的优先级高。



图 3.7 定义作用域不同的变量

VB6.0 基础教程 3.2.5 静态变量

在过程中，既可以使用 Dim 语句声明局部变量，也可以使用 Static 语句声明局部变量。并且 Static 语句的一般形式与 Dim 语句相同：

Static 变量名 As 数据类型。

使用 Static 语句声明的变量称为静态变量，它与用 Dim 语句声明的变量的不同之处在于：当一个过程结束时，过程中所用到的静态变量的值会保留，下次再调用此过程时，变量的初值是上次调用结束时被保留的值。

对于使用 Dim 语句声明的局部变量，随过程的调用而分配存储单元，并进行变量的初始化。一旦过程结束，变量的内容自动消失，占用的存储单元也被释放。因此，每次调用过程时，变量都将重新初始化。

下面，用一个实例来说明静态变量的特点。

实例 3.1 静态变量

在【代码】窗口中编写窗体的 Click 事件过程如下：

```
Private Sub Form_Click ()  
  
Dim Sum As Integer  
  
Print Sum  
  
Sum= Sum +1  
  
End Sub
```

运行程序，在窗体上单击数次，窗体上显示的数字始终是 0,如图 3.8 所示。

将上述代码中的 Dim 替换成 Static,如下所示：

```
Private Sub Form_Click ()  
  
Static Sum As Integer  
  
Print Sum  
  
Sum=Sum + 1  
  
End Sub
```

再次运行程序，则每单击一次窗体，显示的数字加 1,如图 3.9 所示。



图 3.8 使用 Dim 语句



图 3.9 使用 Static 语句

VB6.0 基础教程 3.3 常量

在程序执行过程中数值始终不改变的变量称为常量。例如，如果要进行数学计算，则程序中可能多次出现数值 pi (3.14159.....), 如果将这个值使用一个常量 pi 来表示，在程序中就可以使用常量 pi 来替代常数 3.14159,而不必一遍遍地输入 3.14159.

定义常量的形式如下：

Const<常量名>[**As** 类型]=常量值常量的命名规则和变量一样。

As 子句是可选的，它用来说明常量的数据类型，如果省略，则数据类型由表达式决定。常量值可以是数字、字符串或由它们与运算符组合成的简单表达式。

例如：

Const pi As Double=3.14159265358979

Const Str="ABCDEF"

Const Str= (2+3) * 7

常量声明中不能使用函数，例如：

Const Num=Sin (30) .

语句是错误的。

常量声明语句中可以包含其他常量。例如，在数字计算中，数值 `pi` 和数值 `2*pi` 样常用，可以将这两个值都声明为常量如下：

```
Const pi As Double=3.14159265358979
```

```
Const pi2 As Double=2*pi
```

一旦声明了常量，就不能在此后的语句中改变它的数值，这是个安全特性，也是声明常量的一个好处。例如，如果在程序"1"使用赋值语句来给常量赋值，编译程序将产生错误，并弹出消息框提示用户不允许给常量赋值：

常量也有作用范围的概念，这一点与变量相同。例如常量 `pi` 通常在模块中声明为：

```
Public Const pi As Double=3.1415926358979
```

以便每个过程都能访问它。

VB 自身还定义了大量的内部常量。例如，复选框控件的 `Value` 属性的值可以为 0（取消）、1（选定）或 2（变灰）。可以不用下列语句：

```
Check1.Value=0.
```

```
Check2.Value=2.
```

而使用内部常量 `vbUnchecked` 和 `vbGrayed` 来代替 0 和 2:

```
Check1.Value=vbGrayed.
```

```
Check2.Value=vbGrayed.
```

常量 `vbUnchecked` 和 `YbGrayed` 是 VB 语言固有的，无需声明，其符号化的名称使程序更容易阅读和维护。Visuat Basic 固有常量均

用前缀 **vb** 表示，声明自己的常量时不要用这个前缀。其他构件用其他的前缀，例如，数据库访问对象用前缀为 **db** 的常量。

VB6.0 基础教程 3.4.1 算术运算符

程序中对数据的操作，其实就是指对数据的各种运算。被运算的对象，如常数、常量和变量等称为操作数。运算符是用来对操作数进行各种运算的操作符号，如加号（+）、减号（-）等。诸多操作数通过运算符连成一个整体后，就成为一表达式。

VB 中具有丰富的运算符，可分为算术运算符、关系运算符、逻辑运算符和字符串运算符 4 种。

算术运算符用来进行算术运算。**VB** 提供的算术运算符如表 3.3 所示。

表 3.3 算术运算符				
算符	含义	优先级	举例	结果
+	加	6	X=3+2	5
-	减	6	X=7-4	3
-	取负	2	X=-10	-10
*	乘	3	X=3*7	21
/	除	3	X=7/2	3.5
\	整除	3	X=7/2	3
Mod	求余	5	7 Mod 2	1
^	指数	1	2 ^ 3	8

其中取负运算符（-）只需一个操作数，称之为单目运算符。其他运算符都需要两个操作数，称之为双目运算符。

运算符的优先级表示当表达式中有多个操作符时，先执行哪个操作符。

整除运算（\）的结果是商的整数部分。例如，7\2 表示整除，商为 4.5,结果取整数部分 3,不进行四舍五入。如果参加整除的操作数是浮点数，则先按四舍五入的原则将它们变成整数，然后再执行整除运

算。例如，对于 $8.5 \setminus 2$ ，先将 8.5 变成 9 再进行整除，商为 4.5，结果为 4。

取余运算（Mod）是求两个整数相除后的余数。如果参加取余运算的操作数是浮点数，则先按四舍五入的原则将它们变成整数，然后再执行取余运算。例如，对于 $8.5 \setminus 2.1$ ，先将 8.5 变成 9，2.1 变成 2，然后 9 除以 2 与 1，因此取余结果为 1。

VB6.0 基础教程 3.4.2 关系运算符

关系运算符用来对两个操作数进行大小比较。关系运算的结果是一个逻辑量，True（真）或 False（假）。如果关系成立，则值为 True，否则值为 False。在 VB 中，True 用 -1 表示，False 用 0 表示。VB 中有 6 种关系运算符，如表 3.4 所示。

表 3.4 关系运算符

运算符	含义	举例	结果
=	等于	"a" = "A"	False
>	大于	"abc" > "aBc"	True
>=	大于等于	8 >= 7	True
<	小于	8 < 7	False
<=	小于等于	23 <= 23	True
<>	不等于	"a" <> "A"	True

用来比较的操作数可以是数值型，也可以是字符串型。数值以大小进行比较是显然的。字符串的比较是按照字符的 ASCII 码值的大小来比较的。即首先比较两个字符串第一个字符，ASCII 码值大的字符串大。如果第一个字符相同，则比较第二个字符，依次类推。例如，由于小写字母的 ASCII 码大，因此关系表达式 "abc" > "abc" 的值为 True。关于字符的 ASCII 码对照表，读者可参见本书下一章。

VB6.0 基础教程 3.4.3 逻辑运算符

逻辑运算符的作用是对操作数进行逻辑运算。操作数可以是逻辑量（True 或 False）或关系表达式。逻辑运算的结果也是一个逻辑量。表 3.5 中列出了 VB 中的 6 种逻辑运算符。

表 3.5 逻辑运算符

运算符	含义	说明	优先级	举例	结果
Not	取反	若操作数为假，则结果为真，若操作数为真，则结果为假	1	Not ("a" = "A")	True
And	与	操作数均为真时，结果才为真	2	(2>1) And (7<3)	True
Or	或	操作数有一个为真时，结果就为真	3	("a" = "A") Or (2>1)	True
Xor	异或	操作数相反时，结果才为真	4	(2>1) Xor (7<3)	False
Eqv	等价	操作数相同时，结果才为真	5	(2>1) Eqv (7<3)	True
Imp	蕴含	第一个操作数为真，第二个操作数为假时，结果才为假，其余情况下结果均为真	6	(8=8) Imp False	False

VB6.0 基础教程 3.4.4 字符串运算符

字符串运算符有两个："&"和"+"，它们的作用是将两个字符串拼接起来。

例如：

"Visaul Basic"&"程序设计语言结果为" 结果为"Visaul Basic 程序设计语言"

"电脑"+"爱好者" 结果为"电脑爱好者"

StrS="计算机"

str&"与网络" 结果为"计算机与网络"

注意：变量名与&之间一定要加一个空格。因为&本身还是长整型的类型符，不加空格容易造成误会。

"&"运算符会自动将非字符型的数据转换成字符串后再进行连接，例如：

1234&5678&"abcd" 结果为"12345678abcd"

"+"运算符在连接字符串时不能自动转换，例如下面语句在运行时将出现类型不匹配错误：

```
1234+"abcd"
```

VB6.0 基础教程 3.4.5 优先级

在一个表达式中进行多个运算时，每一部分都会按预先确定的顺序进行计算求解，这个顺序被称为运算符优先级。

当表达式有多种运算符时，先处理算术运算符和字符串运算符，接着处理关系运算符，然后再处理逻辑运算符。即各种运算符的优先级如下：

算术运算符>字符串运算符>关系运算符>逻辑运算符。

所有比较运算符有相同的优先级，即按它们出现的顺序从左到右进行处理。算术运算符和逻辑运算符按它们各自的优先级进行处理。当乘法和除法同时出现在表达式中时，按照从左到右出现的顺序处理每个运算符。同样，当加法和减法同时出现在表达式中时，也按照从左到右出现的顺序处理每个运算符。

括号可改变优先级的顺序，强制优先处理表达式的某部分。括号内的操作总是比括号外的操作先被执行。但是在括号内，仍保持正常的运算符优先级。

在书写表达式时，尽管有时候括号不是必须的，但最好还是在表达式适当的地方添加一些括号，使得表达式的层次更分明，以增加程序的可读性。

例如，选拔模特的基本标准是身高（T）要在 175 公分与 185 公分之间，同时，体重（W）要小于 56 公斤。不过，如果文化课成绩（S）在 90 分以上者。即使身高与体重不合格也可以破格录取。

描述以上选拔条件的表达式可以写成如下的形式：

$175 \leq T \text{ And } T \leq 185 \text{ And } W < 56 \text{ Or } S > 90$

但如果适当地加上一些括号，则表达式的层次就一目了然了，如下所示：

$((175 \leq T) \text{ And } (T \leq 185) \text{ And } (W < 56)) \text{ Or } (S > 90)$

VB6.0 基础教程 3.5.1 数学函数

数学函数用来完成一些基本的数学计算，其中一些函数的名称一与数学中相应函数的名称相同。表 3.6 中列出了常用的数学函数。

表 3.6 常用数学函数

函数	说明	举例	结果
Abs(n)	返回参数的绝对值	Abs(-6.5)	6.5
Atn(n)	返回参数的反正切值	Atn(0)	0
Cos(n)	返回参数的余弦值	Cos(0)	1
Exp(n)	返回 e（自然对数的底）的某次方	Exp(2)	7.389
Fix(n)	返回参数的整数部分	Fix(8.2)	8
Int(n)	返回参数的整数部分	Int(-8.4)	-9
Log(n)	返回参数的自然对数值	Log(10)	2.3
Rnd(n)	返回一个随机数值	Rnd	0~1 之间的某数
Sgn(n)	返回参数的正负号	Sgn(-5)	-1
Sin(n)	返回参数的正弦值	Sin(0)	0
Sqr(n)	返回参数的平方根	Sqr(25)	5
Tan(n)	返回参数的正切值	Tan(0)	0

在三角函数中，参数以弧度表示。例如，函数 Sint（30）中的 30 是指弧度，它等于 1718.87,而不是 30 度。为了将角度转换成弧度，可以将角度乘以 $\pi/180$.若将弧度转换成角度，则将弧度乘以 $180/\pi$. 其中 π 是数学常数，近似值为 3.1415926535897932.

Int 函数和 **Fix** 函数的不同之处在于，如果参数 **n** 为负数，则 **Int** 返回小于或等于该参数的第一个负整数，而 **Fix** 则会返回大于或等于参数的第一个负整数。

例如，**Int** (-8.4)=-9,而 **Fix** (-8.4)=-8.

函数 **Sgn** 将根据参数 **n** 的不同取值，返回不同的值。若 **n**>0,则 **Sgn** (**n**) =1;若 **n**=0,则 **Sgn** (**n**) =0;若 **n**<0,则 **Sgn** (**n**) =-1.

随机函数 **Rnd** (**n**) 返回一个介于 0~1 之间（包括 0,但不包括 1）的单精度随机数。参数 **n** 的值决定了 **Rnd** 生成随机数的方式：

如果 **n**<0,则根据 **n** 的值，返回一个特定的随机数。

如果 **n**>0 或省略，则返回随机序列中的下一个随机数。

如果 **n**=0.则返回与上一次产生的相同的随机数。

Rnd 函数所产生的随机数的序列取决于"种子"的初始值。对最初给定的种子都会生成相同的序列，因为每一次调用 **Rnd** 函数都用数列中的前一个数作为下一个数的种了。

使用 **Randomize** (**number**) 语句可初值化随机数生成器，将"种子"的初值指定为参数 **number** 的值。

为了使每次调用 **Rnd** 函数能产生不同的随机序列，在调用 **Rnd** 之前，可先使用无参数的 **Randomize** 语句初始化随机数生成器，这样，随机数生成器具有根据系统计时器得到的种子。由于计时器不断变化，因此种子也就在不断地变化。通常将 **Randomize** 语句放在窗体的 **Load** 事件过程中。

为了生成某个范围内的随机整数，可使用以下公式：

$\text{Int}((\text{upperbound}-\text{lowerbound}+1)*\text{Rnd}+\text{lowerbound})$.

upperbound 这里是随机数范围的上限，lowerbound 则是随机数范围的下限

VB6.0 基础教程 3.5.2 转换函数

转换函数用来完成这种转换工作，例如将十进制数转换成十六进制数，将字符转换成对应的 ASCII 码等。表 3.7 列出了常用的转换函数。

表 3.7 常用的转换函数

函数	说明	举例	值
Asc(s)	将字符转换成 ASCII 码	Asc("a")	97
Chr(n)	将 ASCII 码值转换成字符	Chr(97)	"a"
Hex(n)	将十进制数转换成十六进制	Hex(100)	64
Lcase(s)	将大写字母转换成小写字母	Lcase("HCQ")	"hcq"
Oct(n)	将十进制数转换成八进制	Oct(100)	144
Str(n)	将数值转换为字符串	Str(123.4)	"123.4"
Ucase(s)	将小写字母转换成大写字母	Ucase("hcq")	"HCQ"
Val(s)	将数字字符串转换为数值	Val("12 3.4abc56")	123.4

Lcase 函数仅将大写字母转换成小写字母，所有小写字母和非字母字符保持不变。Ucase 函数的情况与之类似。

例如：

LCase ("Hello World 1234") 返回"hello world 1234"

UCase ("Hello Wurlld 1234") 返回"HELLO WORLD 1234"

Val 函数在执行转换时，在它不能识别为数字的第一个字符上，停止读入字符串。那些被认为是数值的一部分的符号和字符，例如美元号 (\$) 与逗号 (,) ,都不能被识别。但是函数可以识别进位制符号 &O (八进制) 和 &H (十六进制) .空格、制表符和换行符都从参数中被去掉。

例如：

Val (" 1615 198th Street N.E" 返回值为 1615198

VB6.0 基础教程 3.5.3 字符串函数

字符串函数用来完成对字符串的操作与处理，如获得字符串的长度、除去字符串中的空格以及截取字符串等。表 3.8 中列出了 VB 中常用的字符串的数。

表 3.8 常用的字符串函数

函数	说明	举例	结果
Left(s, n)	返回字符串左边的 n 个字符	Left("ABCDEF", 4)	"ABCD"
Len(s)	返回字符串的长度	Len("ABCDEF")	6
Ltrim(s)	去掉字符串左边的空格	Ltrim(" ABC")	"ABC"
Mid(s, n1,n2)	返回字符串 s 中第 n1 位开始的 n2 个字符	Mid("ABCDEF", 2, 4)	"BCDE"
Right(s,n)	返回字符串右边的 n 个字符	Right("ABCDEF", 4)	"CDEF"
Rtrim(s)	去掉字符串右边的空格	Rtrim("ABC ")	"ABC"
Space(n)	产生 n 个空格的字符串	Space(3)	" "
String(n, s)	返回由 s 中首字符组成的包含 n 个字符的字符串	String(4, "ABCDEF")	"AAAA"
InStr(n1, s1, s2, n)	返回字符串 s2 在字符串 s1 中第一次出现的位置	InStr(4, "xxYxYx", "Y")	5
StrComp(s1, s2, n)	返回字符串 s1 与 s2 比较结果的值	StrComp("ABC", "abc")	-1

InStr (n1,s1,s2,n) 函数用来返回一个字符串在另一字符串中第一次出现的位置，如果没有则返回 0.该函数有 4 个参数，其中参数 s1 和 s2 是必选的，s1 是指接受搜索的字符串，s2 是指要搜索的字符串。参数 n1 和 n 是可选的，参数 n1 用于指定搜索的开始位置，如果省略，将从第一个字符的位置开始搜索。参数 n 用于决定是否区分大小写，如果 n=0 或省略，则区分大小写，如果 n=1,则不区分大小写：如果有 n 参数，则必须同时要有参数 n1。

例如：

S1="XXpXXpXXPXXP" 接受搜索的字符串

S2="P" 要搜索的字符串

MyPos=InStr (4,S1 S2,1) 从第四个字符开始不区分大小写比较，值为 6

MyPos=InStr (1,S1 S2,0) 从第一个字符开始区分大小写比较，值为 9

MyPos=InStr (S1 S2) 从第一个字符开始区分大小写比较，值为 9

MyPos=InStr (1,S1,"W") 没有找到"W",值为 0

StrComp (s1, s2,n) 函数用来返回两个字符串的比较结果，其中参数 n 的含义与 InStr 函数中参数 n 是相同的。如果 s1 小于 s2,则值为 -1;如果 s1 等于 s2,则值为 0;如果 s1 大于 s2,则值为 1.

例如：

S1="ABCD"

S2="abcd"

MyComp=StrComp (s1,s2,1) 返回 0

MyComp=StrComp (s1,s2,0) 返回-1

MyComp=StrComp (s2,s1) 返回 1

VB6.0 基础教程 3.5.4 日期函数

日期函数用于操作日期，与时间，例如获取当前的系统时间，求出某一天是星期几等。表 3.9 中列出了 VB 中常见的日期函数。

表 3.9 常见的日期函数

函数	说明	举例	结果
Time	返回当前的系统时间	Time	12:30:35
Timer	返回从午夜开始到现在经过的秒数	Timer	
Date	返回当前的系统日期	Date	00-10-21
Now	返回当前的系统日期与时间	Now	00-10-21 12:30:35
Day	返回日期代号 (1~31)	Day("1977.4.19")	19
Month	返回月份 (1~12)	Month("1977.4.19")	3
Year	返回年份	Year("1977.4.19")	1977
WeekDay	返回表示星期的代号, 星期日为 1, 星期 1 为 2.....	WeekDay("1977.4.19")	3

既可以使用 Time 和 Date 函数来获取当前的系统时间与日期, 也可以使用两者来设置系统的时间与日期。

实例 3.2 获取并设置系统的时间与日期。

编写一个小程序, 来获取当前的系统日期与时间, 并重新设置系统时间为 12 点整, 日期为 1997 年 7 月 1 日。

编写窗体的 Click 事件过程如下:

```
Private Sub Form_Click ()
Print "当前系统时间是: " & Time
Print "当前系统日期是: " & Date
Time=#10:00:0D PM#
Date=#7/1/1997#
Print "当前系统日期是: " & Date
End Sub
```

运行该程序, 在窗体上单击, 则显示出两组系统时间与日期。其中第一组是设置前的系统时间与日期, 第二组是设置后的系统时间与日期, 如图 3.10 所示。



图 3.10 实例 3.2 的运行结果

VB6.0 基础教程 3.6.1 顺序结构

我们知道，VB 采用的是事件驱动机制，即在运行时过程的执行顺序是不确定的，它的执行流程完全由事件的触发顺序来决定。但在一个过程的内部，仍然用到结构化程序的方法，使用流程控制语句来控制程序的执行流程。结构化程序设计有 3 种基本结构：顺序结构、选择结构与循环结构。如果没有流程控制语句，则各条语句将按照各自在程序中的出现位置，依次执行，即顺序结构。我们在前面编写的程序都是顺序结构。

顺序结构是按照程序或者程序段书写顺序执行的语句结构。如果 3.11 所示，先执行操作语句 A,再执行操作语句 B,两者是顺序执行的关系，用户不能期待先执行语句 B,然后才执行语句 A。

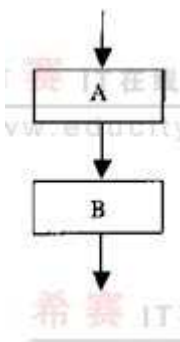


图 3.11 顺序结构

顺序结构是最基本的一种结构，它表明了事情发生的先后情况。在日常生活中有很多这样的例子。例如在淘米煮饭的时候，总是先淘

米，然后才煮饭，不可能是先煮饭后淘米。在编写应用程序的时候，也存在着明显的先后次序。

赋值语句是最常用也是最基本的语句。它的作用是将右边表达式的值赋给左边的变量。赋值语句的一般形式如下：

变量=表达式。

表达式的类型应与变量的类型一致，如同为数值型或同为字符串型。当同为数值型但精度不同时，会强制将表达式的值转换为变量的精度。

例如：

```
Dim i As Integer
```

```
Dim j As Integer
```

```
i=3.4
```

```
j=8.5
```

由于 i,j 都是整型，按照四舍五入的原则将赋给它的值转换为整型。因此，i 的实际值为 3,j 的实际值为 9.

赋值语句还用来在代码中设置属性的值。

例如：

```
Command1.Caption ="确定" 将按钮的标题设置为"确定".
```

```
Text1.Text="文本框" 在文本框中显示文本"文本框".
```

需要指出的是，赋值号。与关系运算符"等号"都是用"="表示，VB 会根据所处的位置自动判断"="是何种意义的符号。

例如：

`I=8=9.`

其中第一个"="是赋值号，第二个"="是关系运算符"等号".语句的含义是将关系运算表达式 `8=9` 赋给变量 `I`,因此，`I` 的值为 `0 (False)`。

VB6.0 基础教程 3.6.2 选择结构

选择结构是指根据所给的条件，选择执行的分支。它的特点是在若干个分支中必选且只选其一。VB 中提供了四种形式的条件语句，分别是 `If Then`、`If ThenElse`、`If Then ElseIf` 和 `Select Case`.在使用时，可以根据不同的条件。选择一种合适的条件语句。

1. `If...Then` 语句（单分支结构）

语句形式如下：

`if<表达式>Then.`

`<语句块>.`

`End if.`

其中<表达式>一般是关系表达式或逻辑表达式，也可以是算术表达式。<语句块>是指一条或多条要执行的语句。如果表达式的值不为零（`True`），即条件为真，则执行 `Then` 后面的语句块。如果表达式的值为零（`False`）即条件为假，则不执行 `Then` 后面的语句块，而直接开始执行 `End If` 后的其他语句。该条件语句只有一个分支，因此称为单分支结构。其流程如图 3.12 所示。

例如：如果甲的年龄（`Age1`）与乙的年龄（`Age2`）相同，则在窗体上显示出他们的年龄，并且显示一行文本"甲与乙同岁".语句如下：

```
If Age1=Age2 Then
```

```
Print Age1
```

```
Print"甲与乙同岁"
```

```
End If
```

如果语句块中只有一条语句，也可以写成一种较简单的形式：

```
If<表达式>Then<语句块>
```

如果语句块中有多条语句，要写成上述简羊形式，则各条语句之间必须以冒号分隔。例如：

```
If Age1=Age2 Then Print Age1:Prin"甲与乙同岁"
```

2.If... Then...Else 语句（双分支结构）

语句形式如下

```
If<表达式>Then
```

```
<语句块 1>
```

```
Else
```

```
<语句块 2>
```

```
End If
```

如果<表达式>的值不为零（True）,即条件为真，则执行 Then 后面的语句块。否则，执行 Else 后面的语句块。该条件语句有两个分支，因此称为双分支结构。其流程如图 3.23 所示。

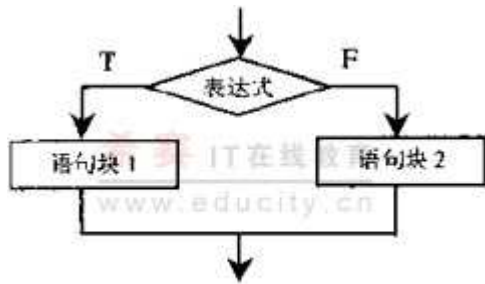


图 3.13 双分支结构

例如：对上例进行扩充，如果甲与乙的年龄不相同，则在窗体上分别显示出它们的年龄，并且显示一行文本"甲与乙不同岁"。语句如下：

```

If Age=Age2 Then
Print Age1
Print"甲与乙同岁"
Else
Print Age1
Print Age2
Print"甲与乙不同岁"
End If
  
```

同样，如果语句块只有一条语句，也可以写成一种较简单的形式：

```

If<表达式>Then<语句块 1> Else<语句块>
  
```

3.If...Then...ElseIf 语句（多分支结构）

语句形式如下：

```

If<表达式 1>Then
<语句块 1>
ElseIf<表达式 2> Then
  
```

<语句块 2>

Else

<语句块 n>

End If

该语句可以有多个分支，称为多分支结构。多分支结构是这样执行的：先测试<表达式 1>，如果值为 0（True），即条件为真，则执行 Then 后面的<语句块 1>；如果<表达式 1>的值不为 0（False），继续测试<表达式 2>的值，如果值为 0（True）执行 Then 后面的<语句块 2>……就这样依次测试下去。只要遇到一个表达式的值为 0，就执行它对应的语句块，然后执行 End If 后面的语句，而其他语句块都不执行。如果所有表达式的值均不为 0，即条件都不成立，则执行 Else 后面的<语句块 n>。其流程如图 3.14 所示。

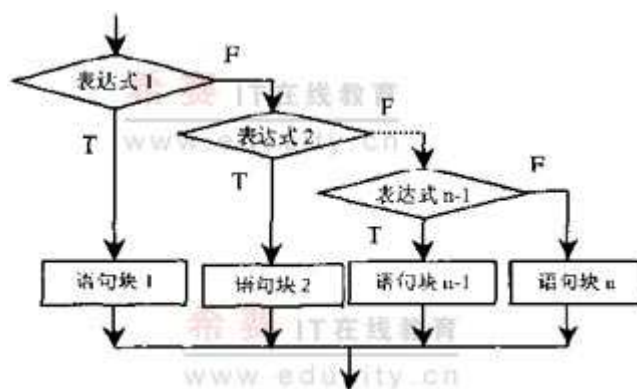


图 3.14 多分支结构

例如，下面代码可用于评定学生的成绩；

```
Dim S As Single
```

```
s=89
```

```
If s >= 0 And S < 60 Then
```

```
Print "差"
```

```
Else If S>=60 And S<75 Then  
Print"中"  
  
Else If S>=75 And S<86 Then  
Print"良"  
  
Else If S>=85 And S<100 Then  
Print"优"  
  
Else  
Print "成绩错误"  
  
End If
```

4. Select Case 语句（情况语句）

Select Case 语句的一般形式如下：

```
Select Case<变量>  
Case<值列表 1>  
    <语句块 1>  
Case<值列表 2>  
    <语句块 2>  
.....  
Case<值列表 n-1>  
[Case Else  
    <语句块 n>]  
End Select
```

Select Case 语句也是用来实现多分支选择的。其中的<变量>可以是数值型或字符串型。每个 Case 子句指定的值的类型必须。与<变量>的类型相同。Case 子句中指定的值可以是下面 4 种形式之一：

(1) 一个具体的值或表达式，例如：

Case 2 变量的值是 2

(2) 一组值，用逗号隔开，例如：

Case 1, 3, 5 变量的值是 1,3 或 5

(3) 值 1 To 值 2,例如：

Case 1 To 10 变量的值在 1 到 10 之间

(4) Is 关系运算符表达式，例如：

Case Is<10 变量的值小于 10

当变量的值与某个 Case 子句指定的值匹配时，就执行该 Case 子句中的语句块，然后执行 End Select 后面的语句。因此。即使变量同时与多个 Case 子句指定的值相匹配，也只是执行第一个与变量匹配的 Case.子句中的语句块。这一点与 If...Then...Elseif 语句相同。Case Else 子句是可选的，如果变量的值与任何一个 Case 子句提供的值都不匹配，则执行 Case Else 子句后面的<语句块 n>.其流程如图 3.15 所示。

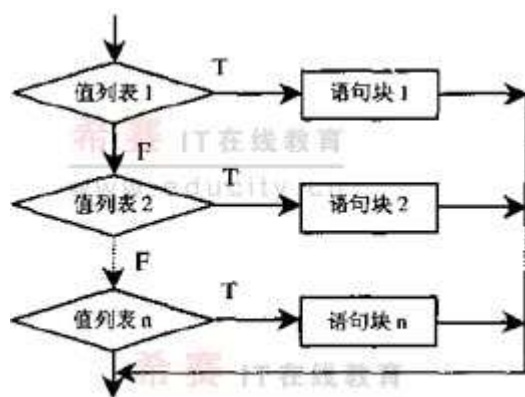


图 3.15 情况语句

VB6.0 基础教程 3.6.3 循环结构

循环结构是指在一定条件下多次重复执行一组语句。VB 中提供了两种循环语句，它们是 For 语句和 Do 语句。

1. For 循环语句

如果已知某一段代码需要重复执行的次数，可以使用 For 循环语句。该语句的一般形式如下：

```

For<循环变量>=<初值>To<
终值>[Step<步长>]
    <语句块>
[Exit For]
Next<循环变量>

例如:

For i=1 To 8 Step1
    Print “这是文本”&i
Next i
  
```

该段代码的功能是在窗体上显示 8 行文本，如图 3.16 所示。如果不采用循环结构，则需要使用 8 条 Print 语句来实现相同的功能。



图 3.16 使用 For 语句 (Step=1)

For 语句中的循环变量必须是数值型，初值、终值以及步长则是具体的数值。

步长用来指定循环变量每次的增量，当所有循环体中的语句都执行后，循环变量就会自动增加一个步长。默认的步长为 1。For 语句的执行流程如图 3.17 所示。

具体执行流程为：

- (1) 将初值赋给循环变量。
- (2) 判断循环变量的取值是否在终值范围内否则结束循环，执行 Next 的下一条语句。
- (3) 将循环变量的取值自动增加一个步长，然后回转到步骤 (2) 继续执行。

若是则执行循环体内的语句，循环中可以在任何位置放置任意个 Exit For 语句，该语句的作用是退出循环，转到 Next 语句的下一条语句。Exit For 语句经常在条件判断之后使用，例如在 If...Then 语句之后。

例如，修改上例如下：

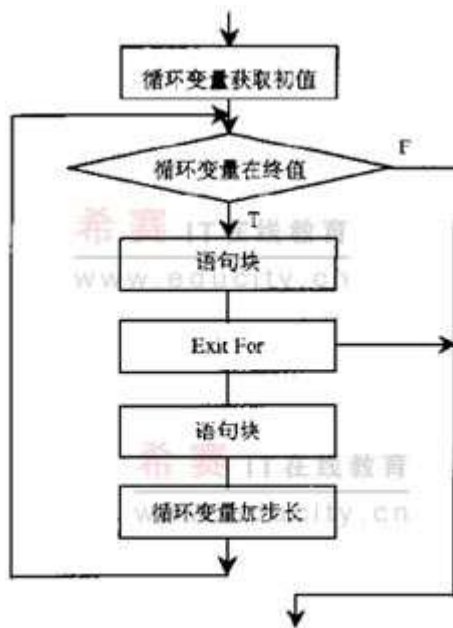


图 3.17 For 循环语句

```

For I=1 To 10

For J=1 To 10

For K= 1 To 10

....

Next K

Next J

Next I
  
```

实例 3.3 打印九九乘法表

本例讲解怎样打印九九乘法表，如图 3.18 所示。九九乘法表的打印看起来很烦琐，但是如果使用嵌套循环，则问题变得非常简单。

1x1=1								
2x2=4	1x2=2							
3x3=9	2x3=6	1x3=3						
4x4=16	3x4=12	2x4=8	1x4=4					
5x5=25	4x5=20	3x5=15	2x5=10	1x5=5				
6x6=36	5x6=30	4x6=24	3x6=18	2x6=12	1x6=6			
7x7=49	6x7=42	5x7=35	4x7=28	3x7=21	2x7=14	1x7=7		
8x8=64	7x8=56	6x8=48	5x8=40	4x8=32	3x8=24	2x8=16	1x8=8	
9x9=81	8x9=72	7x9=63	6x9=54	5x9=45	4x9=36	3x9=27	2x9=18	1x9=9

图 3.18 九九乘法表

打开【代码】窗口，编写窗体的 Click 事件过程如下：

```
Private Sub form_Click()

    Print

    “-----”

九九乘法表

-----”

    Print

    For i=1 To 9

    For j=1 To i

        s=i*j

        Print j & “X” & i & “=” &

s

    Next j

    Print

    Next i

End Sub
```

在该段代码中，首先使用 **Print** 方法打印出标题和一个空行。然后使用了一个两重的嵌套循环结构。关于 **Print** 方法的使用细节请参见下一章。

2.Do 循环语句

如果不知道某一段代码需要重复执行的次数，可以使用 **Do** 循环语句。该语句有两种基本形式：

(1) Do While<表达式>

循环体

[Exit Do]

Loop

这种格式的 **Do** 循环先判断条件，后执行循环体。与 **If...Then** 语句类似，**While** 子句的<表达式>一般是关系表达式或逻辑表达式，也可以是算术表达式。如果表达式的值不为零 (**True**)，即条件为真，则执行循环体。如果表达式的值为零 (**False**)，即条件为假，则终止循环。其流程如图 3.19 所示。



图 3.19 Do While...Loop

例如，通过下列代码，可以求出 2460 和 345 的最大公约数。

```
Private Sub Form_Click()
```

```
m=2460:n=345

Dim s As Single

Do while(n<>0)

s=m Mod n

m=n

n=s

Loop

Print “最大公约数=” & m

End Sub
```

在 Do...Loop 中可以在任何位置放置任意个数的 Exit Do 语句，随时跳出 Do-Loop 循环。Exit Do 通常用于条件判断之后，例如 If...then,在这种情况下，Exit Do 语句将控制权转移到紧接在 Loop 命令之后的语句。

如果 Exit Do 使用在嵌套的 Do...Loop 语句中，则 Exit Do 会将控制权转移到 Exit Do 所在位置的外层循环。

While 子句也可以出现在 Loop 语句后，形式如下：

Do.

循环体。

[Exit Do].

Loop While<表达式>.

这种格式的特点是先执行循环体，后判断条件。也就是说第一次进入循环是无条件的。循环体至少会被执行一次，其流程如图 3.20 所示。而如果 While 子句出现在 Do 后，则可能一次也不执行。

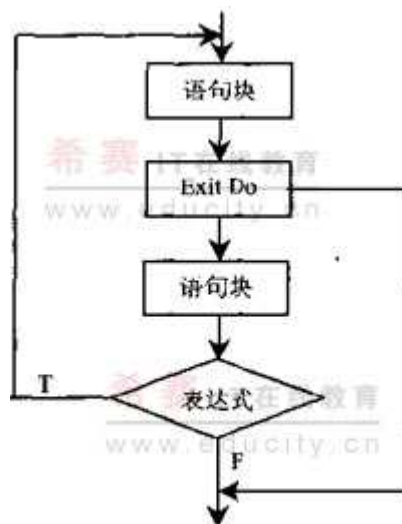


图 3.20 Do ...While Loop

(2) Do Until<表达式>.

循环体。

[Exit Do].

Loop.

将 While 子句换成 Until 子句后，情况正好相反。只有当表达式的值不为 0 (ture),即条件为真时才终止循环，否则继续循环。

Until 子句也可以出现在助 Loop 的后面，也表示先执行循环体，后判断条件。

VB6.0 基础教程 3.7 数组

数组是指具有相同名称和类型的一组变量，数组中的每个变量称为数组元素。由于有了数组，可以用相同名字引用一系列变量，并用索引号（下标）来识别它们。在许多场合，使用数组可以缩短和简化

程序。例如，要编写一个程序，来计算 100 个学生某门功课的总成绩与平均成绩。如果定义 100 个不同的变量来分别存储每个学生的成绩，则在程序编写时的烦琐可想而知。如果在程序中使用数组，则类似这样的问题都可以迎刃而解。

1.一维数组

在使用数组前必须先声明它，声明数组的一般形式如下：

Dim<数组名>（下标）As<数据类型>.

其中"下标"的一般形式为：**[下界] To 上界**，用于确定数组中元素的个数。

省略下界时，默认值为 0.数组中元素的个数称为数组的大小，因为数组的元素在上下界内是连续的，因此，一维数组的大小为：（上界一下界）+1.

例如：

Dim Sc（3 To 6） As Integer.

声明了一个名称为 Sc、大小为 4 的一维数组，该数组包含 4 个元素，它们分别是 Sc（3）, Sc（4）. Sc（5）和 Sc（6）,并且每个元素都是整型的。

再如：

Dim Sn（5） As String.

声明了一个名称为 Sn 的一维数组，该数组包括 6 个元素，它们分别是 Sn（0）,Sn（1）,Sn（2）、Sn（3）、Sn（4）和 Sn（5）.

一个数组中的所有元素具有相同的数据类型。与声明变量一样，如果在声明数组时忽略 **As** 子句，则数组为变体型。变体型数组的各个元素能够包含不同类型的数据，如字符串、数值等等。

在声明了数组后，**VB** 会自动为其中的每个元素赋初值。如果是数值型数组，则每个元素的初值都为 0；如果是字符串型数组，则每个元素都将是一个空字符串。

数组是程序设计中经常用到的结构类型，将数组元素的索引号和循环语句结合起来使用，能解决大量的实际问题。如求一组数据的总和、平均值以及最大值与最小值等。对数组的操作，是针对某个具体的元素进行的，一个元素可以看成是一个独立的变量。例如：

```
Sn (2) = "ABC".
```

该语句为数组 **Sn** 中的第三个元素赋值"ABC",而语句 **Sn="ABC"** 则是错误。

下面通过两个实例，进一步理解数组的概念，学会在程序中恰当地使用数组。

实例 3.4 为数组赋值。

在该实例中，声明了一个包含 6 个元素的整型数组，分别为它们赋值 1、2、.....6.然后在窗体上打印出各元素的值、它们的总和以及平均值。

在【代码】窗口中编写窗体的 Click 事件过程如下：

```
Private Sub Form_Click()  
    Dim Num(5) As Integer
```

```

Dim S AS integer

Dim A AS Singel

S = 0

For i=0 To 5

    Num(i)=i+1

    s=s+Num(i)

    Print “第” & i+1 & “个元素的
值为： ” & Num (i)

Next i

A = S /6

Print 空行

Print “总和为： ” & S

Print “平均值为： ” & A

```

运行该程序，单击窗体，则结果如图 3.21 所示。

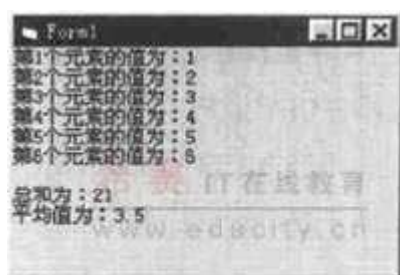


图 3.21 实例 3.3 的运行结果

实例 3.5 求最大值与最小值。

假设已使用 `Dim Num (99) As Integer` 语句定义了一个数组，并且在该数组中存储了 100 个学生的成绩，请编写代码在窗体上打印出最高分与最低分。

代码如下：

```
Dim Max As Integer  
Dim Min As Integer  
Max=Num(0)  
Min=Num(0)  
For i= 1 To 99  
    If Num(i)>Max Then Max =  
Num(i)  
    If Num(i)<Min Then Min =  
Num(i)  
Next i  
Print “最高分是： ” & Max  
Print “最低分是： ” & Min
```

代码中定义了变量 **Max** 来保存最高分，并且将第一个学生的成绩（保存在元素 **Num (0)** 中）赋给它。然后使用 **For** 循环语句让变量 **Max** 与数组的其他元素依次比较。在循环语句中又使用了 **If...Then** 语句来判断其他元素与 **Max** 的大小，如果某元素比 **Max** 大，则将该元素的值赋给 **Max**,**Max** 再以这个值同其他元素进行比较。因此，在比较结束后，**Max** 中保留的是所有元素中最大的值。

变量 **Min** 用来保存最低分，算法与求最高分的算法类似。

2. 多维数组

声明多维数组的形式如下

Dim<数组名> (下标 1, 下标 2, ...) **As**<数据类型>.

其中下标的形式与一维数组中的下标相同，下标的个数决定了数组的维数。

多维数组的大小为每一维的大小乘积，每一维大小为（上界一下界）+1.

例如：

Dim A (3, 3 To 5) As Integer.

声明了一个整型二维数组，数组的大小为 (3-0+1) X (5-3+1) = 12. 其中各元素如表 3.10 所示。

表 3.10 二维数组 A 中的元素

A(0, 3)	A(0, 4)	A(0, 5)
A(1, 3)	A(1, 4)	A(1, 5)
A(2, 3)	A(2, 4)	A(2, 5)
A(3, 3)	A(3, 4)	A(3, 5)

又如：

Dim B (2, 3, 4) As String

声明了一个字符串型三维数组，数组的大小为 3*4*5=60.

与变量相同，数组的作用范围也是由其声明方式与声明位置决定的，即：

建立全局数组，在模块的声明段用 **Public** 语句声明数组。

建立模块级数组，在模块的声明段用 **Private** 语句声明数组。

建立局部数组，在过程中用 **Dim** 语句或 **Static** 语句声明数组。

VB6.0 基础教程 3.8.1 子过程

VB 程序是由多个过程构成的，这些过程可分为两大类；其中一类是系统提供的事件过程，例如窗体或按钮的 Click 事件过程等。事件过程是构成 VB 应用程序的主体，由事件触发执行，例如单击按钮，则按钮的 Click 事件过程就会执行。在前面的一些实例中用到的都是事件过程。

另一类过程是通用过程，由用户根据需要自行定义，以供事件过程调用。在程序中，有些处理需要经常重复进行，这些处理的代码是相同的，只不过每次都以不同的参数调用。例如，要计算整数 1~n 的累加结果，n 的大小可以由用户决定，因此它是不确定的。这样就可以定义一个以 n 为参数的过程，为了得到不同 n 时的累加结果，以不同的参数 n 调用该过程就可以了。使用过程的好处就在于使得程序简练，同时也便于程序的设计与维护。

通用过程又可以分为 Sub 子过程（简称子过程）和 Function 函数过程（简称函数过程）。

子过程用来完成特定的任务，其定义有两种方法。

1. 直接在【代码】窗口中输入。

打开窗体或标准模块的【代码】窗口，将插入点定位在所有现有过程的外面，然后输入子过程即可。

子过程的形式如下：

[Private][Public][Static]Sub<过程名>[(参数表)]

<语句>

[Exit Sub]

<语句>

End Sub

具体说明如下：

Sub 是子过程的开始标记，End Sub 是子过程的结束标记，<语句>是具有特定功能的程序段，Exit Sub 语句表示退出子过程。

如果在子过程的前面加上 Private,则表示它是私有过程，其作用范围局限于本模块。如果在子过程的前面加上 Public,则表示它是公用过程，可在整个应用程序范围内调用。可见，子过程的作用域与变量的作用域类似。

表 3.11 中详细列出了子过程的作用范围及调用规则。

表 3.11 子过程的作用域

过程作用域	模块级（私有）		全局级（公用）	
	窗体模块	标准模块	窗体模块	标准模块
定义方式	子过程名前加 Private		子过程名前加 Public	
能否被本模块的其他过程调用	能	能	能	能
能否被本应用程序的其他模块调用	不能	不能	能，但必须在过程名前加窗体名	能，但过程名必须唯一。否则要加标准模块名

如果在子过程名的前面加上 Static,则表示该过程中的所有局部变量都是静态变量。

参数是调用子过程时给它传送的信息子。过程可以有参数，也可以不带参数，没有参数的过程称为无参过程。如果带有多个参数，则各参数之间使用逗号隔开。子过程中的参数表形式为：

[ByVal]<参数 1>[As<类型>][.[ByVal]<参数 1>[As<类型>1,...].

参数可以是变量，也可以是数组。当参数是数组时，参数声明时应省略其维数，但括一号不能省。这里的参数称为形参，在定义时它没有位。在参数名前有 **Byval** 关键字表示当该过程被调用时，参数的传递方式是传值，否则是传址。关于参数以及参数的传递在后面还会详细介绍。

2.使用【添加过程】对话框

为了省去输入子过程框架的麻烦，也可以通过【添加过程】对话框在【代码】窗口中自动添加，步骤如下：

- (1) 打开想要添加子过程的【代码】窗口。
- (2) 执行【工具】菜单中的[添加过程]命令，打开【添加过程】对话框，如图 3.22 所示。
- (3) 在【名称】文本框中输入子过程名，在【类型】组中选中【子程序】单选按钮，在【范围】组中选择子过程的作用范围。如果选中【所有本地变量为静态变量】复选框，在子过程名前将加上 **Static** 关键字。
- (4) 设置完毕后，单击【确定】按钮，则代码窗口中就出现了相应的子过程的框架，如图 3.23 所示。其中在【添加过程】对话框中输入的名称是 **Abc** 选择的范围是"公有的"。



图 3.22 【添加过程】对话框



图 3.23 在代码窗口中添加了子过程的框架

提示：使用【添加过程】对话框添加的子过程总是无参过程，用户可自行对子过程添加参数和编写过程体。

在定义了子过程后，就可以在事件过程或其他过程中调用了。调用子过程有两种方法：

使用 Call 语句：Call<过程名>（参数表）

直接使用过程名：<过程名>[<参数表>]

注意：当使用 Call 关键字时，参数表必须放在括号内，若省略 Call 关键字，也必须同时省略括号。

VB6.0 基础教程 3.8.2 函数过程

函数过程与子过程一样，也是用来完成特定功能的独立程序代码。与子过程不同是，函数过程可以返回一个值给调用程序。

函数过程的形式如下：

```
[Private][Public][Static]Function<函数名>[(参数表)][As 类型]
    <语句>
    [Exit Function]
    <语句>
End Function
```

可见，函数过程的形式与子过程的形式类似。Function 是函数过程的开始标记，End Function 是函数过程的结束标记，<语句>是具有特定功能的程序段，ExitFunction 语句表示退出函数过程。As 子句决

定函数过程返回值的类型，如果忽略 **As** 子句，则函数过程的类型为变体型。其他各部分的功能同子过程中相应部分的含义完全相同。

由于函数过程要返回一个值，因此，在过程内部应该至少有一条为<函数名>（函数名就像一个变量）赋值的语句。

同样，也可以使用【添加过程】对话框在【代码】窗口中添加函数过程的框架，只要在【类型】组中选择【函数】单选按钮即可。

函数过程是用户自定义的函数，它的调用与使用 **VB** 的内部函数没有区别。最简单的情况是将函数的返回值赋给一个变量，其形式为：

变量名=函数名（参数表）

实例 3.6 使用函数过程

编写一个函数，用来计算整数 1~n 的累加结果，n 的大小在调用函数时指定。为此，编写一个以 n 为参数的函数过程，代码如下：

```
Private Function Sum (n As Integer) As  
Integer  
    Dim s As Integer  
    S=0  
    For j=1 To n  
        s=s+j  
    Next  
    Sum=s  
End Function
```

在窗体的 Click 事件过程中调用自定义的函数，代码如下：

```
Private Sub Prom_Click()  
  
Dim r As Integer  
  
r=Sum(100)  
  
Print F  
  
Ebs Sub
```

在调用自定义过程时，调用者是通过参数向过程传递信息的。通常，把自定义过程中的变量称为形参；把调用这个过程时使用的参数称为实参。在 VB 中，参数的传递方式有两种：传址和传值，其中传址也被称为引用，是 VS 默认的参数传递方式。如果在定义过程时，在形参前加上关键字 **ByVal** 则参数传递方式变为传值。

传址是指在调用过程时，将实参的地址传递给形参。因此，在被调用的过程体中对形参的任何操作都变成了对实参的操作。例如，当形参的值变为原来的 2 倍时，则实参的值也变为原来的两倍。

传值是指在调用过程时，将实参的值赋给形参，而实参本身与形参没有联系。因此，在被调用过程体中对形参的任何操作都不会影响到实参。

实例 3.7 传址和传值

编写一个用于交换两个数的子过程 Swl,该过程采用传址方式传递参数，过程代码如下：

```
private Sub Swl (a As Single,b As Single)  
  
Dim s As Single
```

```
s=b  
  
b=a  
  
a=s  
  
End Sub
```

再编写一个过程体与 Sw1 完全相同的子过程 Sw2,不同的是 Sw2 采用的是传值方式传递参数，过程代码如下：

```
Private Sub Sw2 (Byval a As Single,Byval b  
As single)  
  
Dim s As Single  
  
s=b  
  
b=a  
  
a=s  
  
End Sub
```

编写窗体的 Click 事件过程，在该过程中使用相同的参数分别调用子过程 Sw1 和 Sw2,代码如下：

```
Private Subb Form_click()  
  
Dim x As single  
  
Dim y AS Single  
  
x=10  
  
y=20  
  
Print “x= “& x  
  
Print “y= “& y
```

```

swl x, y 调用子过程 swl

Print "使用子句 swl 交换两个数后:

Print "x= "& x

Print "y= "& y

x=10 重新给 x, y 赋值

y=20

sw2x, y 调用子过程 sw2

Print "使用子过程 sw2 交换两个数后: "

Print "x=" & x

Print "y=" & y

End Sub

```

运行该程序，单击窗体，结果如图 3.24 所示。从打印出的结果可以看出，调用子程序 Swl 达到了交换两个数的目的，而调用子程序 Sw2 后，两个数却没有发生任何变化。这就是因为子程序 Sw2 采用的是传值的方式，实参 x 和 Y 仅仅是将它们的值分别赋给了形参 a 和 b,而 x/y 本身与 a/b 没有任何联系，因此，在过程体中对形参 a 和 b 的操作不会影响到 x 和 y。



图 3.24 实例 3.7 的运行结果

VB6.0 基础教程 3.8.3 可选参数

在前面的实例中，我们可以看到自定义过程的形参个数总是固定的，并且在调用它时，必须提供与形参相对应的实参，即实参的个数与类型要与形参的完全相同。

在 VB 中，也可以定义具有可选参数的过程，所谓"可选参数"是指过程中的一类特殊形参，在调用过程时，可以为它们提供实参也可以不提供实参。如果不提供实参，则可选参数使用它的默认值。

将某个形参定义为可选参数的方法是在它的名称前加上关键字 **Optional**，在定义的同时也可指定它的默认值。需要注意的是，可选参数一定要位于子过程或函数过程的末尾。

实例 3.8 使用可选参数

编写一个用于计算两个实数的和以及平均值的子过程，该子过程具有一个可选参数，在调用子过程时，如果不提供与可选参数对应的实参，或实参的值为 0，则在窗体上只打印两个数的和；如果提供了一个不为 0 实参，则在窗体上还将打印出两个数的平均值。

具有可选参数的子过程的代码如下：

```
Private Sub(x As Single,y As Single,Optional  
n As Boolean=0)  
  
Dim s As Single,A As Single  
  
s=x+y  
  
A=S/2  
  
If n=False Then  
  
Print “总和为： ” & S
```

```

Else

Print “总和为：” & S

Print “平均值为：” & A

End If

End Sub

```

在窗体的 Click 事件过程中调用子过程 Sum.代码如下：

```

Private Sub Form_Click()

Print “不提供与可选参数对应的实参”

Sun 6, 9

Print “提供与可选参数对应的实参，并且
实参为 0：”

Sun 6, 9,0

Print 空行

Print “提供与可选参数对应的实参，并且
实参不为 0：”

Sun 6, 9,1

End Sub

```

运行程序，单击窗体，结果如图 3.25 所示。



图 3.25 实例 3.8 的运行结果

VB6.0 基础教程 3.8.4 递归

递归是推理和问题求解的一种强有力方法，原因在于许多对象，特别是数学研究对象具有递归的结构。简单地说，如果通过一个对象自身的结构来描述或部分描述该对象就称为递归。最简单而易于理解的一个例子是阶乘的递归定义。如果以函数 $f(n)$ 表示自然数 n 的阶乘的值，则有定义：

递归定义使我们能够用有限的语句描述一个无穷的集合。本例描述一个无穷的集合只用了两个语句。

VB 程序设计语言允许一个过程体有调用自身的语句，称为递归调用。也允许调用另一过程，而该过程又反过来调用本过程，称为间接递归调用。这种功能为求解具有递归结构的问题提供了强有力手段，使程序语言的描述与问题的自然描述完全一致，因而使程序易于理解，易于保证和维护。例如，对于上面的 n 阶乘的递归定义，可以写出相应的 VB 函数过程，如下面的实例 3.9.这个过程的推理（计算）路线与原来函数的（递归）数学定义完全一致。

实例 3.9 求 n 阶乘的 VB 函数过程

```
Private Function F(byval n As Integer) As Integer
    If n=1 Then
        F=1
    Else
        F=n*F(n-1)
    End If
```

End Function

让我们来跟踪这个程序的计算过程，令 $n=4$ 调用这个函数，用下面的形式来表示递归求解的过程：

- (1) $F(4)=4*F(3)$ $n=4$ 调用函数过程 $F(3)$
- (2) $F(3)=3*F(2)$ $n=3$ 调用函数过程 $F(2)$
- (3) $F(2)=2*F(1)$ $n=2$ 调用函数过程 $F(1)$
- (4) $F(1)=1$ $n=1$ 求的 $F(1)$ 的值
- (5) $F(2)=2*1=2$ 回归， $n=2$ ，求得 $F(2)$ 的值
- (6) $F(3)=3*2=6$ 回归， $n=3$ ，求的 $F(3)$ 的值
- (7) $F(4)=4*6=24$ 回归， $n=3$ ，求得 $F(4)$ 的值

上面第 1 步到第 4 步求出 $F(1)=1$ 的步骤称为递推，从第 4 步到第 7 步求出 $F(4)=4*6$ 的步骤称为回归。

从这个例子可以看出，递归求解有两个条件：

给出递归终止的条件和相应的状态。

在本例中递归终止的条件是 $n=1$ ，状态是 $F(1)=1$ 。

2. 给出递归的表述形式，并且这种表述要向着终止条件变化，在有限步内达到终止条件。

在本例中，当 $n>1$ 时，给出递归的表述形式为 $F(n)=n*F(n-1)$ 。函数值 $F(n)$ 用函数值 $F(n-1)$ 来表示。参数的值向减少的方向变化，在第 n 步出现终止条件 $n=1$ 。

第四章 窗体的设计

VB6.0 基础教程 4.1 窗体的属性

常见的窗体的外观如图 4.1 所示，它有一个标题栏，标题栏的左端显示控制图标和窗体的标题。单击控制图标可出现一个下拉菜单，其中提供了对窗体的各种控制命令。标题栏的右端是 3 个窗体状态控制按钮。单击它们分别可使窗体最小化、最大化（或还原）和关闭。窗体的背景色一般为灰色。从【窗体设计】窗口中可以看出，VB 默认的窗体外观也正是这样的。



图 4.1 常见的窗体外观

在使用 VB 创建应用程序时，用户可根据需要改变窗体的外观。如更改控制图标和窗体的标题，确定显示哪些状态控制按钮，甚至可以决定是否显示窗体的标题栏。

设置窗体的外观属性很简单，不需要编写任何程序代码，只需在窗体的【属性】窗口为各属性设置相应的值即可。在表 4.1 中列出了窗体的几个主要的外观属性。

表 4.1 窗体的主要外观属性

属性	说明
Caption	决定窗体标题栏中显示的文本
MaxButton 和 MinButton 属性	决定窗体的最大化或最小化按钮是否有效
ControlBox	决定是否显示窗体的控制菜单图标与状态控制按钮
BorderStyle	设置窗体的边框样式、是否显示标题栏、是否可以调整大小等
BackColor	设置窗体的背景颜色
Picture	设置窗体的背景图片
Icon	设置控制菜单的图标
Height 和 Width	设置窗体的大小

其中 BorderStyle 属性有 5 个可取的值，每个值对应一种窗体的外观，表 4.2.中列出了 BorderStyle 属性的取值及其对应的窗体外观。

表 4.2 BorderStyle 属性取值

取值	说明
0-None	窗体无边框与标题条，且窗体无法移动与调整大小。建议尽量不要使用这种窗体
1-Fixed Single	窗体有可见边框及标题栏，无最小化与最大化按钮，不能调整大小
2-Sizable	这是默认值，也是最常见的窗体形式，窗体的大小与位置均可调整
3-Fixed Dialog	窗体可移动，但不能调整大小
4-Fixed ToolWindows	有标题栏，但无控制菜单图标，且标题栏较窄；无最小化与最大化按钮，不能调整大小
5-Sizable ToolWindows	与 4 相同，但可以调整大小

除了可以设置窗体的各种外观属性外，还可以设置窗体的其他一些属性。如窗体的位置以及窗体启动时的状态等。如表 4.3 所示的是窗体的其他一些重要的属性。

表 4.3 窗体的其他一些重要的属性

属性	说明
Left 和 Top	根据屏幕的左上角确定窗体的位置
ShowInTaskbar	决定窗体是否在任务栏中显示
Moveable	决定窗体是否可以移动
Visible	决定窗体在程序运行时是否可见
WindowState	设置窗体在启动时的状态。如最大化、最小化或正常大小

窗体和各种控件的设置窗体或控件的名称，【属性】窗口中，第一个属性都是"名称",这个属性用来在程序代码中就是使用这个名称来引用窗体或控件的。

提示：首次在工程中添加窗体时，该窗体的名称缺省为 Form1; 添加第二个窗体，其名称缺省为 Form2,以此类推。在代码中就是用这个名称来引用该窗体的。因此，最好给 Name 属性设置一个有实际意义的名称，如给一个关于窗体命名为 About.

要熟悉这些窗体属性，最好的办法是实践。在【属性】窗口中更改窗体的一些属性，然后运行该应用程序并观察修改后的效果。如果

想得到某个属性的详细信息，可以选择该属性并按 F1 键查看联机帮助。

VB6.0 基础教程 4.2.1 鼠标事件

在第 2 章已经介绍过，鼠标事件有共有 5 种，它们分别是 **MouseDown**（按下鼠标键）、**MouseUp**（释放鼠标键）、**MouseMove**（移动鼠标）、**Click**（单击）与 **DblClick**（双击）。

1. Click 事件与 DblClick 事件

窗体的 Click 事件过程的形式如下：

```
Private Sub Form_Click ()  
  
End Sub
```

在该事件过程中添加一段代码，运行程序时，当使用鼠标单击窗体时，则该段代码就会被执行。

提示：使用双击窗体的方法打开【代码】窗口。出现在代码窗口中的事件过程不是 Click 事件过程，用户可以在【代码】窗口的事件框中选择 Click 事件，则 Click 事件过程的框架就会出现在代码编辑区中。用户也可以自行输入事件过程的框架。对于其他一些事件，如 DblClick 事件，待况与此类似。

窗体的 DblClick 事件过程的形式与 Click 事件过程的形式类似，如下所示：

```
Private Sub Formes DblClick ()  
  
End Sub
```

在该事件过程中添加一段代码，运行程序时，当使用鼠标双击窗体时，则该段代码就会被执行。

注意：双击鼠标会同时触发 Click 事件与 Db1Click 事件，即在程序运行时，当用户双击窗体时，则 Click 事件过程与 Db1Click 事件过程都将被执行。

2. MouseDown 事件与 MouseUp 事件

窗体的 MouseDown 事件过程与 MouseUp 事件过程类似，形式如下：

```
Private Sub Form_McuseDown(Button As Integer,shift As Integer,X As Single,Y As Single)

End Sub

Private Sub Form_MouseUp(Button As Integer,shift As Integer,X As Single,Y As Single)

End Sub
```

与 Clicik 与 Db1Click 事件过程不同，在这两个事件过程中，含有 Button、Shift,X 和 Y 四个参数，其中参数 Button 用来判断用户按下的是鼠标的哪一个键。参数 Shift 用来判断是否按下 Shift, Ctrl 或 Alt 键构成组合状态，参数 X 和 Y 用来返回指针所在的位置。

表 4.4 中列出了 MouseDown 事件过程中参数 Button 的返回值与对应的操作。它同样也适用于 MouseUp 事件过程。

表 4.4 参数 Button 的返回值与对应的操作

返回值	操作
0	未按任何键
1	按下左键
2	按下右键
3	同时按下左键和右键
4	按下中键
5	按下中键和左键
6	按下中键和右键
7	同时按下左、中、右键

实例 4.1 识别用户所按的键

在该程序中,当用户将鼠标移动到窗体上时,如果按下左键,则窗体上显示"您按下的是左键",如图 4.2 所示,如果按下右键,则窗体上显示"您按下的是右键",如图 4.3 所示。



图 4.2 按下左键的效果



图 4.3 按下右键的效果

打开窗体的【代码】窗口,将下列代码添加到 Form_MouseDown 事件过程中:

```
Private Sub Form_MouseDown(Button As Integer, shift
As Integer,X As Single,Y As Single)

    Select Case Button

    Case 1

        Form1.Print "您按下的是左键"

    Case 2

        Form1.Print "您按下的是右键"

    End Select
```

End sub

在该段代码中，使用了 **Select Case** 语句来判断参数 **Button** 的值，使用窗体的 **Print** 方法来在窗体上显示文本。**Print** 方法是窗体的一个很重要的方法，在很多实例中都使用到了该方法。

运行该程序，当在窗体中按下鼠标的键时，就会触发 **Form_MouseDown** 事件过程，并将所按键代表的数值赋给参数 **Button**。因此，**Select Case** 语句就可以通过参数 **Button** 的值来判断用户所按的键。

表 4.5 中列出了 **MouseDown** 事件过程中参数 **Shift** 的返回值与对应的操作。它同样也适用于 **MouseUp** 事件过程。

表 4.5 参数 Shift 的返回值与对应的操作

返回值	操作
0	三个键都未按
1	按下 Shift 键
2	按下 Ctrl 键
3	同时按下 Shift 和 Ctrl 键
4	按下 Alt 键
5	同时按下 Shift 和 Alt 键
6	同时按下 Ctrl 和 Alt 键
7	同时按下三个键

同样可以通过 **Select Case** 语句判断 **MouseDown** 事件过程中 **Shift** 参数的返回值，来获取用户所按下的组合键。用户可参照实例 4.1 自行编制一个小程序来熟悉 **Shift** 参数的使用。

3. Mouse Move 事件

窗体的 **Mouse Move** 事件过程的形式如下：

```
Private Sub Form_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
End Sub
```

MouseMove 事件过程中参数的含义及其用法与MouseDown 事件过程中的相应参数完全相同。这里给出一个探测鼠标位置的实例。

实例 4.2 探测鼠标的位置

在该程序中，当用户在窗体上移动鼠标时，则在窗体上的文本框中会显示出当前鼠标的位置。

在窗体的左上角放置一个文本框，如图 4.4 所示。窗体与控件的属性设置如表 4.6 所示。



图 4.4 实例 4.2 的窗体设计

表 4.6 实例 4.2 中对象的属性设置

对象	属性	值
窗体	Caption	探测鼠标的位置
文本框	名称	TexMove
	Text	置空

打开窗体的【代码】窗口，将下列代码添加到 Form_MouseMove 事件过程中：

```
Private Sub From_MouseMove (Button As Integer,Shift As Integer,X As
Single,Y As Single)
Texts1.Twxt="X '& X &"Y=" & Y
End sub
```

由于在移动鼠标时，**MouseMove** 事件不断被触发，其中的代码也就不断地执行。因此，在移动鼠标时，文本框中的内容会不断被更新。

运行该程序，在窗体中移动鼠标，则文本框中动态显示出鼠标的位置，如图 4.5 所示。



图 4.5 实例 4.2 的运行效果

VB6.0 基础教程 4.2.2 键盘事件

键盘事件有共 3 种，分别是 **KeyDown**（按下键）、**KeyUp**（释放键）与 **KeyClick**（敲击键）。

注意：只有当窗体为当前活动窗体时，按键才能触发窗体的键盘事件。另外，如果窗体上有能获得焦点的控件，则按键触发的将是控件的键盘事件。如果希望按键后，总是能触发窗体的键盘事件，应该将窗体的 **KeyPreview** 属性设置为 **Ture**。

1.KeyClick 事件

窗体的 **KeyClick** 事件过程的形式如下：

```
Private Sub Form_KeyPress (KeyAscii As Integer)
End Sub
```

其中参数 **KeyAscii** 是一个整数，用来返回用户所按键的 ASCII 码。利用该参数可以判断出用户按的是哪一个键。

实例 4.3 显示所按键的 ASCII 码

运行该程序，当用户按键盘上某键时，在窗体上显示用户所按键的 ASCII 码，例如，按回车键，则在窗体中显示"所按键的 ASCII 码值是：13。

打开【代码】窗口，将下列代码添加到 Form_KeyPress 事件过程中：

```
Private Sub Form_KeyPress (KeyAscii As Integer)
```

```
Print"所按键的 ASCII 码值是："&KeyAscii
```

```
End Sub
```

运行该程序，按键盘中某键，则窗体上就会显示出所按键的 ASCII 码，如图 4.6 所示的是按下 Enter 键和 q 键的效果。表 4.7 中列出了键盘按键的 ASCII 码。



图 4.6 实例 4.3 的运行效果

表 4.7 键盘按键的 ASCII 码表

按键	ASCII 码	按键	ASCII 码
Back Space (退格键)	8	- (减)	45
Tab (制表键)	9	. (小数点)	46
Enter (回车键)	13	/	47
Esc (取消键)	27	0~9	48~57
Space (空格键)	32	, (逗号)	58
'	39	; (分号)	59
(	40	< (小于)	60
)	41	= (等号) > (大于)	61
* (乘号键)	42	> (大于)	62
+ (加号键)	43	? (问号)	63
, (逗号键)	44	A~Z	65~90
		A~z	97~122

2. KeyDown 与 KeyUp 事件过程

KeyDown 与 KeyUp 事件过程的形式相同，如下所示：

```
Private Sub Form_KeyCode As Integer, Shift As Integer)
```

```
End Sub
```

```
Private Sub Form_keyUp (keyCode As Integer, Shift As Integer)
```

```
End Sub
```

KeyDown 与 KeyUp 事件过程中的 Shift 参数与 MouseDown 事件过程中的 Shift 参数的含义相同。参数 KeyCode 则是用来返回按键的键码。

功能键、换档键以及编辑键等没有 ASCII 码，但所有的键都有一个键码。每一个键（不是字符）对应一个键码，大键盘上的数字键和小键盘上的数字键的键码也不同。字母键的键码就是其大写字母的 ASCII 码。对于有上下档的按键，其键码是下档字符的 ASCII 码。表 4.8 中列出了一些按键的键码。

表 4.8 一些按键的键码

按键	键码	按键	键码
F1~F10	112~121	End	36
Back Space	8	Ins	45
Tab	9	Del	46
Enter	13	←	37
Esc	27	↑	38
Page Up	33	→	39
Page Down	34	↓	40
Home	35		

读者可参照实例 4.3, 自行编写一个可以显示用户所按键键码的程序。

VB6.0 基础教程 4.2.3 其他事件

除鼠标与键盘事件外，窗体对象还有其他一些事件，如表 4.9 所示。

表 4.9 窗体所能响应的其他事件

事件	说明
Load	在将窗体装入内存时发生，该事件过程主要用来进行一些初始化操作
Activate	当窗体变为活动窗体时发生
DeActivate	与 Activate 相对应，当窗体由活动状态变成不活动状态时，则该事件发生
QueryUnload	在用户关闭窗体时，QueryUnload 最先发生
Unload	将窗体从内存中卸载时发生，它发生在 QueryUnload 事件以后
Resize	在窗体初次装载或用户改变其大小后发生。该事件较常用

可以在【代码】窗口中查到窗体所支持的所有事件。在【代码】窗口的【对象】框中选择 Form,单击【事件】框，即可弹出窗体的事件列表，如图 4.7 所示。



图 4.7 窗体的事件列表

一些应用程序，当用户改变窗体的大小后，如拖动窗体边框或最大化等，则窗体中的控件也随着改变以适合窗体的大小。这里，我们编写一个程序来模拟这种情形。

实例 4.4 窗体的 Resize 事件

在该程序中，当用户改变窗体的大小时，则窗体中按钮的大小也将成比例地改变，并且按钮始终处于窗体的中心。

在窗体中放置一个按钮控件，如图 4.8 所示。窗体与按钮的属性设置如表 4.10 中所示。



图 4.8 实例 4.4 的窗体设计

表 4.10 实例 4.4 中各对象的属性设置

对象	属性	值
窗体	名称	ForSize
	Caption	窗体的 Resize 事件
按钮	名称	ComSize
	Caption	按钮

打开【代码】窗口，将下列代码添加到 Form- Resiae 事件过程中：

```
Private Sub Form_Resize()

    ComSize.width=ForSize.Width/5

    ComSize.Height=ForSize.Height/6

    ComSize.Top=ForSize.Height/2-ComSize.Height/2

    ComSize.Left=ForSize.width/2-ComSize.Width/2

End Sub
```

在该段代码中，我们将按钮的宽度设置为窗体宽度的 1/5,按钮的高度设置为窗体高度的 1/6,并将按钮的位置设置在窗体的中心。

运行该程序，则窗体如图 4.9 所示。可见，在窗体启动后，窗体的 Resize 事件就被触发了。使用鼠标拖动窗体的边界来改变它的大小，或单击最大化按钮使窗体最大化，可以发现，窗体中按钮的大小也随着成比例地改变，并且始终处于窗体的中心，如图 4.10 所示。



图 4.9 窗体启动后的效果



图 4.10 改变窗体大小后的效果

由于 QueryUnload 事件在窗体卸载之前发生，因此，可以通过编写 QueryUnload 事件过程来完成一些工作，如文件的保存等。该事件在 MDI 应用程序中相当重要，这在第七章中还会讲到。这里，只是通过一个实例介绍 QueryUnload 事件的使用。

实例 4.5 窗体的 QueryUnload 事件

打开窗体的【代码】窗口，在事件列表中选择 QueryUnload 事件，则 QueryUnload 事件过程的框架就出现在代码编辑区中，形式如下：

```
Private Sub Form_QueryUnload (Cancel As Integer,Unload (Cancel
As Integer, UnloadMode As Integer)
End Sub
```

其中参数 Cancel 是一个整数。若在 QueryUnload 事件过程中给此参数赋一个非零值 (True),则将阻止窗体的关闭;若赋 0 值 (False),则将关闭窗体。如果忽略此参数，窗体将被关闭。

UnloadMode 参数是事件的返回值，它表示引起 QueryUnload 事件的原因。表 4.11 中列出了该参数的返回值及其含义。

编写 QueryUnload 事件过程如下：

```
Private Sub Form_QueryUnload(Cancel As
```

```
Integer,UnloadMode As Integer)

    If UnloadMode=0 Then

        Print “单击关闭按钮关闭我，休想！”

        Cancel=Ture

    End If

End Sub
```

运行该程序，单击窗体右上角的【关闭】按钮，窗体并不关闭，并且显示"单击关闭按钮关闭我，休想！"，如图 4.11 所示。



图 4.11 单击【关闭】按钮的情形

VB6.0 基础教程 4.3.1 Print 方法

Print 方法是用来输出数据和文本的一个重要方法。除窗体对象外，图片框控件也有 **Prim** 方法。本节将详细介绍 **Print** 方法以及相关的输出格式。

Print 方法的一般格式为：

对象名。**Print** 表达式

表达式可以是数值也可以是字符串。对于数值表达式，先计算出表达式的值，然后输出；字符串表达式将按原样输出，并且，字符串一定要放在双引号内。如果忽略表达式，则输出一个空行。

例如，编写窗体的 **Click** 事件过程如下：

```

Private Sub Form Click()

x=20

y=30

Print x

Print y

Print x+y

Print

Print “ABcdEFgh”

Print “清华大学”

End Sub

```

程序运行后单击窗体，则在窗体上的输出结果如图 4.12 所示。



图 4.12 使用 Print 语句

也可以使用一个 **Print** 语句输出多个表达式。各表达式之间需要用分隔符隔开。分隔符可以是逗号、分号、空格或&符号。如果表达式使用逗号分隔，在输出时，各表达式之间间隔 14 个字符的位置。如果使用其他几种分隔符，则表达式将按紧凑格式输出。

例如，将上例代码改写如下：

```

Private Sub Form_Click( )

x=20

```

```

y=30

Print "x+y" , x+y

Print "x+y" ; x+y

Print

Print x,y "清华大学" ; " ABcdEFgH " & x+y

End Sub

```

程序运行后单击窗体，则在窗体上的输出结果如图 4.13 所示。

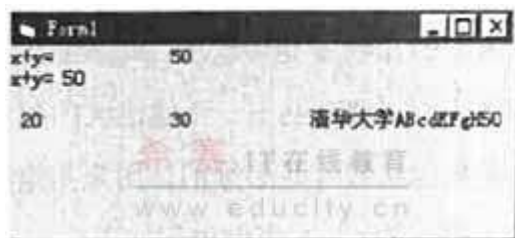


图 4.13 在 Print 语句中使用分隔符

在一般情况下，每执行一次 Print 方法都会自动换行，即后一个 Print 语句的执行结果总是显示在前一个 Print 语句的下一行。为了仍在同一行上显示，可以在 Print 语句的末尾加上逗号或分号。

例如，将上例代码改写如下：

```

Private Sub Form_Click()

x= 20

y= 30

Print "x+y= " ,

Print x+y

Print

Print "x+y= " ;

```

```
Print x+y
```

```
End Sub
```

程序运行后单击窗体，则在窗体上的输出结果如图 4.14 所示。



图 4.14 在同一行输出

从上面的输出结果中可以看出，输出内容总是显示在窗体的最左端。可以人为地在 `Print` 语句中加一些空格来确定内容的输出位置，例如：

```
Print"清华大学 计算机系"
```

但这不是一个好办法。VB 提供了两个专门的函数：`Tab (n)` 和 `Spc (n)`，它们与 `Print` 方法一起使用，就可以在指定的位置输出内容。

`Tab (n)` 函数的参数 `n` 是可选的，用来指定表达式输出时的起始列数。若忽略此参数，则将输出点移动到下一个输出区的起点。

`Spy (n)` 函数的参数 `n` 是必须的，用来指定输出表达式之前插入的空格数。

例如，编写窗体的 `Click` 事件过程如下：

```
Private Sub Form_Click()
```

```
Print Tab(20); “清华大学”
```

```
Print
```

```

Print Tab(6); “清华大学” ; Spc (10); “计算机
系”

Print Tab(7); “清华大学” ; Spc (11); “计算机
系”

Print Tab(8); “清华大学” ; Spc (12); “计算机
系”

Print Tab(9); “清华大学” ; Spc (13); “计算机
系”

End Sub

```

运行程序后单击窗体，则输出结果如图 4.15 所示。

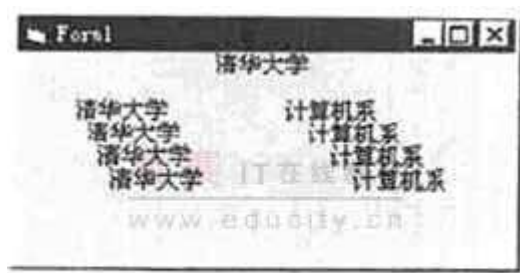


图 4.15 使用 Tab 和 Spc 函数

VB6.0 基础教程 4.3.2 Cls 方法

Cls 方法用来清除由 Print 方法在窗体上显示的文本或使用图形方法（参见第 9 章）在窗体上绘制的图形。图片框控件也有 Print 与 Cls 方法。

Cls 方法的语句很简单，如下所示：

【对象名】 . Cls

如果对象名缺省，则表明清除本窗体上的内容。这一点对于其他方法也适用。

实例 4.5 使用 Cls 方法

在该程序中，当用户单击窗体时，则在窗体上显示一行文本，当用户双击窗体时，窗体上的文本即被清除。

打开【代码】窗口，编写窗体的 Click 与 DblClick 事件过程如下：

```
Private Sub Form_Click ()  
Print"双击可以清除窗体的内容"  
End Sub  
  
Private Sub Form_DblClick ()  
Cls  
End Sub
```

运行该程序，单击窗体，则窗体上显示"双击可以清除窗体中的内容",如图 4.15 所示。双击窗体，则文本被清除，如图 4.17 所示。再次单击窗体，则文本仍然显示在窗体的最上面，而不会显示在下一行。



图 4.16 单击窗体的效果



图 4.17 双击窗体的效果

VB6.0 基础教程 4.3.3 Move 方法

窗体与大多数控件（如按钮控件、文本框控件等）都有 **Move** 方法，使用该方法可以使对象移动，在移动的同时还可以改变对象的大小。

Move 方法的一般格式为：

[对象名. **JMove Left [Top],[Width],[Height]**

Move 方法有 4 个参数，其中参数 **Left** 与 **Top** 分别是指对象左上顶点的横坐标与纵坐标，参数 **Width** 与 **Height** 分别是指对象的宽度与高度。参数 **Left** 是必需的，其他参数是可选的。

实例 4.7 使用 **Move** 方法

在该程序中，每当用户单击一次鼠标，则窗体向它的右下方移动一定的距离并且越来越小。

打开【代码】窗口，编写窗体的 **Click** 事件过程如下：

```
Private Sub Form_Click ()  
  
Move Left + 250,Top=300,Width=250,Height=250  
  
End Sub
```

在该段代码中，**Left**、**Top** 等是窗体的属性，这里省略了窗体名。

窗体还有 **Show** 与 **Hide** 等方法，它们的功能分别是显示与隐藏窗体，下一节将专门介绍有关窗体的加载、显示、隐藏与卸载等内容。

VB6.0 基础教程 4.4.1 窗体的操作

在应用程序启动时，会显示它的一个主窗体，在运行过程中，常常还会显示其他窗体或隐藏某些窗体。在前面的讲解中，涉及到的只是一个窗体，在程序运行时它自动显示出来。如果应用程序包含若干

个窗体，在启动时，只显示其中的一个窗体（启动窗体），而其他窗体的显示则需要使用相应的语句来执行。

窗体的状态有 3 种：

未装入：窗体在磁盘文件中，不占用内存资源。

装入但未显示：窗体已装入内存，占用了所需资源，准备显示。

显示：窗体已显示，用户可以对窗体进行交互操作。

可以使用窗体的以下几个方法来在应用程序中实现窗体的加载、显示、隐藏与卸载。

Load 与 Unload 要装载御载某个窗体，需要使用 Load/unload 方法，装载御载窗体的语句如下：

Load 窗体名。

Show 与 Hide 要显示/隐藏某个窗体，需要使用 Show/Hide 方法，显示/隐藏窗体的语句如下：

窗体名。show/Hide.

例如，要显示窗体 Form2,语句如下：

Form2_Show.

调用 Show 方法与设置窗体 Visible 属性为 True 具有相同的效果。语句 Form2.Visible=True 也可以使窗体 Form2 显示出来。

使用 Load 方法将窗体装载后，窗体并不显示出来。而使用 Show 方法则可以使指定的窗体显示在最上面。单独使用 Show 方法也可以将窗体装载并显示。那么窗体装载的意义又在哪呢？单独装载窗体的原因有两个：

这里以一个实例来学习如何设置启动窗体，以及窗体的装载与显示。

对于多重窗体应用程序，需要指定程序运行时的启动窗口。其他窗体的装载与显示由启动窗体控制。在默认情况下，系统会以创建的第一个窗体为启动窗体。如果想指定其他窗口为启动窗口，打开【工程】菜单，执行【工程属性】命令，出现如图 4.18 所示的【工程属性】对话框的【通用】选项卡。在【启动对象】列表框中列出了当前工程的所有窗体，从中选择要作为启动窗体的窗体后，单击【确定】按钮即可。

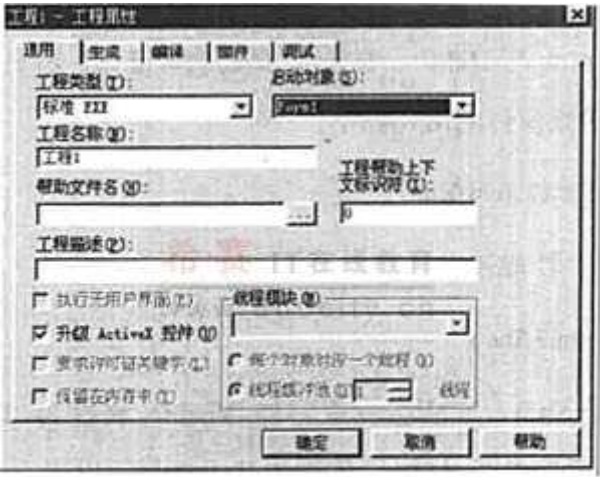


图 4.18 【工程属性】对话框的【通用】选项卡

这里以一个实例来学习如何设置启动窗体，以及窗体的装载与显示。

实例 4.8 多窗体应用程序。

使用【工程】菜单中的【添加窗体】命令为当前工程添加两个新的窗体。它们的属性设置如表 4.12 所示。

表 4.12 实例 4.8 中窗体的属性设置

对象	属性	值
窗体 1	名称	ForMain
	Caption	启动窗体
窗体 2	名称	ForSub1
	Caption	窗体 1
窗体 3	名称	ForSub2
	Caption	窗体 2

在主窗体上放置 3 个按钮，各按钮控件的属性设置如表 4.13 所示。

表 4.13 启动窗体中按钮属性的设置

对象	属性	值
按钮 1	名称	ComOpen
	Caption	打开窗体
按钮 2	名称	ComClose1
	Caption	关闭窗口 1
按钮 3	名称	ComClose2
	Caption	关闭窗口 2

双击【打开窗体】按钮，打开【代码】窗口，将下列代码添加到 ComOpen_Click 事件过程中：

```
Private Sub ComOpen_click ()
    ForSub1.show
    ForSub2.show
End Sub
```

将下列代码添加到 ComClose1_Click 事件过程中：

```
Private Sub ComClose_Click ()
    Unload ForSub1
End Sub
```

将下列代码添加到 ComClose2_Click 事件过程中：

```
Private Sub ComClose2_Click ()
```

Unload ForSub2

End Sub

在【工程属性】对话框的【通用】选项卡中设置启动窗体为 ForMain.

单击工具栏中的【运行】按钮运行该程序，则屏幕上出现启动窗体，单击【打开窗体】按钮，则窗体 1 与窗体 2 出现在屏幕上。

单击【关闭窗体 1】按钮与【关闭窗体 2】按钮，则分别可关闭窗体 1 与窗体 2.

VB6.0 基础教程 4.5 设置窗体的位置

在前面的一些实例中，窗体启动后，它的位置是随意的。对于友好的用户界面，窗体在启动后一般总是出现在屏幕的中心。用户可以通过窗体的属性或【窗体布局】窗口来设置窗体启动后的默认位置。

窗体的位置是相对于屏幕而言的，且由 Left 属性和 Top 属性来决定它的位置。其中 Left 属性的值是指窗体左上顶点的横坐标，Top 属性的值是指它左上顶点的纵坐标。通过设置这两个属性的值就能实现窗体的精确定位。

此外，还可以通过【窗体布局】窗口来设置窗体的位置，这种方法更直观。【窗体布局】窗口通常不出现在 VB 的主窗口中，可以通过执行【视图】菜单中的【窗体布局】窗口命令来打开它。

在【窗体布局】窗口中有一个"显示器",在"显示器"中有一个窗体图标，如果当前工程包含多个窗体，则"显示器"中也会出现多个窗体

图标。窗体图标在"显示器"中的位置就是窗体启动后出现在屏幕上的位置。因此,通过【窗体布局】窗口就可以预览到窗体启动后的效果。

将鼠标指针移动到"显示器"中的窗体图标上,指针的形状将变成指向四个方向的箭头状,拖动鼠标,则窗体图标也会随着移动,将它拖动到合适的位置后释放鼠标。则窗体的启动位置就被改变了。窗体的 Left 与 Top 属性的值也会自动调整为该位置所对应的值。

VB6.0 基础教程 4.6 创建工具栏

工具栏已经成为许多基于 Windows 的应用程序的标准功能,它提供了对应用程序中最常用的菜单命令的快速访问。工具栏一般处于菜单栏下面,由多个按钮排列组成,用户可以通过单击这些按钮来执行一些操作。与菜单相比,使用工具栏更方便快捷。

要创建工具栏,需要两个控件:工具栏控件(Tollbar)与图像列表控件(ImageList)。在 Visual Basic 的专业版与企业版中都提供了这两个控件。工具栏控件。设置工具栏按钮与处理用户的操作,图像列表控件则负责提供在按钮上显示的图标。工具栏的整个设计过程可分为下面几个步骤:

- (1) 将工具栏控件与图像列表控件添加到工具箱中。
- (2) 将工具栏控件与图像列表控件放置到窗体上。
- (3) 向图像列表控件添加图片。
- (4) 使用工具栏控件建立按钮。
- (5) 编写按钮的程序代码。

VB6.0 基础教程 4.6.1 添加工具栏与图像列表控件

在默认情况下，工具栏与图像列表控件不出现在工具箱中，在使用它们前，用户需要将它们添加到工具箱中。

执行【工程】菜单中的【部件】命令，出现如图 4.22 所示的对话框。在控件列表中选中 Microsoft Windows Common Controls 6.0 选项，单击【确定】按钮。可以发现，工具箱中多了 9 个控件，其中包括工具栏与图像列表控件。



图 4.22 【部件】对话框

VB6.0 基础教程 4.6.2 向图像列表控件添加图片

图像列表控件不能独立使用，只是作为一个便于向其它控件提供图像的资料中心。它需要第二个控件显示所储存的图像。第二个控件可以是任何能显示图像的控件。正是由于图像列表控件的这一特点，它常常被用来与工具栏控件一起创建工具栏。

工具栏通常用图标代表应用程序的功能。例如，软盘的图标一般代表"保存文件"功能。要使工具栏能够显示这样的图标，可以首先将所要的按钮图标添加到图像列表控件中。然后将图像列表控件与工具栏控件相关联。本节将介绍如何向图像列表控件中添加图片。

将图像列表框控件放置在窗体上，在运行时，该控件不出现在界面中，因此，不必在意它在窗体中的位置。将鼠标移动到图像列表框控件上。单击右键，在弹出的快捷菜单中执行【属性】命令，弹出【属性页】对话框。

选择【图像】选项卡。单击其中的【插入图片】按钮，打开【选定图片】对话框，在该对话框中选择某一图片，单击【打开】按钮即可将该图片添加到图像列表控件中。重复插入图片操作，可以为图像列表控件添加多个图片。单击【确定】按钮即可完成操作。

在添加了图片后，系统自动为每个图片设置了一个索引号，第一个添加的图片的索引号为 1,第二个为 2,依次类推。图片的索引号很重要，在工具栏控件与图像列表控件关联时，就是以图片索引号来调用各图片的：也可以使用关键字来调用图片，因此，最好每一个图片指定一个唯一的關鍵字；在【图像】列表框中单击选中某个图片，单击【删除图片】按钮可将该图片从图像列表控件中删除。

这里为图像列表控件添加 3 个图片，如图 4.25 所示，以便在以后创建工具栏时使用。表 4.14 中列出了添加的图片的文件名称以及对应的索引号和关键字。

表 4.14 添加的图片及设置

文件名称	索引号	关键字
Open.bmp	1	Lopen
Save.bmp	2	Lsave
Print.bmp	3	Lprint

提示：如果在安装 VB 时选择可安装图片，则在 VB 的安装目录 \Common\Graphics\Bitmaps\TIBR_95 文件夹中包含了大量的 Windows 的标准按钮图标。

VB6.0 基础教程 4.6.3 使用工具栏控件

将工具栏控件放置在窗体上。工具栏控件总是出现在窗体的上方，并且不能改变它的大小与位置。这是因为在默认情况下，工具栏控件的 `Alignment` 属性的值为 `1-vbAlignTop`。通过设置该属性，还可以使得工具栏沿窗体的其他边对齐。例如，将 `Alignment` 属性的值设置为 `2 -vbAlignBottom`，则工具栏沿窗体的底边对齐。如果要创建一个浮动的工具栏，可以设置 `Alignment` 属性的值为 `0-vbAlignNone`。用户可以调整它的大小与位置。

工具栏控件的【属性】窗口中还有其他一些重要的属性。在工具栏控件的【属性页】对话框中也可以设置控件的属性，并且更直观。对于初学者，建议通过【属性页】对话框设置属性。

将鼠标移动到工具栏控件上，单击右键，弹出一个快捷菜单，执行其中的【属性】命令即可打开如图 4.27 所示的【属性页】对话框。单击【图像列表】的下三角按钮，在下拉列表中选择 `ImageList1` 选项（`ImageList1` 是在前面放置在窗体上并添加了图片的图像列表控件），这样就建立了工具栏控件与图像列表控件的关联。

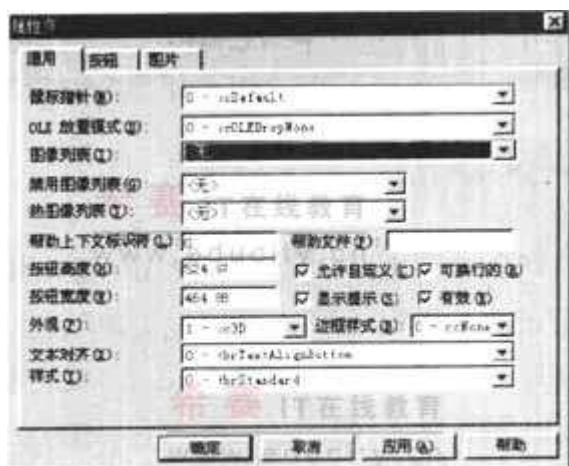


图 4.27 工具栏控件的【属性页】对话框

其他一些较为重要的属性的含义如下：

允许自定义（AllowCustomize）属性决定用户是否可以通过双击工具栏打开【自定义工具栏】对话框栏重新设置工具栏。

显示提示（ShowTips）属性确定鼠标停留在按钮上时是否显示工具提示。

可换行的（Wrappable）属性确定若在一行内容纳不下全部按钮时，是否以两行显示按钮。

有效（Enabled）属性确定按钮是否可用。

注意：括号内的英文在【属性】窗口中显示的属性。

在【属性页】对话框的【通用】选项卡中还可以设置工具栏的外观属性，如外观、边框和样式等。在设置这些属性后，单击【应用】按钮，即可在窗体中预览到设置的效果。读者可自行试一试更改各外观属性后的效果。

在【通用】选项卡中设置的是有关整个工具栏的属性，要为工具栏建立按钮，需要在【按钮】选项卡中执行。

打开【按钮】选项卡，如图 4.28 所示。单击【插入按钮】按钮，通过在其中设置一些按钮的属性，即可在工具栏中建立一个按钮。重复插入按钮操作，可以为工具栏建立多个按钮。按钮的一些重要属性的含义如下：



图 4.28 【按钮】选项卡

按钮就好象数组中的元素一样，在程序中可以通过它们的索引号来引用。

例如：

`Toolbar1.Buttons(1).Caption="打开"`该语句是将索引号为 1 的按钮的标题设置为"打开".

标题（Caption）属性用来设置要在按钮上显示的文本，如果不输入任何内容，则按钮上只显示图标，不显示文本。大多数工具栏中的按钮上都不显示文本。

关键字（Key）属性是指按钮的名称，在程序中也可以以关键字来引用按钮。

样式（Style）属性用来设置按钮的类型。表 4.15 中列出了该属性的取值及对应的按钮类型。

工具提示文本 (ToolTipText) 属性用来设置当鼠标停留在按钮上时, 显示的工具提示。

图像 (Image) 属性指定在按钮上显示的图片的索引号或关键字。其中图片的索引号与关键字是在图像列表控件的【属性页】对话框中指定的。

表 4.15 Style 属性的取值

取值	按钮类型	举例
0 (默认)	标准按钮, 即单击按钮后, 按钮会自动弹回原状	Word 中的新建按钮
1	按钮在按下后保持下沉状态, 再次单击时才恢复原状	Word 中的黑体按钮
2	一组中的一个按钮, 每次只能有一个按钮被按下	Word 中的对齐按钮
3	用于在按钮之间提供间隔的按钮, 它具有 8 个像素的宽度	Word 中的字体组合框
4	在按钮之间保留一个空间, 以建立其他非按钮的工具栏成员。可以在后面的【宽度】文本框中设置要保留空间的宽度	VB 中的添加窗体按钮
5	下拉按钮, 单击按钮右边的向下箭头会弹出一个按钮列表。需要同时在【按钮菜单】设置区中创建按钮列表中包含的按钮	

这里为工具栏建立 3 个按钮。它们的属性设置如表 4.16 所示, 其他属性均采用默认设置。

表 4.16 工具按钮的属性设置

按钮	关键字	工具提示文本	图像
打开	Bopen	打开文档	1 或 Lopen
保存	Bsave	保存文档	2 或 Lsave
打印	Bprint	打印文档	3 或 Lprint

到此, 在窗体的工具栏控件上出现了 3 个按钮。运行该程序, 将鼠标停留在某个按钮上, 会显示出该按钮的工具提示文本。但是, 单击按钮不会执行任何操作, 这是因为还没有为该按钮编写事件过程。

VB6.0 基础教程 4.6.4 为工具栏编写代码

本节将为上一节中所创建的工具栏编写代码, 使按钮能执行一定的操作。

在前面创建了工具栏的窗体中放置一个文本框, 如图 4.31 所示。各对象的属性设置如表 4.17 所示。

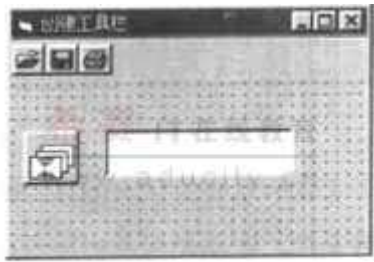


图 4.31 在窗体中放置一个文本框

表 4.17 对象的属性设置

对象	属性	值
窗体	Caption	创建工具栏
文本框	名称	Text1
	Text	置空

在运行模式下，每当用户单击工具栏中的按钮时，就会触发该工具栏的 ButtonClick 事件。因此，为工具栏编写代码其实就是编写它的 ButtonClick 事件过程。

双击工具栏，打开【代码】窗口，工具栏的 ButtonClick 事件过程的框架自动出现在【代码】窗口中，编写该过程如下：

```
Private Sub Toolbar1_ButtonClick(Byval Button As
MSComctlLib.Button)
Select Case Button.Key
Case "Bopen"
Text1.Text="文档打开成功！"
Case "Bsave"
Text1.Text="文档保存完毕！"
Case "print"
Text1.Text="正在打印文档..."
```

```
End Select
```

```
End Sub
```

在代码中，使用了 **Select Case** 语句来判断按钮的关键字。对于不同的按钮，单击后在文本框中显示不同的内容。

运行该程序，单击某按钮，在文本框就会显示出与该按钮相关的内容。

状态栏一般用来显示系统的状态信息。在 **Vg** 中可以使用状态栏控件方便地为应用程序创建状态栏。在添加工具栏控件时，状态栏控件也一同被添加到工具箱中。

状态栏的创建与工具栏的创建有很多类似的地方，这里只是给出一个创建状态栏的实例。

实例 4.9 创建状态栏

在该实例中，创建如图 4.32 所示的用户界面。其中状态栏分为三个窗格，在第一格内显示系统日期；第二格能根据鼠标指针的位置显示相应的内容；第三格显示 **Caps Lock** 键的状态。

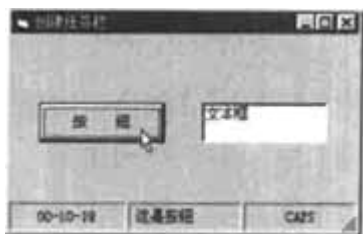


图 4.32 实例 4.9 的用户界面

在窗体上放置一个按钮控件、一个文本框控件和一个状态栏控件，如图 4.33 所示。其中各对象的属性设置如表 4.18 所示。在将状

态栏控件放置在窗体上时，它自动出现在窗体的底边。也可以使用 Alignment 属性来设置状态栏的位置。

表 4.18 实例 4.9 中各对象的属性设置

对象	属性	值
窗体	Caption	创建状态栏
按钮	名称	按钮
	Caption	Command1
文本框	名称	Text1
	Text	文本框
状态栏	名称	StatusBar1

将鼠标移动到状态栏控件上，单击右键，弹出一个快捷菜单，执行其中的【属性】命令，即可打开状态栏的【属性页】对话框，如图 4.34 所示。



图 4.33 实例 4.9 的窗体设计

图 4.34 状态栏的【属性页】对话框

在本例中使用【通用】选项卡中的默认设置。单击【窗格】选项卡。在【索引】文本框中显示数字 1,表示以下各属性是针对第一个窗格而言的。在程序中通过窗格的索引号或关键字来引用它，这一点与引用工具栏的按钮相同。

窗格的样式 (Style) 属性是很重要的一个属性，它决定在窗格中显示的内容。表 4.19 中列出了样式属性的取值及其含义。

表 4.19 样式属性的取值

常数	对应值	含义
SbrText	0	显示文本或位图。显示的文本由文本属性 (Text) 设置, 位图由图片 (Picture) 属性设置
SbrCaps	1	显示 Caps Lock 键的状态。当该键处于打开状态时, 窗格中以黑体字显示 CAPS; 处于关闭状态时, 窗格中以灰体字显示 CAPS
SbrNum	2	显示 Num Lock 键的状态。当该键处于打开状态时, 窗格中以黑体字显示 Num; 处于关闭状态时, 窗格中以灰体字显示 Num
SbrIns	3	显示 Insert 键的状态。当该键处于打开状态时, 窗格中以黑体字显示 INS; 处于关闭状态时, 窗格中以灰体字显示 INS
SbrScrl	4	显示 Scroll Lock 键的状态。当该键处于打开状态时, 窗格中以黑体字显示 SCRL; 处于关闭状态时, 窗格中以灰体字显示 SCRL
SbrTime	5	显示当前的时间
SbrDate	6	显示当前的日期

窗格的样式 (style) 属性可以在设计阶段设置, 也可以在运行阶段通过代码设置。例如 `StatusBar1.Panels (1).Style=sbfTime` 语句是将状态栏的第一个窗格的样式属性的值设置为 `sbrTime`, 即在该窗格内显示当前的系统时间。

在本例中, 由于在第一个窗格内显示的是系统日期, 因此, 在【样式】列表框中选择 `6-SbriDate` 选项。同时, 在【对齐】列表框中选择 `1-sbrCenter` 选项, 即居中显示。单击【应用】按钮, 在状态栏上就显示出当前的系统时间。

在默认情况下, 状态栏只包含一个窗格。使用【插入窗格】按钮可在状态栏中添加窗格。单击【插入窗格】按钮后, 就会发现在状态栏中多了一个窗格。

注意: 插入窗格后在【窗格】选项卡中看不出任何变化, 窗格的索引号仍然是 1, 用户可以通过单击【索引】文本框后的箭头按钮来切换窗格。不要在单击【插入窗格】按钮后就开始设置属性, 那样设置的仍然是第一个窗格的属性。

单击【索引】文本框后的向右箭头，此时，【索引】文本框中的索引号变为 2,表明以下各属性是针对第二个窗格而言的。在本例中，第二个窗格显示的是文本。因此，使用样式属性的默认值就可以了。在【文本】文本框中输入"欢迎"两个字，单击【应用】按钮后这两字将显示在窗格中。在程序中可以也使用窗格的 Text 属性来设置要显示的内容。

重复插入窗格的操作，向状态栏中添加第三个窗格，并且在【样式】列表框中选择 1- SbrCaps 选项,在【对齐】列表框中选择 1-sbrCenter 选项。单击【确定】按钮，即可关闭【属性页】对话框，完成状态栏的设置。

接下来的工作就是为状态栏添加代码。对于本例中创建的状态栏，只需为第二个窗格添加代码，使它能根据鼠标指针的位置显示出相应的内容。其他两个窗格无需编写任何代码。

双击按钮控件，打开【代码】窗口，编写按钮控件的 MouseMove 事件过程如下

```
Private Sub Command1_MouseMove(Button As Integer,shift As Integer,X As Single,Y As Single)
    StattusBartl.Panels(2).Text="这是按钮"
End Sub
```

该段代码的作用是：当鼠标移动到按钮上，在状态栏的第二个窗格中显示"这是按钮"几个字。

同理，编写窗体与文本框的 **MouseMove** 事件过程如下：

```
Private Sub Form_MouseMove(Button As Integer,shift As Integer,x  
As Single,Y As Single)  
StatusBar1.Panels(2).Text="欢迎"  
End Sub  
  
Private Sub Text_MouseMove(Button As Integer,shift As Integer,x  
As Single,Y As Single)  
StatusBar1.Panels(2).Text="这是文本框"  
End Sub
```

这样，一个简单的状态栏就创建完毕。运行程序，出现界面。在状态栏的第一格中显示的是当前的系统时间：第二格中显示"欢迎"两个字；第三格中以灰色显示 **CAPS**,表明当前的 **Caps Lock** 键处于关闭状态。

将鼠标指针移动到按钮上，第二个窗格中的内容变成"这是按钮",将鼠标指针移动到文本框上，则变成"这是文本框".若鼠标指针处于窗体的其他位置，窗格中恢复显示"欢迎".按 **Caps Lock** 键，第三窗格中的 **CAPS** 字符的颜色会在黑色与灰色之间切换。图 4.40 所示的是将鼠标移动到文本框上，并使 **Caps Lock** 键处于打开状态时的情形。

第五章 基本控件的使用

VB6.0 基础教程 5.1 标签控件

在 Windows 应用程序的各种对话框中，都显示有一些文本提示信息，在 VB 中可以使用标签控件来实现在窗体中显示这些文本提示信息。表 5.1 中列出了标签控件的一些主要属性。

表 5.1 标签控件的一些主要属性

属性	值
Alignment	用于指定在标签上显示文本的位置。值为 0 表示左对齐；值为 1 表示右对齐；值为 2 表示居中
AutoSize	值为 True 时，标签框将按照其所显示内容的多少来自动调整大小；值为 False 时，超出标签框的内容将显示不出来
BackStyle	设置标签的背景是否透明。值为 0 时表示透明，值为 1 时表示不透明
BackColor	如果设置标签的背景是不透明的，则可用该属性设置背景的颜色
BorderStyle	设置标签框是否有边框，值为 0 时无边框，值为 1 时有边框
Caption	设置或返回标签框上显示的文本
Font	设置标签上文本的字体以及字号等

既可以在程序设计阶段通过【属性】窗口设置标签的属性，也可以在程序运行阶段在代码中设置窗体的属性。如将标签（名称为 Labell）显示的文本设置为"欢迎"的语句如下：

Labell.Caption=欢迎

实例 5.1 标签的使用

在该程序中，窗体上显示一行提示用户执行操作的文本，当用户单击或双击窗体时，窗体上还会显示出用户所执行的操作。

在窗体上放置两个标签控件，它们的属性设置如表 5.2 所示。

表 5.2 实例 5.1 中各对象的属性设置

对象	属性	值
窗体	名称	Form1
	Caption	标签的使用
标签 1	Caption	请您单击或双击窗体
	FontSize	14
标签 2	名称	Lab1
	AutoSize	True
	Caption	置空
	FontSize	14

打开【代码】窗口，将下列代码添加到 Form_Click 事件过程中：

```
Private Sub Form_Click()  
  
Labl.Borderstyle=0  
  
Labl.Caption="您单击了窗体！"  
  
End Sub
```

当单击窗体时，则触发 Form_Click 事件，该事件中的第一行语句是设置标签无边框（BorderStyle 属性的值为 0），第二行语句是设置标签上显示的文本。

与此类似，将下列代码添加到 Form_DblClick 事件过程中：

```
Private Sub Form_Click()  
  
Labl.Borderstyle=1  
  
Labl.Caption="您单击了窗体！"  
  
End Sub
```

运行该程序，单击窗体，则窗体上显示"您单击了窗体！"，如图 5.2 所示。双击窗体，则窗体上显示"您双击了窗体！"，并且文本有一个边框，如图 5.3 所示：

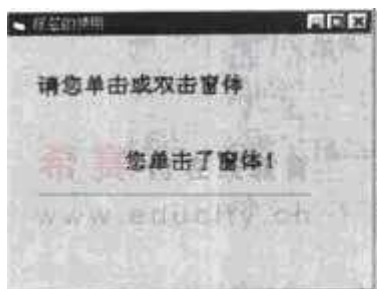


图 5.2 单击窗体后的效果

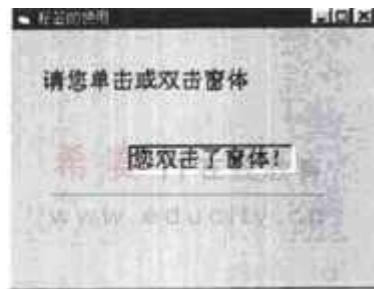


图 5.3 双击窗体后的效果

VB6.0 基础教程 5.2 按钮控件

在应用程序中，按钮控件常常被用来启动、中断或结束一个进程，用户可以通过简单的单击按钮来执行操作。只要用户单击按钮，就会触发它的 Click 事件过程，通过编写按钮的 Click 事件过程，就可以指定它的功能了。

按钮控件的常用属性如表 5.3 所示。

表 5.3 按钮控件的常用属性

属性	值
Enabled	设置按钮是否有效。当值为 True 时，按钮可用，当值为 False 时，按钮不可用，且以浅灰色显示
Default	设置缺省的活动按钮
Cancel	设置取消按钮
Caption	设置按钮上显示的文本
Picture	当 Style 属性的值为 1 时，该属性用于设置按钮上显示的图片
DownPicture	当 Style 属性的值为 1 时，该属性用于设置按钮被按下时显示的图片
DisabledPicture	当 Style 属性的值为 1 时，该属性用于设置按钮无效时显示的图片
Style	设置按钮的外观。值为 0 时，为标准的 Windows 按钮，值为 1 时，可以在按钮上放置图片

VB6.0 基础教程 5.2.2 多功能按钮

通常，每个按钮都有一个固定的标题（Caption）和一个特定的功能，用户也可以设计出多功能按钮。单击这样的按钮，按钮的名称会发生变化，并且会执行与按钮标题相应的操作。

实例 5.3 多功能按钮。

在该程序中，窗体上只有一个按钮，单击该按钮，按钮的标题会在"显示日期"与"显示时间"间切换，并且在窗体中将显示出与当前按钮标题相应的内容。

在窗体上放置一个标签控件和一个按钮控件，其中各对象的属性设置如表 5.5 所示。

表 5.5 实例 5.3 中各对象的属性设置

对象	属性	值
窗体	Caption	多功能按钮
标签	名称	Label1
	AutoSize	True
	BorderStyle	1
	Caption	置空
按钮	名称	Command1
	Caption	显示日期

双击【显示日期】按钮，打开【代码】窗口，将下列代码添加到 Command1_Click 事件过程中：

```
Private Sub Command1_Click()
    If Command1.Caption="显示日期"Then
        Label1.Caption=Date
        Command1.Caption="显示时间"
    Else
        Label1.Caption =Time
        Command1.Caption="显示日期"
    End if
End Sub
```

在该段代码中，使用了一个 If 语句来判断当前按钮的标题，然后做出相应的操作。

运行该程序，单击【显示日期】按钮，则在标签中显示当前的系统日期，并将按钮的标题改变为"显示时间",如图 5.9 所示；再次单击按钮。则在标签中显示当前的系统时间，并且按钮的标题恢复为"显

示日期",如图 5.10 所示。这样,通过一个按钮就可以循环显示当前系统的时间与日期了。



图 5.9 单击【显示日期】按钮的效果



图 5.10 单击【显示时间】按钮的效果

VB6.0 基础教程 5.2.3 使用键盘操作按钮

按钮控件的 Click 事件不仅可以由鼠标单击触发,还可以由以下几种方法触发:

按 Tab 键把焦点移到相应按钮上。然后按 Space 键或 Enter 键。

如果某按钮是窗体的缺省命令按钮,即使将焦点移到其他控件上(非按钮控件),按下 Enter 键也会选中该按钮。在设计时,通过设置按钮的 Default 属性为 True,就可指定它为窗体的缺省命令按钮。在一个窗体中只能有一个按钮为缺省命令按钮。

如果某按钮是窗体的缺省取消按钮,即使把焦点移到其他控件上,也能通过按 Esc 键选中该按钮。在设计时,通过设置某按钮的 Cancel 属性为 True,就可指定它为窗体的缺省取消按钮。在一个窗体中只能有一个按钮为缺省取消按钮。

按按钮的访问键 (Alt+带下划线的字母)

可通过 Caption 属性创建命令按钮的访问键,为此,只需在作为访问键的字母前添加一个连字符 (&)。例如,要为标题为 Print 的按

钮创建访问键，应在字母 P 前添加连字符（&）,于是得到&Print.运行时，字母 P 将带下划线，同时按 Alt+P 键就可选定命令按钮。

注意：如果不创建访问键，而又要使标题中包含连字符但不创建访问键，应添加两个连字符（&&）.这样一来，在标题中就只显示一个连字符而不显示下划线。

使用 Enter 键来执行某项操作和使用 Esc 键来取消某项操作是人们的操作习惯，因此，在设计程序时，最好设置窗体的缺省命令按钮和缺省取消按钮，来迎合人们的操作习惯。

实例 5.4 使用键盘操作按钮

在该程序中，用户可以使用 Enter 键，Esc 键和访问键来选中按钮。若按 Esc 键，则退出程序。

在窗体中放置一个标签控件、一个文本框控件和两个按钮控件，其中各对象的属性设置如表 5.6 所示。

表 5.6 实例 5.4 中各对象的属性设置		
对象	属性	值
窗体	Caption	使用键盘操作按钮
标签	Caption	当前时间
文本框	Font	宋体、四号
	Text	TexTime
按钮 1	名称	ComOk
	Caption	确定(&Ok)
	Default	True
按钮 2	名称	ComEsc
	Caption	确定(&Esc)
	Cancel	true

双击【确定】按钮，打开【代码】窗口，将下列代码添加到 ComOK_Click 事件过程中：

```
TexTime.Text =Time
```

```
End Sub
```

将退出程序的 end 语句添加到 ComEsces_Click 事件过程中:

```
Private Sub ComEsc_Click()
```

```
End
```

```
End Sub
```

运行该程序,单击【确定】按钮或按下 Alt+O 组合键都将在文本框中显示当前的时间。将焦点移动到文本框上,按下 Enter 键也可以在文本框中显示当前的时间。单击【取消】按钮、按下 Esc 键或按下 Alt+E 组合键将退出程序。

VB6.0 基础教程 5.2.4 图片按钮

在 Windows 程序界面中,标准的按钮形式是立体的长方形,在其上显示文本提示信息,表明按钮的功能。为了使用户界面更加生动,一些按钮上不是用文字,而是用图片来表明按钮的功能,如按钮的功能是保存,则在按钮上显示一个磁盘图片。

在 VB 中,如果将按钮控件的 Style 属性的值设置为 1,就可以通过 Picture 属性来设置要在按钮上显示的图片,通过 DownPicture 属性设置按钮被按下时显示的图片,通过 DisablePicture 属性设置按钮无效时显示的图片。

实例 5.5 图片按钮。

在该程序中,按钮上显示有图片,形象地说明了该按钮的功能。并且,按钮上的图片还会根据用户的操作,做出相应的变化。

在窗体上放置一个标签控件和两个按钮控件，如图 5.13 所示，其中各对象的属性设置如表 5.7 所示。



图 5.13 实例 5.5 的窗体设计

表 5.7 实例 5.5 中各对象的属性设置

对象	属性	值
窗体	Caption	图片按钮
标签	名称	Label1
	AutoSize	True
对象	属性	值
按钮 1	BorderStyle	1
	Caption	置空
	名称	ComLight
	Caption	关灯
	Picture	d:\Microsoft visual Studio\Common\Graph-ics\Icons \Misc\Lightoff.ico
按钮 2	Style	1
	名称	ComFace
	Caption	置空
	Picture	D:\ Microsoft visual Studio\Common\Gra-phics\Icons \Misc\Face02.ico
	Style	1

提示：在安装 VB 时，如果选择了安装图形选项，则在 VB 安装目录下的\Common\Graphics 目录将有大量的位图文件、元文件以及图标文件等。在需安各种图形或图标时，可以在那里找到。

双击第一个按钮控件，打开【代码】窗口，将下列代码添加到 ComLight_Click 事件过程中：

```
Private Sub ComLight_Click()
```

```

If ComLight.Caption= “关 灯” Then

    ComLight.Picture=LoadPicture( “d:\Microsoft visual
studi o\Common\Graphics\Tcons\Misc\Lighton.ico “

    ComFace.picture=Loadpicture( “d:\Microsoft visual
studi o\Common\Graphics\Tcons\Misc\Face04.ico “

    ComLight.Caption= “开 灯”

Else

    ComLiht.Picture=LoadPicture(( “d:\Microsoft visual
studi o\Common\Graphics\Tcons\Misc\Lightoff.ico “)

    ComFace.picture=Loadpicture( “d:\Microsoft visual
studi o\Common\Graphics\Tcons\Misc\Face02.ico “)

    ComLight.Caption= “关 灯”

End If

End Sub

```

【关灯】按钮是一个多功能按钮，与实例 5.3 一样，使用了 If 语句来判断按钮当前的标题。在程序运行时，设置对象的 Picture 属性的格式如下：

对象名 Picture = LoadPicture (“文件名”)

注意：在程序运行时，不能直接将文件名赋予控件的 Picture 属性，而要使用 LoadPicture（）函数。

将下列代码添加到 ComFace_Click 事件过程中：

```
Private Sub ComFace_Click()
```

```
If ComLight.Caption= “关 灯” Then  
    Labell.Caption= “我 高 兴！ ”  
Else  
    Labell.Caption= “我 生 气！ ”  
End If  
End Sub
```

运行该程序，单击【关灯】按钮，则该按钮上的图片变成一个发亮的灯泡，提示文本也由"关灯"变成了"开灯",且另一个按钮上的笑脸图片变成了哭脸图片。单击【哭脸】按钮，则在标签中显示"我生气!",如图 5.14 所示。再次单击【开灯】按钮，则该按钮'上的图片恢复为一个关灭的灯泡，提示文本"开灯"变成了"关灯",且另一个按钮上的哭脸图片恢复为笑脸图片。单击【笑脸】按钮，则在标签中显示"我高兴!",如图 5.15 所示。

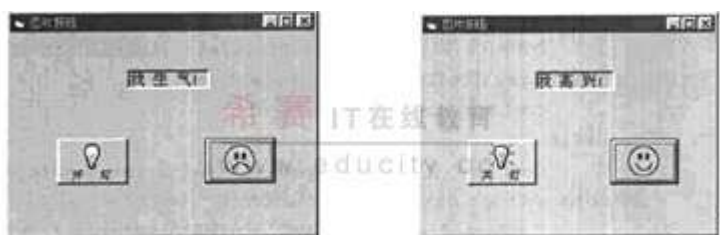


图 5.14 单击【关灯】与【哭脸】按钮的情形 图 5.15 单击【开灯】与【笑脸】按钮的情形

VB6.0 基础教程 5.3.1 文本框的基本属性

在前面的一些实例中，已经接触过文本框的 **Text** 属性。该属性是文本框最重要的一个属性，在设计时，使用该属性可以指定文本框的初始值。在程序中。**Text** 属性用来返回用户在文本框中输入的内容。

如要将用户在文本框（名称为 Text1）中输入的内容显示在窗体上，可以使用以下语句：

Print Text1.Text

表 5.8 列出了文本框的其他一些属性。

表 5.8 文本框的基本属性	
属性	说明
BorderStyle	决定文本框是否有边框
BackColor	设置文本框的背景颜色
Font	设置文本框中的字体以及字号
ForeColor	设置文本的颜色
Locked	决定是否将文本框锁定。文本框被锁定后，就不能对其中的文本执行编辑
MaxLength	设置文本框最多能接收的字符数。默认值是 0，表示对字符数没有限制。 文本框中可以放置的最大字符容量约为 64KB，单行文本框则只能放置 255 个字符
MultiLine	默认情况下，该值为 False，表明文本框只能接收单行文本。将该值设置为 True，则在文本框可以输入多行文本
Password Char	将用户输入的字符在文本框中显示为指定的字符，例如，将该值设置为星号 (*)，则用户输入的任何字符在文本框中均显示为 (*)。文本框用来输入密码时，经常使用到该属性
ScrollBars	决定文本框是否可以带有滚动条。该属性有 4 种取值，各取值的含义如表 5.9 所示

表 5.9 ScrollBars 属性的取值及其含义	
属性值	含义
0	文本框没有滚动条（默认值）
1	文本框有水平滚动条
2	文本框有垂直滚动条
3	文本框既有水平滚动条又有垂直滚动条

如果文本框的 MultiLine 属性的值设置为 False,则这样的文本框称为单行文本框。在单行文本框中输入的内容均处于一行，即便是按 Enter 键也不能实现换行。超出文本框的内容不显示出来，可以将插入点置于文本框中，然后按方向键来查看未显示出的内容。

如果文本框的 MultiLine 属性的值设置为 True,则这样的文本框称为多行文本框。对于多行文本框，如果 ScrollBars 属性使用默认值（值为 0）,则在文本框中输入内容时，当一行填满之后，就会自动转换到下一行，也可以按 Enter 键强制换行。将插入点置于文本框中，然后

按方向键即可查看超出文本框中的内容。如果 `ScrollBars` 属性的值不为 0,则文本框中会出现滚动条,通过滚动条可以方便地查看超出文本框的内容。

表 5.8 文本框的基本属性

属性	说明
<code>BorderStyle</code>	决定文本框是否有边框
<code>BackColor</code>	设置文本框的背景颜色
<code>Font</code>	设置文本框中的字体以及字号
<code>ForeColor</code>	设置文本的颜色
<code>Locked</code>	决定是否将文本框锁定。文本框被锁定后,就不能对其中的文本执行编辑
<code>MaxLength</code>	设置文本框最多能接收的字符数。默认值是 0,表示对字符数没有限制。文本框中可以放置的最大字符容量约为 64KB,单行文本框则只能放置 255 个字符
<code>MultiLine</code>	默认情况下,该值为 <code>False</code> ,表明文本框只能接收单行文本。将该值设置为 <code>True</code> ,则在文本框可以输入多行文本
<code>Password Char</code>	将用户输入的字符在文本框中显示为指定的字符,例如,将该值设置为星号 (*),则用户输入的如何字符在文本框中均显示为 (*),文本框用来输入密码时,经常使用到该属性
<code>ScrollBars</code>	决定文本框是否可以带有滚动条。该属性有 4 种取值,各取值的含义如表 5.9 所示

表 5.9 ScrollBars 属性的取值及其含义

属性值	含义
0	文本框没有滚动条(默认值)
1	文本框有水平滚动条
2	文本框有垂直滚动条
3	文本框既有水平滚动条又有垂直滚动条

如果文本框的 `MultiLine` 属性的值设置为 `False`,则这样的文本框称为单行文本框。在单行文本框中输入的内容均处于一行,即便是按 `Enter` 键也不能实现换行。超出文本框的内容不显示出来,可以将插入点置于文本框中,然后按方向键来查看未显示出的内容。

如果文本框的 `MultiLine` 属性的值设置为 `True`,则这样的文本框称为多行文本框。对于多行文本框,如果 `ScrollBars` 属性使用默认值(值为 0),则在文本框中输入内容时,当一行填满之后,就会自动转换到下一行,也可以按 `Enter` 键强制换行。将插入点置于文本框中,然后按方向键即可查看超出文本框中的内容。如果 `ScrollBars` 属性的值

不为 0,则文本框中会出现滚动条,通过滚动条可以方便地查看超出文本框的内容。

图 5.16 所示的是在几个 MultiLine 属性和 ScrollBars 属性设置不同的文本框中输入相同的内容后的情形,其中各文本框的 MultiLine 属性和 ScrollBars 属性的设置如表 5.10 所示。



图 5.16 文本框的 MultiLine 属性和 ScrollBars 属性

表 5.10 图 5.16 中各文本框的 MultiLine 属性和 ScrollBars 属性的设置

文本框	MultiLine 属性	ScrollBars 属性
文本框 1	False	无所谓
文本框 2	True	0
文本框 3	True	1
文本框 4	True	3

VB6.0 基础教程 5.3.2 字体与字号

大多数控件都有 Font 属性,用来设置显示在控件上文本的字体与字号。通过【属性】窗口设置 Font 属性的方法是:单击 Font 属性,则在属性行的右端会出现一个显示有"..."符号的按钮,单击该按钮则打开【字体】对话框。在该对话框中选择一种需要的字体(如隶书)、样式(如规则)、字号(如小四)和效果(如下划线),单击【确定】按钮即可。

与其他属性不同,在代码中不能使用 Font 属性,如下列语句是错误的。

Text1.Font="宋体".

Text1.Font="宋体， 四号".

事实上，在【属性】窗口中通过设置 Font 属性同时也设置多项属性，如字体、字号和效果等。而在代码中，每一个属性都对应一个属性名，如字体的属性名为 FontName.表 5.11 中列出了在代码中设置字体、字号、黑体和下划线等属性的属性名以及示例。

表 5.11 在代码中设置字体等属性

属性名	说明	示例
FontName	设置字体	Text1.FontName = "隶书"
FontSize	设置字号	Text1.FontSize = 14
FontBold	设置黑体，若值为 True 则文本为黑体	Text1.FontBold = True
FontItalic	设置斜体，若值为 True 则文本为斜体	Text1.FontItalic = True
FontStrikethru	设置删除线，若值为 True 则文本有删除线	Text1.FontStrikethru = True
FontUnderline	设置下划线，若值为 True 则文本有下划线	Text1.FontUnderline = True

VB6.0 基础教程 5.3.3 选择文本

文本框控件还提供了 3 个属性，用于操作用户所选择的文本。且这 3 个属性不能在【属性】窗口中设置，只能在代码中使用。表 5.12 中列出这 3 个属性以及它们的含义。

表 5.12 有关选择文本的三个属性

属性	含义
SelText	该属性返回用户所选的文本
SelStart	设置或返回所选文本的第一个字符的位置，若没有选中任何文本，则为插入点的位置
SelLength	属性设置或返回所选文本的长度，若没有选中任何文本，则值为 0

要在程序中操作用户所选的文本，如将文本替换成指定的文本以及更改所选文本的大小写等，都可以使用 SelText 属性。

例如，将用户在文本框（Text1）中所选文本替换成 3 个 A 的语句如下：

Text1.SelText=""AAA"".

要删除当前所选的文本，只需向 SelText 属性赋予空字符串即可，语句如下：

```
Text1.SelText="".
```

将所选文本转换成大写，可以使用 Ucase（）函数，语句如下：
Text1.SelText=UCase（Text1.SelText）.

实例 5.6 替换文本。

在该程序中，用户在一个文本框中输入一段文本，使用鼠标拖动选中要替换的字符串，则在窗体上显示出所选字符串的起始位置和字符串的长度。在另一个文本框中输入替换内容后，单击【替换】按钮即可将所选的字符串替换，如图 5.19 所示，将用户所选的字符串替换成了"改变"两个字。

在窗体中放置五个标签控件、两个文本框控件和一个按钮控件，如图 5.20 所示。其中各对象的属性设置如表 5.13 所示。



图 5.19 所选字符串被替换



图 5.20 实例 5.6 的窗体设计

表 5.13 实例 5.6 中各控件属性表

控件	属性	值
窗体	Caption	替换文本
标签 1	Caption	位置:
标签 2	名称	LabStart
	Caption	置空
标签 3	Caption	长度:
标签 4	名称	LabLength
	Caption	位置:
标签 5	Caption	替换成:
文本框 1	名称	TexSel
	MultiLine	True
	Text	置空
文本框 2	名称	TexCh
	Text	置空
按钮	名称	ComCh
	Caption	替换

使用鼠标拖动选中文本框中的字符串后，释放鼠标，则窗体上就显示出所选字符串的信息。因此，可以将显示所选字符串信息的代码添加到文本框的 MouseUp 事件中。Texsel_MouseUp 事件过程如下所示：

```
Private Sub Textsel_MouseUp(Button As Integer,Shift As Integer,x As Single,Y As Single )
    LabStart.Vaption=TextSel.SelStart
    LabStart.Caption=TextSel.selLength
End Sub
```

在按钮的 Click 事件中添加如下代码：

```
Private Sub Comch_Click ( )
    Text.SelText Texch.Text
End Sub
```

这样，一个替换文本程序就创建完毕。

VB6.0 基础教程 5.3.4 密码框

密码框是一种特殊的文本框，它的特殊之处在于：当用户向密码框中输入文本时，不论用户输入的是什么字符，在密码框中总是显示特定的字符。如*、#等。这样，别人在密码框中就看不到用户所输入的实际内容，从而达到了保密的效果。

通过设置文本框的 **Password Char** 属性就可以将普通的文本框设置成为密码框。在缺省情况下，**Password Char** 属性的值为空字符串。这时用户在键盘上输入什么字符，在文本框中就显示什么字符。如果将 **Password Char** 属性的值设置为某个字符，如设置为星号（*），则用户在文本框中输入任何字符都将显示为*。例如，输入的是 Hcq，显示的则是***。

但是，文本框的 **Password Char** 属性并不影响 **Text** 属性，尽管在文本框中显示的是在 **Password Char** 属性中指定的字符，但 **Text** 属性返回的仍然是用户输入的实际内容。根据这一点，可以编写一个验证密码的小程序。

实例 5.7 验证密码

在该程序中，要求用户输入密码，如果输入正确，则用户可以继续下一步操作，否则，在窗体上显示"密码输入错误，请再试一次！"，并且用户只有三次输入密码的机会，如果三次输入错误，则文本框变为无效，不能接受用户的任何输入。在本例中，认为正确的密码为abcd。

单击【添加窗体】按钮向当前工程中再添加一个窗体，其中一个窗体用作验证密码。在用作验证密码的窗体上放置两个标签控件、一个文本框控件和一个按钮控件，如图 5.21 所示。各对象属性设置如表 5.14 所示。在另一个窗体上放置一个标签控件和一个按钮控样，如图 5.22 所示，各对象的属性如表 5.15 所示。



图 5.21 验证密码窗体的设计



图 5.22 另一窗体的设计

表 5.14 密码对话框中各对象的属性设置

对象	属性	值
窗体	名称	ForPass
	Caption	验证密码
标签 1	Caption	请输入密码:
标签 2	名称	LabMsg
	Caption	置空
文本框	AutoSize	True
	名称	TextPass
	Text	置空
	Pass	*
按钮	名称	ComOk
	Caption	确定

表 5.15 另一窗体中各对象的属性设置

对象	属性	值
窗体	Caption	应用程序
标签	Caption	欢迎使用 VB
按钮	Font	隶书、粗体、一号
	名称	ComClose
	Caption	关闭

双击验证密码窗体中的【确定】按钮，打开【代码】窗口，将下列代码添加到 ComOk_Click 事件过程中：

```
Private Sub ComOK_Click()

Static i As Integer

If i<=2 Then
```

```

If TexPass.Text= "abcd" Then

Unload ForPass

Formain.Show

Else

LabMsg.Caption= "密码错误，请再试一遍！"

End If

Else

LabMsg.Caption= "三次输入错误，拒绝重新输入！"

TextPass.Enabled=False

End If

i=i+1

End Sub

```

在该段代码中，首先定义了一个静态变量 *i*，它用来记录用户输入密码的次数。*i* 的初值为 0，每单击一次按钮，则 *i* 的值增 1 ($i=i+1$)。然后使用 If 语句来判断 *i* 的值，如果 *i* 的值小于 3，即用户输入密码不超过三次，又使用了一个 If 语句来判断用户所输入的密码是否正确。如果正确（即输入的是 abcd），则验证密码窗体消失，同时启动另一个窗体。如果输入的密码不正确，则会在窗体的标签上显示"密码错误，请再试一遍！"。如果第三次输入密码也不正确，此时 *i* 的值已经累加到 3。再次输入密码，程序不会再判断密码是否正确（因为 $1 \leq 3$ ），而是在窗体上显示"三次输入错误，拒绝重新输入！"，并且将文本框置为无效。因此，即便是用户在第四次输入了正确的密码，也无济于事。

双击另一个窗体上的按钮控件,将程序结束语句 **End** 添加到按钮的 Click 事件中,如下所示:

```
Private Sub ComMain_Click ( )
```

```
End
```

```
End Sub
```

在【工程属性】对话框中设置启动窗体为 **ForPass**,运行该程序,则出现【验证密码】窗体,在文本框中输入字符串 **abcd**,文本框中显示的是"****",如图 5-23 所示。单击【确定】按钮,则验证密码窗体消失,另一个窗体显示出来,如图 5.24 所示。单击【关闭】按钮可以退出该程序。



图 5.23 验证密码窗体

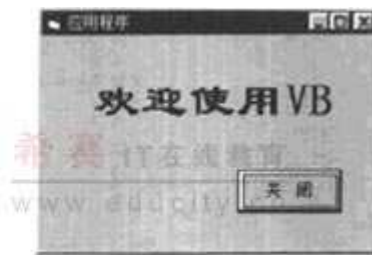


图 5.24 另一个窗体

再次运行该程序,在【验证密码】窗体的文本框中随意输入一个字符串(不是 **abcd**),单击【确定】按钮,则窗体上显示"密码错误,请再试一遍!".如图 5.25 所示。连续三次输入错误的密码,当第四次输入密码时,无论密码正确与否,单击【确定】按钮则窗体上显示"三次输入错误,拒绝重新输入!",并且将文本框置为无效,用户无法继续输入密码,如图 5.26 所示。



图 5.25 输入了错误的密码



图 5.26 三次都输入了错误的密码

VB6.0 基础教程 5.3.5 Change 事件

文本框也有 Click, DblClick 等事件, 但文本框的这些事件并不常用。文本框较常用的一个事件是 **Change** 事件, 一旦文本框中的内容被改变, 就会触发它的 **Change** 事件。

实例 5.8 利用 Change 事件

在该程序中, 用户在文本框中输入内容时, 窗体上就会同步显示出用户所输入的内容。并且如果用户修改了文本框内容, 则窗体上的内容也会同步修改。

要使窗体上显示的内容总是与文本框中的内容同步改变, 需要使用文本框的 **Change** 事件, 因为该事件能随时感知到文本框中内容的改变。

在窗体中放置一个标签控件、一个文本框控件和一个按钮控件, 如图 5.27 所示, 其中各对象的属性设置如表 5.16 所示。

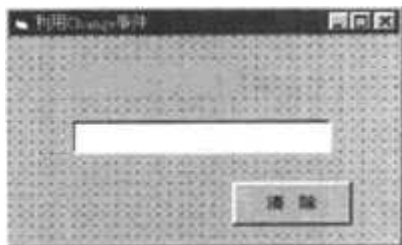


图 5.27 实例 5.8 的窗体设计

表 5.16 实例 5.8 中各对象的属性设置

对象	属性	值
窗体	Caption	利用 Change 事件
标签	名称	LabCh
	AutoSize	True
	Caption	置空
	Font	隶书, 小三
对象	属性	值
文本框	名称	TexCh
	Text	置空
按钮	名称	ComClear
	Caption	清除

双击文本框控件，打开【代码】窗口，在代码编辑区中自动出现了 Change 事件的框架。

编写 Change 事件过程如下：

```
Private Sub Textch_Change ()
LabCh.Caption=TexCh.Text
End Sub
```

再编写按钮的 Click 事件过程如下：

```
Private Sub comClear Click ()
TexCh.Text=""
End Sub
```

运行该程序，在文本框中输入内容，则窗体上就会同步显不出用户所输入的内容，改变文本框中的内容，则窗体上的内容也会随着改变。图 5.29 所示的是在文本框中输入"清华大学计算机系"后的情形。单击【清除】按钮，则文本框中的内容被清除，并且窗体中的内容也被清除，如图 5.30 所示。

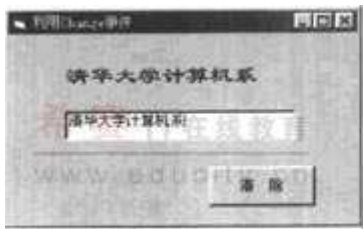


图 5.29 在文本框中输入内容



图 5.30 清除文本框中的内容

尽管在 ComClear_Click 事件过程中只有清除文本框中内容的语句，但由于在文本框中的内容被清除后，触发了它的 Change 事件，因此，单击【清除】按钮后，Change 事件过程也被执行了。

VB6.0 基础教程 5.3.6 使用剪贴板交换文本

大多数 Windows 应用程序都有"复制"和"粘贴"命令，用户使用这些命令，就可以通过剪贴板来交换信息了。在 VB 中，可以使用 Clipboard 对象来操作剪贴板。Clipboard 对象没有任何属性与事件，但使用它的方法可以实现对剪贴板的操作。Clipboard 对象的方法可分为三类：GetText 和 SetText 方法，用来传送文本；GetData 和 SetData 方法，用来传送图形；GetFormat 和 Clear 方法，可以处理文本和图形两种格式。本节只讲述使用剪贴板交换文本。

SetText 方法是将文本复制到剪贴板上，替换先前存储在那里的文本。可将 SetText 作为一条语句使用。其语法如下：

Clipboard.SetText 数据[格式]

GetText 方法是返回存储在剪贴板上的文本。也可将它作为函数使用，其语法如下：

目标=Clipboard.GetText () Clear 方法是清除剪贴板中的内容。需要注意的是：在使用 SetText 方法将文本复制到剪贴板时，都要先

用 Clear 方法将剪贴板清空。因为如果在剪贴板中存放着不同格式的数据，则剪贴板不会自动清空。

实例 5.9 使用剪贴板交换文本。

在该程序中，用户可以通过剪切板来交换两个文本框中的文本。

在窗体中放置两个文本框控件和三个按钮控件，如图 5.31 所示，其中各对象的属性设置如表 5.17 所示。

表 5.17 实例 5.9 中各对象的属性设置		
对象	属性	值
窗体	Caption	使用剪贴板交换文本
文本框 1	名称	TexS
	MultiLine	True
	Text	置空
文本框 2	名称	TexD
	MultiLine	True
	Text	置空
按钮 1	名称	ComCopy
	Caption	复制
按钮 2	名称	ComCut
	Caption	剪切
按钮 3	名称	ComPaste
	Caption	粘贴

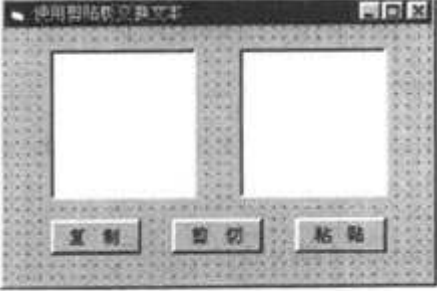


图 5.31 实例 5.9 的窗体设计

双击【复制】按钮，打开【代码】窗口，将以下代码添加到 ComCopy_Click 事件过程中：

```
Private sub ComCopy_Click()  
  
If TexS.SelLength>0 Then  
  
Clipboard.Clear
```

```
Clipboard.SetText TexS.SelText
```

```
End If
```

```
End Sub
```

在该段代码中，使用了一个 If 语句来判断用户是否在文本框 1 中选中了文本，如果没有选中，则不执行任何操作，如果选中了文本。则首先将剪贴板中内容清除，然后将用户所选的文本传送到剪贴板中。

【剪切】与【复制】的区别是，【剪切】不仅将用户所选的文本传送到剪贴板中，并且将所选文本删除。因此，只需在【复制】按钮的 Click 事件过程中添加一行删除所选文本的代码，即可得到【剪切】按钮的 Click 事件过程，ComCopy_Click 事件过程如下：

```
Private Sub ComCut_Click ()
```

```
If TexS.SelLength>0 Then
```

```
Clipboard.Clear
```

```
Clipboard.SetText TexS.SelText
```

```
TexS.SelText=""
```

```
End If
```

```
End Sub
```

【粘贴】按钮的 Click 事件过程如下：

```
Private Sub ComPaste_Click ()
```

```
TexD_SelText=Clipboard.GetText ()
```

```
End Sub
```

GetText 方法将返回剪贴板上当前的文本字符串，然后用一条赋值语句将该字符串复制到文本框 2 的指定位置 (TexD.SelText)。如果当前没有被选定的文本，则将该文本粘贴在文本框中插入点处。

运行该程序，在文本框 1 中输入一段文本，然后使用鼠标在文本框中拖动选中一段文本，单击【复制】按钮，再单击【粘贴】按钮。则用户所选文本就粘贴到文本框 2 中了，如图 5.32 所示。再在文本框 1 中选中一段文本，单击【剪切】按钮，则所选文本被删除。将插入点置于文本框 2 中的某位置，单击【粘贴】按钮，则所选文本就粘贴到插入点处，如图 5.33 所示。

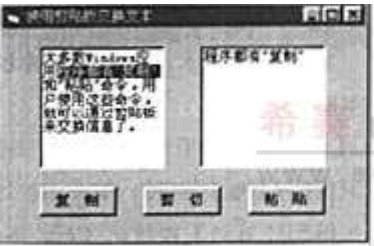


图 5.32 复制与粘贴文本

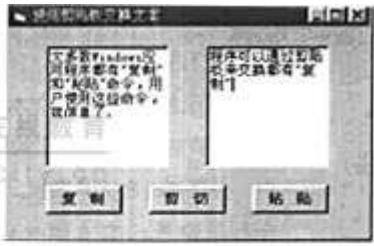


图 5.33 剪切与粘贴文本

VB6.0 基础教程 5.4 单选项控件

单选项用来接收用户作出的选择，它通常以单选项组的形式出现，用户每次只能在一组单选项中选中其中的一个。

表 5.18 中列出了单选项的几个基本属性。

表 5.18 单选项的几个基本属性

属性	说明
Caption	设置单选项按钮的标题。使用该属性还可以为单选项创建键盘访问键。只要在作为访问键的字母前添加一个连字符 (&) 即可。例如, 要为单选项标题 Pentium 创建访问键, 应在字母 P 前添加连字符: &Pentium。运行时, 字母 P 将带下划线, 同时按 Alt+P 组合键就可选定此单选项
Enabled	确定单选项是否有效。当值为 False 时, 则运行时将显示暗淡的选项按钮, 这意味着按钮无效
Style	确定单选项的外观。值为 0 时, 为标准的单选项按钮, 即一个圆形按钮及标题。值为 1 时, 外观类似于命令按钮。单击选中该项, 则按钮处于下沉状态。单击选中其他选项后, 按钮恢复原状
Picture	当 Style 属性的值为 1 时, Picture 属性用来设置选项按钮上显示的图片
DownPicture	当 Style 属性的值为 1 时, DownPicture 属性用来设置选项按钮被按下时 (选中状态) 显示的图片
Value	设置或返回单选项的状态。值为 True 时, 表明该单选项被选中; 值为 False 时, 表明该单选项未被选中。在同一组中的单选项, 只能有一个的 Value 为 False。当设置某个单选项 Value 属性的值为 True 时, 则其他单选项的 Value 值同时被设置为 False

注意: 要使标题包含连字符但不创建访问键, 就应使标题包含连字符 (&&)。这样, 标题中将显示一个连字符, 而且没有字符带下划线。

实例 5.10 单选项的属性

该实例用来演示不同属性设置下的单选项的外观。在窗体中放置 4 个单选项控件, 各单选项控件的属性设置如表 5.19 所示, 没有列出的属性使用的是系统默认值。

表 5.19 实例 5.10 中各对象的属性设置

对象	属性	值
单选项 1	Caption	&File
单选项 2	Caption	Edit
	Value	True
单选项 3	Caption	
	Style	1
	Picture	某位图文件
	DownPicture	某位图文件
单选项 4	Caption	View
	Style	1
	Picture	某位图文件
	DownPicture	某位图文件

运行该程序, 可见第一个单选项的标题 File 的字母 F 下有一个下划线, 表明 F 是该单选项的访问键, 同时按 Alt+F 组合键就可选定该

单选项。单击第 4 个单选项，则该选项按钮呈下沉状态，并且在其上显示用户在 DownPicture 属性中设置的图片，如图 5.35 图所示。

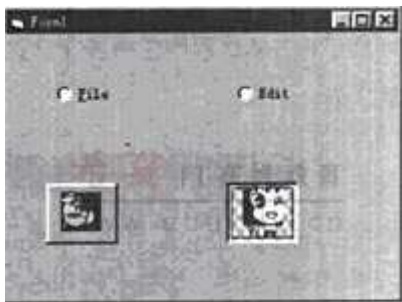


图 5.35 选中单选项按钮的情形

VB6.0 基础教程 5.4 单选项控件

单选项用来接收用户作出的选择，它通常以单选项组的形式出现，用户每次只能在一组单选项中选中其中的一个。

表 5.18 中列出了单选项的几个基本属性。

表 5.18 单选项的几个基本属性

属性	说明
Caption	设置单选项按钮的标题。使用该属性还可以为单选项创建键盘访问键。只要在作为访问键的字母前添加一个连字符 (&)即可。例如，要为单选项标题 Pentium 创建访问键，应在字母 P 前添加连字符：&Pentium。运行时，字母 P 将带下划线，同时按 Alt+P 组合键就可选定此单选项
Enabled	确定单选项是否有效。当值为 False 时，则运行时将显示暗淡的选项按钮，这意味着按钮无效
Style	确定单选项的外观。值为 0 时，为标准的单选项按钮，即一个圆形按钮及标题。值为 1 时，外观类似于命令按钮。单击选中该选项，则按钮处于下沉状态。单击选中其他选项后，按钮恢复原状
Picture	当 Style 属性的值为 1 时，Picture 属性用来设置选项按钮上显示的图片
DownPicture	当 Style 属性的值为 1 时，DownPicture 属性用来设置选项按钮被按下时（选中状态）显示的图片
Value	设置或返回单选项的状态。值为 True 时，表明该单选项被选中；值为 False 时，表明该单选项未被选中。在同一组中的单选项，只能有一个的 Value 为 False。当设置某个单选项 Value 属性的值为 True 时，则其他单选项的 Value 值同时被设置为 False

注意：要使标题包含连字符但不创建访问键，就应使标题包含连字符 (&&)。这样，标题中将显示一个连字符，而且没有字符带下划线。

实例 5.10 单选项的属性

该实例用来演示不同属性设置下的单选项的外观。在窗体中放置 4 个单选项控件，各单选项控件的属性设置如表 5.19 所示，没有列出的属性使用的是系统默认值。

表 5.19 实例 5.10 中各对象的属性设置

对象	属性	值
单选项 1	Caption	&File
单选项 2	Caption	Edit
	Value	True
单选项 3	Caption	
	Style	1
	Picture	某位图文件
	DownPicture	某位图文件
单选项 4	Caption	View
	Style	1
	Picture	某位图文件
	DownPicture	某位图文件

运行该程序，可见第一个单选项的标题 File 的字母 F 下有一个下划线，表明 F 是该单选项的访问键，同时按 Alt+F 组合键就可选定该单选项。单击第 4 个单选项，则该选项按钮呈下沉状态，并且在其上显示用户在 DownPicture 属性中设置的图片，如图 5.35 图所示。

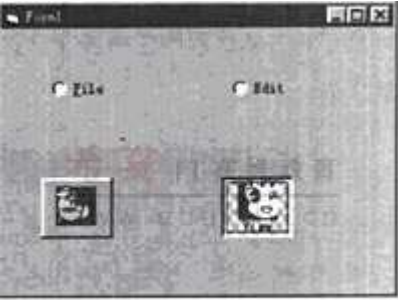


图 5.35 选中单选项按钮的情形

VB6.0 基础教程 5.4.2 在程序中使用单选项

选定选项按钮时将触发其 Click 事件。是否有必要响应此事件，取决于应用程序的功能。例如，当希望通过更新文本框的内容向用户提供有关选定项目的信息时，对此事件作出响应是很有益的。

实例 5.11 单选项的 Click 事件

在该实例中，当用户选定某单选项时，则在文本框中显示用户所选项目的有关信息。该实例主要使用到单选项控件对 Click 事件的响应。

在窗体中放置三个单选项控件和一个文本框控件，如图 5.36 所示。其中各控件的属性设置如表 5.20 所示。

表 5.20 实例 5.11 中各对象的属性设置

对象	属性	值
单选项 1	名称	Opc
	Caption	中国
单选项 2	名称	Opa
	Caption	美国
单选项 3	名称	Opj
	Caption	日本
文本框	名称	Text1
	Text	置空
	Multiline	True



图 5.36 实例 5.11 的窗体设计

双击【中国】单选项，打开【代码】窗口，将下列代码添加到 Opc_Click 事件过程中：

```
Private Sub Opc_Click ()  
Text1.Text="中国是个人口大国"  
End Sub
```

双击【美国】单选项，将下列代码添加到 Opa_Click 事件过程中：

```
Private Sub Opa_Click ()
```

```
Text1.Text="美国是个军事强国"
```

```
End Sub
```

双击【日本】单选项，将下列代码添加到 Oj_Click 事件过程中：

```
Private Sub Opj_Click ( )
```

```
Text1.Text="日本是个经济强国"
```

```
End Sub
```

单击工具栏中的工具按钮运行该程序，选中某单选项，则在文本框中将显示出该单选项的有关信息，如图 5.37 所示的是选中【美国】单选项的情形。

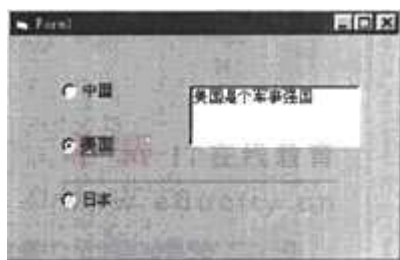


图 5.37 选中【美国】单选项

VB6.0 基础教程 5.5 框架控件

在上面的一些操作中，可以发现所有直接添加到窗体中的单选项总是属于同一个组，用户只能选定其中的一个。在一些应用程序中常常需要有多组选项，用户可在每组选项中作出一个选择。这时，就需要使用到框架控件，用户可首先在窗体中放置几个框架控件，然后再将单选项控件放置在框架中，则处于同一框架中的单选项属于同一组。

实例 5.12 单选项的分组。

在该实例中，要求用户选择所毕业的学校以及学历，如果用户选择的毕业学校是清华大学，选择的学历是博士，单击按钮后，文本框中将显示"您符合我公司的用人要求",否则，显示"您不符合我公司的用人要求".

该实例主要使用到单选项的 **Value** 属性，**Value** 属性可用来设置单选项组的初始状态，也可以在代码中返回用户所做的选择。

在窗体中并排放置两个框架控件，在两个控件中分别放置三个单选项控件，在框架外的窗体上再放置一个按钮控件和一个文本框控件，如图 5.38 所示。其中各控件的属性设置如表 5.21 中所示。



图 5.38 实例 5.12 的用户界面

表 5.21 实例 5.12 中各控件的属性设置

控件	属性	值
窗体	Caption	单选项的分组
框架 1	Caption	毕业学校
框架 2	Caption	学历
单选按钮 1	名称	Opqh
	Caption	清华大学
	Value	True
单选按钮 2	名称	Opbj
	Caption	北京大学
单选按钮 3	名称	Opfd
	Caption	复旦大学
单选按钮 4	名称	Opbk
	Caption	本科
	Value	True
单选按钮 5	名称	Opas
	Caption	硕士
单选按钮 6	名称	Opbs
	Caption	博士
按钮	名称	Command1
	Caption	提交
文本框	名称	Text1
	Caption	
	MultiLine	True
	Locked	True

双击【提交】按钮，打开【代码】窗口，将下列代码添加到 Command1_Click 事件过程中：

```
Private Sub Command1_Click()

    If Opqh.Value=True And Opbs.Value=True Then

        Text1.Text="您符合我公司的用人要求"

    Else

        Text1.Text="您不符合我公司的用人要求"

    End If

End Sub
```

运行该程序，用户可以在两个框架中分别选中一个单选项，单击【提交】按钮，则文本框中就会显示出相关信息。图 5.39 所示的是选中【清华大学】与【博士】单选项的情形。



图 5.39 实例 5.12 的运行效果

VB6.0 基础教程 5.6 复选框控件

复选框控件和单选项控件看起来功能相似，都是用来接收用户作出的选择。但它们却有一个重要区别：用户每次只能在单选项组中选中一个单选项，与此对照的是，用户可选定任意数目的复选框。

复选框也有两种状态：选中与不选。当复选框被选中时，复选框中显示一个"√"标记，当复选框不选时，复选框中的"√"标记消失。每单击一次复选框，它的状态在"选中"与"不选"之间切换一次，"√"标记也随之在有无之间切换。

复选框的外观属性与单选项的相应属性类似，用户可参照单选项的属性设置来设置复选框。这里给出一个使用复选框的实例。

实例 5.13 复选框的使用。

在该实例中，用户在文本框中输入一段文字后，可以根据需要改变文本的字体、字型、字号或颜色。

在窗体中放置一个文本框控件和四个复选框控件，如图所示，其中各对象的属性设置如表 5.22 中所示。



图 5.40 实例 5.13 的窗体设计

表 5.22 实例 5.13 中各对象的属性设置

对象	属性	值
窗体	Caption	复选框的使用
	Caption	请在文本框中输入文字
标签	Font	宋体, 五号
文本框	名称	Text1
	Caption	置空
	Font	宋体, 12 磅
	MultiLine	True
框架	Caption	效果
复选框 1	名称	ChFont
	Caption	隶书
复选框 2	名称	ChLine
	Caption	下划线
复选框 3	名称	ChSize
	Caption	18 磅
复选框 4	名称	ChItalic
	Caption	斜体

双击第一个复选框控件打开【代码】窗口，将下列代码添加到

ChFont_Click 事件过程中：

```
Private Sub chFont_Click()

    If chFont.Value= 1 Then

        Text1.FontName="隶书"

    Else

        Text1.FontName="宋体"

    End If

End Sub
```

单选项控件的 **Value** 属性有两个值：**True** 和 **False**,分别对应单选项被选中与未选中。复选框的 **Value** 属性也是用来设置与返回复选框的当前状态，不过它有三个值，值为 0 是表示复选框未选中，值为 1 表示选中，值为 2 时表示"不可用",若复选框呈浅灰色，且始终为选中状态，则在程序运行时用户不能对其进行操作。

在上述代码中，使用一个 **If** 语句来判断复选框的 **Value** 值是否为 1,如果是（即复选框被选中）,则使文本框的 **FontName** 属性值为"隶书",如果不是（即复选框未被选中）,则使文本框的 **PontName** 属性值为"宋体".由于代码处于复选框的 **Click** 事件中，因此，每次单击复选框，程序都将判断复选框当前的状态，文本框中文本的字体也将随着在隶书与宋体之间切换。

其他三个复选框的 **Click** 事件过程与此类似，只是功能不同罢了。下面是它们的事件过程。

将下列代码添加到 **Chlinees_Click** 事样过程中：

```
Private Sub chine_Click()  
    If Chline.Value=1 Then  
        Text1.FontUnderline=Ture  
    Else  
        Text1.FontUnderline=False  
    End If  
End Sub
```

将下列代码添加到 **ChSize_Click** 事件过程中：

```
Private Sub ChSize_ Click()  
  
If ChSiaeValue=1 Then  
  
Text1.FontSize=18  
  
E1Se  
  
Text1.FontSize=10  
  
End If  
  
End Sub
```

将下列代码添加到 ChItalic_Click 事件过程中：

```
Private Sub ChItalic_Click()  
  
If ChItalic.Value=1 Then  
  
Text1.FontItalic=True  
  
Else  
  
Text1.FontItalic=False  
  
End It  
  
End Sub  
  
Private Sub ChItalic_Click()  
  
If ChItalic.Value=1 Then  
  
Text1.FontItalic=True  
  
Else  
  
Text1.FontItalic=False  
  
End It  
  
End Sub
```

运行该程序，在文本框中输入一段文字，选中各复选框，可以发现，每单击一个复选框，文本的外观就随之改变。如选中【下划线】复选框，则文本框中的文本就出现了下划线，再次单击，则下划线消失。图 5.41 所示的是没有选中任何复选框情况下文本的外观，图 5.42 所示的选中所有复选框后文本的外观。



图 5.41 没有选中任何复选框



图 5.42 选中所有复选框

VB6.0 基础教程 5.7.1 图片框控件

设计包含有图片的窗体的方法是：首先，在窗体上要显示图片的位置放置一个图片框控件。然后，将所要的图片加载到图片框控件中即可。

可以加载到图片框控件中的图形格式有以下几种：

位图（bitmap），文件后缀为。bmp.

图标（icon），文件后缀为，ico.

Windows 图元文件，文件后缀为。wmf.

JPEG 或 GIF 文件，文件后缀分别为。jpg 和。gif.

既可以在程序设计阶段向图片框加载图片，也可以在程序运行阶段加载图片。为图片框加载图片的方法有：

在程序设计时，通过在【属性】窗口中设置 Picture 属性来加载图片，加载方法与为窗体加载背景图片的方法相同。

在程序设计时，利用剪贴板加载图片。例如，有时可能希望添加由 Windows 画图板创建的位图图像，可以直接把图像复制到剪贴板上，选定图片框或图像框控件，然后使用键盘快捷方式 Ctrl+V 或使用【编辑】菜单的【粘贴】命令即可。

在程序运行时，使用 LoadPicture（）函数加载图片。加载语句如下：

图片框名，Picture= LoadPicture（文件名）

注意：在程序运行时，不能直接将文件名赋予控件的 Picture 属性，语句如下：

要清除图形框控件中的图形，应使用不指定文件名的 LoadPicture 函数，详句如下：

Picture1.Picture=LoadPicture.

实例 5.14 加载图片。

在该实例中，程序运行后，用户可以更改图片框中的图片，也可以清除图片框中的图片，在更改或清除图片后，用户还可以恢复图片框中原有的图片。

在窗体中放置一个图片框控件和三个按钮控件，如图 5.43 所示。其中各对象的属性设置如表 5.23 所示。



图 5.43 实例 5.14 的用户界

表 5.23 实例 5.14 中对象的属性设置

对象	属性	值
窗体	Caption	加载图片
图片框	名称	Picture1
	Picture	d:\programs\office97\Clipart\Popular\Agree.wmf
按钮 1	名称	ComChange
按钮 2	Caption	更换图片
	名称	ComClear
按钮 3	Caption	清除图片
	名称	ComUndo
	Caption	恢复图片

双击【更换图片】按钮，打开【代码】窗口，将下列代码添加到 ComGhangeClick 事件过程中：

双击【清除图片】按钮，将下列代码添加到 ComClear_Chick 事件过程中：

```
Private sub comchangle_click()

Picture1.picture=LoadPirture

End sub
```

双击[恢复图片]按钮，将下列代码添加到 ComUndo_Click 事件过程中：

```
Private Sub ComUndo_Click()

Picture.oricture=LoadPricture("d:
```

```
programs\office97\Clipart\Popular\Agree.wmf")  
  
End Sub
```

```
Private sub comchangle_click()  
  
Picture1.picture=LoadPirture("d:  
programs\office97\Clipart\Popular\Agree2.wmf")  
  
End sub
```

运行该程序，单击【更换图片】按钮，则图片框中的图片就被更换为另一副图片，如图 5.44 所示。单击【清除图片】按钮，则图片框中的图片被清除，如图 5.45 所示。单击【恢复图片】按钮，则图片框中又出现初始的图片。



图 5.44 更换图片



图 5.45 清除图片

在将工程保存后，VB 为每个包含有图片的窗体生成一个扩展名为 FRS 的文件，这是应用程序存放图片的地方。

在默认情况下，图片框控件的大小不随其中加载图片的大小而变化，并且图片框控件不提供滚动条，因此，如果加载的图片比图片框控件大，则超过的部分显示不出来（除 .wmg 格式的文件外，该格式的文件会自动调整大小以填满图片框）。要使图片框控件自动调整

大小以显示完整图形，应将其 **AutoSize** 属性设置为 **True**。这样，图片框控件将自动调整大小以适应加载的图形。

图片框控件可以作为其他控件的容器，在分组单选项时，可以使用图片框控件替代框架控件。

与窗体一样，图片框控件也可以使用 **Print** 方法来输出文本，使用 **Cls** 方法清除文本。例如，下列语句将在图片框 **Pictures** 中显示"在图片框中显示文本"几个字：

```
Picture1.Print "在图片框中显示文本"
```

VB6.0 基础教程 5.7.2 图像框控件

图像框控件和图片框控件相似，都可用来显示应用程序中的图形，都支持相同的图形格式，且图形的加载方法也相同。它们的不同之处在于：

图片框控件可以作为其他控件的容器，可以使用 **Print** 方法在其中显示文本，而图像框不具有这些功能。

将图片加载到图片框中，图片框可以自动调整其大小以适应加载的图形。将图片加载到图像框中，图片则可以自动调整其大小以适应图像框的大小。

图像框的 **Stretch** 属性决定图片是否能自动调整其大小以适应图像框的大小，当 **Stretch** 属性的值为 **Ture** 时，则图片将自动调整其大小；当 **Stretch** 属性的值为 **False** 时，则图像框自动调整其大小以适应图片的大小。在图 5.46 中，左边图像框 **Stretch** 属性的值为 **False**，右边图像框 **Stretch** 属性的值为 **Ture**。



图 5.46 图像框的 Stretch 属性

实例 5.15 压缩与拉伸图片。

本实例利用图像框的特点 Stretch 属性来实现对图片的压缩与拉伸。在窗体中放置一个图像框控件和两个按钮控件，如图 5.47 所示，其中各对象的属性设置如表 5.24 所示。



图 5.47 实例 5.15 的用户界面

表 5.24 实例 5.15 中各对象的属性设置

对象	属性	值
窗体	Caption	压缩与拉伸图片
图像框	名称	Image1
	Picture	某位图图片
	Height	1900
按钮 1	名称	ComPress
	Caption	压缩
按钮 2	名称	ComStretch
	Caption	拉伸

双击【压缩】按钮，打开【代码】窗口，将下列代码添加到 ComPress_Click 事件过程中：

```
Private Sub ComPress_Click()

If Image1.Height <100 Then
```

```
ComPress.Enabled=False  
  
Else  
  
Imagel.Height=Imagel.Height - 100  
  
ComStretch.Enabled=Ture  
  
End If  
  
End Sub
```

同理，双击【拉伸】按钮，将下列代码添加到 ComPress_Click 事件过程中：

```
Private Sub ComPress_Click()  
  
If Imagel.Height >1900 Then  
  
ComPress.Enabled=False  
  
Else  
  
Imagel.Height=Imagel.Height + 100  
  
ComStretch.Enabled=Ture  
  
End If  
  
End Sub
```

在该程序中，为了避免由于将图片的高度无限拉伸或无限压缩而出错，使用了一个 If 语句来判断当前图片的高度，如果图片的高度小于 100,则将【压缩】按钮置为无效，否则，将当前高度减少 100;如果图片的高度大于 1900,则将【拉伸】按钮置为无效，否则，将当前高度增加 100。

运行该程序，每单击一次【压缩】按钮，图片的高度就缩小一点，当缩小到一定程度时，【压缩】按钮变为无效，不能再压缩图片。此时，单击【拉伸】按钮，则图片被拉伸一点，同时，【压缩】按钮变为有效。不断单击【拉伸】按钮，图片不断被拉伸，当拉伸到一定程度，【拉伸】按钮变为无效，不能再拉伸图片，此时，单击【压缩】按钮，则图片被压缩一点，同时，【拉伸】按钮变为有效。图 5.48 所示的是程序运行后，单击数次【压缩】按钮后图片被压缩的情形。



图 5.48 单击数次【压缩】按钮的效果

VB6.0 基础教程 5.8.1 计时器控件的特点

在窗体上放置计时器控件的方法与绘制其他控件的方法相同：单击工具箱中的计时器按钮并将它拖动到窗体上。计时器控件只在设计时出现在窗体上，可以选定这个控件，查看它的属性以及编写它事件过程。运行时，计时器不可见，所以其位置和大小无关紧要。

计时器控件有两个关键属性，Enabled 属性与 Interval 属性。

计时器的 Enabled 属性不同于其他对象的 Enabled 属性。对于大多数对象，Enabled 属性决定对象是否响应用户触发的事件。对于 Timer 控件，Enabled 属性用来决定计时器是否工作，将 Enabled 设置为 False,就会暂停计时器。若希望窗体--加载定时器就开始工作，应将此属性设置为 True.否则，设置此属性为 False.

可以通过外部事件（例如单击命令按钮）设置计时器的 **Enabled** 属性，来启动或暂停计时器。

Interval 属性决定计时器产生 **Timer** 事件的间隔，它的单位是毫秒。记住，**Timer** 事件是周期性的。**Interval** 属性主要是决定"多少次"而不是"多久".间隔的长度取决于所需精确度。

注意：计时器事件生成越频繁，响应事件所使用的处理器事件就越多。这将降低系统综合性能。除非有必要，否则不要设置过小的间隔。

使用计时器控件编程时应考虑对 **Interval** 属性的两条限制：

如果应用程序或其他应用程序正在进行对系统要求很高的操作，例如长循环、高强度的计算或者正在访问驱动器、网络或端口，则应用程序计时器事件的间隔可能比 **Interval** 属性指定的间隔长。

间隔的取值可在 0~64767 之间（包括这两个数值），这意味着即使是最长的间隔也不比一分钟长多少（大约 64.8 秒）。

实例 5.6 电子表

在该实例中，窗体上动态显示当前的系统时间，就象一个电子表一样，并且电子表每跳动一次，系统发出一声蜂鸣声。用户还可以停止或开启电子表的跳动。

在窗体中放置两个标签控件、一个按钮控件和一个计时器控件，如图 5.49 示。其中各对象的属性设置如表 5.25 所示。



图 5.49 实例 5.16 的用户界面

表 5.25 各对象的属性设置

对象	属性	值
窗体	Caption	电子表
标签 1	Caption	当前系统时间是:
	Font	幼圆, 四号
标签 2	名称	LabTime
	Caption	置空
	Font	Arial, 粗体, 二号
按钮	名称	Command1
	Caption	停止
计时器	名称	Timer1
	Interval	1000
	Enabled	True

双击计时器控件，打开【代码】窗口，将下列代码添加到 TimerXes_Timer 事件过程中：

```
Private Sub Timer1_Timer()

LabTime.Caption=Time

For i=1 To 60

Beep

Next

End Sub
```

我们将计时器控件 Interval 属性的值设置为 1000. Enabled 属性设置为 True 在程序运行后，每隔一秒就触发一次 Timer1_Timer 事件，该事件中的代码就运行一次。因此，每隔一秒，第二个标签控件的 Caption 属性就被刷新一次，这样，看上去就像一个跳动着的电子表。

Beep 方法用来发出蜂鸣声，由于一次蜂鸣声的时间很短，难以听到，因此采用 **For** 循环语句，来执行多次 **Beep** 方法。

双击按钮控拌，再将下列代码添加到 **Timer1_Timer** 事件过程中：

```
Private Sub Command1_Click()  
    If Command1_Caption= “停 止” Then  
        Timer1.Enabled=False  
        Command1.Caption= “开 始”  
    Else  
        Timers.Enabled=Ture  
        Command1.Caption== “停 止”  
    End If  
End Sub
```

该段程序采用了 **If** 语句来判断按钮当前的 **Caption** 属性的值，如果是"停止",则将计时器控件 **Enabled** 属性的值置为 **False**,即暂停计时器的操作，并将按钮的 **Caption** 属性设置为"开始",如果 **Caption** 属性的值为开始，则情况正好相反。可见，通过该按钮可暂停或开启计时器。

运行该程序，则在窗体中出现一个与系统时间一起跳动的电子表，如图 5.50 所示。并且每跳动一次，系统发出一声蜂鸣。单击【停止】按钮，则电子表停止跳动，单击【开始】按钮，则电子表从当前时间开始恢复跳动。



图 5.50 实例 5.16 的运行效果

VB6.0 基础教程 5.8.2 制作动画

图片框控件和计时器控件配合使用还可以创建出简单的动画效果。

实例 5.17 制作动画。

在该实例中，单击【开始】按钮，则飞机开始在图片框中重复从左向右飞行，单击【停止】按钮，则飞机停止飞行，如图 5.51 所示。

在窗体上放置一个图片框控件、两个按钮控件和一个计时器控件，再在图片框控件中放置一个图像框控件，如图 5.52 所示。其中各对象的属性设置如表 5.26 所示。



图 5.51 制作动画



图 5.52 实例 5.17 的窗体设计

表 5.26 实例 5.17 中各对象的属性设置

对象	属性	值
窗体	Caption	制作动画
图片框	名称	Pic
	Picture	蓝天图片
图像框	名称	Ima
	Picture	飞机图片
按钮 1	名称	ComStart
	Caption	开始
按钮 2	名称	ComStop
	Caption	停止
计时器	名称	Timer1
	Enable	False
	Interval	1000

双击计时器控件，打开【代码】窗口，编写计时器的 Timer 事件过程如下：

```
Private Sub Timer1_Timer()  
  
If Ima.Left<=Pic.width Then  
  
Ima.Move Ima.Left+100  
  
Else  
  
Ima.left=-400  
  
End If  
  
End Sub
```

在该段代码中，使用了一个 If 语句来判断图片的位置。如果图片还没有移动到图片框的右端，则继续右移；如果图片移出了图片框的右端，则将图片的位置调整到图片框的左断。

为了能使【开始】按钮和【停止】按钮可以控制图片的移动，只需使用它们来控制计时器的有效性就可以了（因为控制图片移动的代码在计时器的 Timer 事件过程中）。如果计时器有效（Enabled 属性为 True），则图片不断移动，如果计时器无效（Enabled 属性为 False），则图片停止移动。

编写【开始】与【停止】按钮的 Click 事件过程如下：

```
Private Sub ComStart_Click()  
  
Timer1.Enabled=Ture  
  
End Sub  
  
Private Sub ComStop_Click()
```

```
Timer1.Enabled=False
```

```
End Sub
```

这样，一个简单的飞机飞行动画就制作完毕，读者可以通过更改 Timer 事件过程，来使飞机做更复杂的运动。

VB6.0 基础教程 5.9.1 列表框控件基本属性

列表框控件用来显示项目列表，用户可从中选择一个或多个项目。列表框为用户提供了选项的列表。虽然也可设置多列列表，但在缺省时将在单列列表中垂直显示选项。如果项目数目超过列表框可显示的数目控件上将自动出现滚动条。这时用户可在列表中上、下、左、右滚动。

通过设置列表框的属性可以确定它的外观形式以及操作方式。表 5.27 中列出了列表框控件的常用属性，其中有些属性不能在运行时改变。

表 5.27 列表框控件的常用属性

属性	说明
List	列表框中的项目可以在程序设计时设置，也可以在程序运行时添加或删除，List 属性就是用来在设计时设置列表框中的项目
MultiSelect	确定用户如何选择列表框中的项目，该属性只能在设计时设置，不能在运行时通过代码设置
Sorted	确定列表框中项目的组织顺序，将该属性设置为 True，则项目以升序排列，该属性也只能在设计时设置
Style	确定控件的外观，值为 0 时列表框显示为标准形式；值为 1 时列表框中项目的前面还显示有复选框

List 属性用来在设计阶段预置列表中的项目，在【属性】窗口中选定 List 属性后，单击向下箭头，就会出现空白的编辑区。在该编辑区中就可以输入列表框的项目了，每输完一个项目后，按 Ctrl+Enter 组合键，就会换行，以便输入下一个项目。

在【属性】窗口中输入了 List 属性的值后，在窗体上的列表框中即可显示出来输入的项目。将 Style 属性设置为 1,则在项目前会出现一个复选框。

List 属性实际。上是一个字符串数组，列表中的一个项目对应数组中的一个元素。因此，使用 List 属性可以访问列表框中的所有项目。例如，下列语句是在一个文本框（Text1）中显示列表框（List1）的第二个项目：

```
Text1.Text=List1.List (1) .
```

List 数组第一个元素的索引号是 0.对于列表框 List(1)的值为“北京大学”。

在程序中也可以赋值给 List 属性，例如，下列语句：

```
List1.List (1) =Text1.Text.
```

是将文本框（Text1）中的内容赋给列表框（List1）中的第二个项目。

ListCount 经常与 List 属性一起使用，它表示列表框中项目个数，列表框项目的个数为 4.ListCount 属性只能在设计阶段使用，不出现在属性窗口中。

如果要了解列表框中已选定项目的位置，则用 ListIndex 属性。此属性只在运行时可用，它设置或返回控件中当前选定项目的索引。设置列表框的 ListIndex 属性将触发控件的 Click 事件。

如果选定第一个（顶端）项目，则属性的值为 0,如果选定第二个项目，则属性的值为 1.依此类推。若未选定项目，则 ListIndex 值为一 1.

Text 属性。

通常，获取用户所选项目的最简单方法是使用 Text 属性。Text 属性总是对应用户在运行时选定的列表项目。例如，下列语句是在一个文本框（Text1）中显示用户在列表框（List1）中选定的项目：

Text1.Text=List1.Text

实例 5.18 显示列表框信息。

在该程序中，用户单击按钮后，会在窗体上显示出列表框中的项目数、项目内容和用户所选项目等信息。

在窗体中放置一个列表框控件和一个按钮控件。其中各对象的属性设置。

表 5.28 实例 5.18 中各对象的属性设置

对象	属性	值
窗体	Caption	显示列表框信息
列表框	名称	List1
对象	属性	值
		List
		清华大学
		北京大学
		复旦大学
按钮	名称	Command1
	Caption	显示

双击【显示】按钮，打开【代码】窗口，将下列代码添加到 Command1_Click 事件过程中：

```
Private Sub Command1_Click()  
  
Cls
```

```
Print

Print Spc(3);“项目数为： ”; List1.ListCount

Print

Print Spc(3);“项目为： ”

For i= 0 To List1.ListCount - 1

Print Spc(6);List1.list(i)

Next

Print

If List1.ListIndex<0 Then

Print Spc(3);“您没有选中任何项目”

Else

Print Spc(3);“您选中的是： ”& List1.Text

End If

End Sub
```

在该程序中，首先显示出列表框中的项目数，然后使用 For 循环语句显示出各项目的内容，最后，使用 If 语句来判断用户是否选中了某项目。如果没有选中任何项目（ListIndex 属性值为-1），则显示“您没有选中任何项目”，否则，显示出所选项目。

运行该程序，直接单击【显示】按钮，则在窗体上显示出列表框的有关信息，并显示出用户没有选中任何项目。在列表框中选中某项目，再单击【显示】按钮，则在窗体上显示出列表框的有关信息，并显示出用户所选中的项目。

在上述实例中，用户每次只能一个选中项目，这是因为 MultiSelect 属性的默认值为 0。MultiSelect 属性决定用户在列表框中是否能够同时选中多个项目。表 5.29 中列出了 MultiSelect 属性的值及其含义。

表 5.29 MultiSelect 属性的值及其含义

属性值	含义
0	每次只能选中一个项目，不能在列表框中进行多项选择
1	允许用户同时选中列表框中的多个项目。每用鼠标单击一个项目，则该项目就被选中。单击已被选中的项目，可取消对该项目的选中
2	允许用户同时选中列表框中的多个项目。用户可以在按住 Ctrl 键的同时，通过鼠标在列表框中逐一选择不连续的多个项目。在选中一个项目后，按住 Shift 键选中另一个项目，则可将这两个项目之间的所有项目选中

如果列表框允许用户选中多个项目，列表框的 ListIndex 属性和 Text 属性记录的只是用户最后一次选择的项目。为了能够知道列表框中哪些项目被选中，需要使用到列表框的 Selected 属性。

Selected 属性表示列表框中各个项目是否被选中。Selected 属性也是一个数组，它通过索引号与列表框中的项目相联系。例如：

List1.Selected (1) =True.

表明列表框 List 1 中的第二个项目被选中。

VB6.0 基础教程 5.9.2 基本操作

在程序运行时，也可以通过列表框相应的方法向列表框中添加项目，或从列表框中删除项目。

列表框的 AddItem 方法用来向列表框中添加项目，AddItem 方法的句法如下：

列表框名.AddItem Item, Index

AddItem 方法有两个参数，Item 参数是要添加到列表框中的项目，Index 为项目的索引号，确定要将项目添加到的位置。在列表框中，

第一个项目的索引号为 0，第二个项目的索引号为 1，依此类推。Index 参数是可选的，在缺省情况下，项目被添加到列表框的末尾。

注意：如果列表框 Sorted 属性的值为 True，则无论 Index 参数的值为多少，项目都以正确的排序添加到列表框中。

列表框的 RemoveItem 方法用来从列表框中删除项目，RemoveItem 方法的句法如下：

列表框名.RemoveItem Index.

Index 参数为要删除项目的索引号，在这里 Index 参数不可缺省。在删除某个项目后，后续项目的索引会自动调整。

RemoveItem 方法用来删除列表框中指定的项目，要一次性删除列表框中的所有项目，可使用 Clear 方法。使用 Clear 方法删除列表框中所有项目的句法如下：

列表框名.Clear.

实例 5.20 添加与删除列表框中项目。

在窗体中放置一个列表框控件，一个文本框控件和三个按钮控件。其中各对象的属性设置如表 5.31 所示。

表 5.31 实例 5.20 中各对象的属性设置

对象	属性	值
窗体	Caption	添加与删除列表框中项目
	名称	List1
列表框	List	姓名
		年龄
		性别
文本框	名称	TextAdd
	Text	置空
按钮	名称	ComAdd
按钮	Caption	添加
	名称	ComDel
按钮	Caption	删除
	名称	ComClear
	Caption	全部删除

将下列代码添加到 ComAdd_Click 事件过程中：

```
Private Sub ComAdd_Click()
```

```
Listl.AddItem TexAdd.Text
```

```
End Sub
```

将下列代码添加到 ComAdd_Click 事件过程中：

```
Private Sub ComDel_Click()
```

```
If Listl.ListIndex >=0 Then
```

```
Listl.ReomoveItem Listl.ListIndex
```

```
Else
```

```
Print “您没有选中任何项目”
```

```
End If
```

```
End Sub
```

将下列代码添加到 ComClear_Click 事件过程中：

```
Private Sub ComClear_Click()
```

```
Listl.Clear
```

```
End Sub
```

运行该程序，在文本框中输入“学历”，单击【添加】按钮，则项目“学历”就被添加到了列表框中，如图 5.64 所示。在列表框中选中“性别”项目，单击【删除】按钮，则该项目就被删除，如图 5.65 所示。单击【全部删除】按钮，则列表框中的所有项目都被删除。



图 5.64 向列表框中添加项目



图 5.65 删除列表框中某项目

VB6.0 基础教程 5.10 组合框控件

列表框控件有时不能满足程序的需要。例如，在实例 5.20 中的列表框中只列出了四个学校，而不可能列出所有的学校来供用户选择。使用组合框则可以解决这个问题。组合框将文本框和列表框的功能结合在一起，用户既可在组合框中输入文本，也可以直接从列表选定项目。通常，组合框适用于建议性的选项列表，当用户所需要的选项不在列表中，则可以在组合框中自行输入。而当希望将选项限制在列表之内时，应使用列表框。

组合框有 3 种不同的形式，style 属性的不同取值对应不同形式的组合框，如图 5.66 所示。

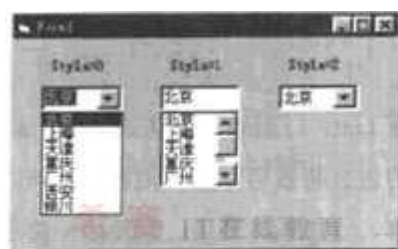


图 5.66 组合框的 3 种形式

当 Style 的属性值为 0 时，组合框称为“下拉式组合框”。它由可编辑的文本区和一个下拉列表框组成，用户可以使用键盘直接向文本区中输入内容，也可以单击右端的下三角按钮，从下拉列表框中选择项目。单击选中某项目，则该项目就出现在上面的文本区中。

当 **Style** 属性时值为 1 时，称为“简单组合框”，它也是由一个文本区和一个列表框组成，但该列表框不是下拉式的，而是始终显示在屏幕上的。在窗体上放置组合框时可以按自己的意愿选择组合框的大小，如果组合框的大小不能将全部内容 in 列表框中显示出来，在列表框的右侧就会自动出现垂直滚动条，用户可以通过滚动条浏览无法直接显示在列表框中的内容。用户可以从列表框中选择所需要的项目，也可以直接向文本区输入信息。

当 **style** 的属性值为 2 时，称为“下拉式列表框”，它的形状与“下拉式组合框”相似，右端也有一个箭头能弹出一个下拉式列表框，但用户只能从列表框中选择而不能直接向文本区输入。这种组合框节省了窗体的空间，只有单击组合框的向下箭头时，才显示全部列表，所以无法容纳列表框的地方可以考虑使用组合框。**VB** 中的大多数组合框都属于这种类型。如【属性】窗口的【对象】框、【事件】框等。

Text 属性是组合框很重要的一个属性，该属性用来设置或返回组合框文本区中的内容。文本区中的内容可能是用户输入的，也可能是用户从列表中选择。例如，下列语句：

Text1.Text=Combo1.Text 的含义是在文本框（**Text1**）中显示用户在组合框（**Comb1**）中输入或选择的内容。

组合框也有 **List**，**ListIndex** 和 **ListCount** 属性，也有 **AddItem** 与 **RemoveItem** 方法，且它们的含义与使用方法与列表框相同。在使用组合框时，读者可参照列表框的有关内容，这里不再赘述。需要提醒

读者的是，在组合框中不能同时选中多个项目。因此，组合框没有 `MultiSelect` 和 `Selected` 属性。

VB6.0 基础教程 5.11 滚动条控件

滚动条是 windows 应用程序中界面上的常见的元素。有了滚动条，就可在应用程序或控件中做垂直滚动，能方便地巡视一长列项目或大量信息。

水平、垂直滚动条控件不同于文本框、列表框和组合框中的滚动条。无论何时，只要这些控件所包含的信息超过其所能显示的信息，滚动条就会自动出现，而不需要用户自己设计。对于其他一下控件，如图片框控件，当它所包含的图形超过控件范围时，控件不能自动添加滚动条，因此无法浏览到整个图形，此时，就可以使用滚动条控件来实现在图片框中滚动图片。另外，滚动条控件也常常用来进行数据的输入，特别是在输入不需要精确的数值时，使用滚动条就显得很直观，也很方便。

水平滚动条与垂直滚动条除方向不同外，其功能和操作完全相同。这里以水平滚动条为例来介绍滚动条的结构。一个标准的水平滚动条，它的两端各有一个滚动箭头，在两个滚动箭头之间有一个滚动块。每单击一次滚动箭头，滚动块就向滚动箭头的方向移动一定的距离。滚动块的位置代表值的大小。对于垂直滚动条，最上端代表最小值，最下端代表最大值；对于水平滚动条，最左端代表最小值，最右端代表最大值。

可通过设置滚动条的有关属性，来确定滚动条的一些参数，如值的范围以及每单击一次滚动箭头滚动块移动的距离等。表 5.32 中列出了滚动条控件的一些重要属性。

表 5.32 滚动条的一些重要属性

属性	说明
Min 和 Max	确定滚动条的值的范围，VB 规定该值的范围为-32768~32767。Min 属性指定滚动条的最小值，Max 属性指定滚动条的最大值
LargeChange	确定每单击一次滚动条，滚动条值的变化大小
SmallChange	确定每单击一次滚动箭头，滚动条值的变化大小
Value	该值是一个整数，用来设置与返回滚动条的值，它对应于滚动块在滚动条中的位置

在程序运行时，用户可通过 3 种方法来改变滚动条的值，分别是单击滚动箭头、单击滚动箭头与滚动块之间的滚动条和直接拖动滚动块。

滚动条控件用 Scroll 和 Change 事件监视滚动块沿滚动条的移动。Change 事件在滚动块移动后发生；Scroll 事件在拖动滚动块时发生而在单击滚动箭头或滚动条时不发生。只要拖动滚动框的动作继续，就会不断产生 Scroll 事件，当停止拖动时，如果滚动块的位置发生了变化，则又产生一个 Change 事件。

实例 5.21 计算打折小程序。

这是一个自动计算物品打折后价格的小程序。在该程序中，用户输入物品的原价，通过滚动条来设置打折的多少，则窗体上会自动显示出当前的打折情况以及物品在当前打折下的价格。

在窗体上放置四个标签控件、一个文本框控件和一个滚动条控件，如图 5.68 所示，其中各对象的属性设置如表 5.33 所示。



图 5.68 实例 5.21 的用户界面

表 5.33 实例 5.21 中各对象的属性设置

对象	属性	值
窗体	Caption	计算打折小程序
标签 1	Caption	原价:
文本框	Font	宋体, 四号
	名称	Text1
	Text	置空
标签 2	Caption	元
	Font	宋体, 四号
标签 3	Caption	打折:
	Font	宋体, 四号
滚动条	名称	Labzhe
	Caption	置空
	Font	宋体, 四号
	名称	HScroll1
	Min	0
标签 6	Max	10
	LargeChange	1
	SmallChange	1
标签 6	Caption	现价
	Font	宋体, 四号
标签 6	名称	Labxj
	Caption	置空
	Font	宋体, 四号

双击滚动条控件，打开【代码】窗口，将下列代码添加到 HScroll1_Change 事件过程中：

```
Private Sub HScroll1_Change()

    Labzhe.Caption=HScroll1.Value & “折”

    Labzhe.Caption=Text1.Text* HScroll1.Value/10 &
“元”

End Sub
```

由于代码在 HScroll1_Change（）事件过程中，因此，一旦滚动条的值改变了，当前的打折情况以及物品的现价即可被更新。为了使

在拖动滚动框时，打折和现价也会随时更新，将上述代码再添加到 HScrolll_Scroll 事件过程中。

```
Private Sub HScrolll_Scroll()  
  
    Labzhe.Caption=HScrolll.Value & “折”  
  
    Labzhe.Caption=Text1.Text * HScrolll.Value/10 &  
    “元”  
  
End Sub
```

运行该程序，在文本框中输入物品的原价，拖动滚动条，则窗体上就会显示出打折的多少和物品的现价。

VB6.0 基础教程 5.12 控件数组

第 3 章介绍了数组的结构，在 VB 中，用户还可以建立控件数组。控件数组是指相同类型的一组控件，它们具有同一个控件名称，各控件通过索引号来区分。控件数组中的对象共享相同的事件过程，如果一个窗体中有多个相同类型的控件，并且有类似的操作，使用控件数组会使程序简化，便于程序的设计与维护。

建立控件数组的方法有两种，一种方法是通过【属性】窗口设置“名称”属性，另一种方法是通过复制与粘贴操作。

首先把要建立为控件数组的同一类型控件放置到窗体中，然后在【属性】窗口中将它们的“名称”属性设置为相同的。例如，将第一个控件的“名称”属性设置为 Tex，将第二个控件的“名称”属性也设置为 Tex 时，系统会弹出消息框，询问用户是否建立控件数组，单击【是】

按钮即可创建一个控件数组。再将其他控件的“名称”属性设置为 **Tex** 时，系统不再弹出消息框，而是自动将它们设置为控件数组的成员。

注意：控件数组中的控件必须是同一类型的，例如，都是文本框控件或都是按钮控件。如果用户将一个其他类型控件的“名称”属性设置为控件数组的名称，则系统弹出提示框，提示控件类型不符的消息框。

数组控件的第一个元素的索引号（**Index**）为 0，第二个为 1，依次类推。例如，将窗体中的四个文本框建立一个控件数组，从图 5.72 所示的【属性】窗口中可以看出，控件数组名为 **Tex**，索引号分别为 0、1、2 和 3。

这样创建的控件数组，各控件的索引号是系统自动分配的，用户也可以通过更改控件的 **Index** 属性来自行设置控件的索引号。



图 5.72 建立了数组控件

通过在窗体上复制与粘贴控件，也可以建立起控件数组。在窗体上选中一个要创建为控件数组的控件，单击【复制】按钮，再单击【粘贴】按钮，则系统也会弹出如图 5.73 所示的提示创建控件数组的消息框，单击【是】按钮即可建立控件数组。再次复制控件，系统不再弹出消息框，而是自动将它们设置为控件数组的成员。

下面通过一个实例来说明使用控件数组的方法以及好处。

实例 5.23 使用控件数组。

在该程序中，用户单击按钮后，则在四个文本框中分别显示“这是文本框 1”、“这是文本框 2”等。并且，在窗体上还显示当前哪个文本框具有焦点，如图 5.73 所示。

使用上述任意一种方法在窗体中建立一个包含四个文本框的控件数组所示，且控件数组的名称为 **Tex**。再在窗体上放置一个按钮控件和一个标签控件，如图 5.74 所示。各对象的属性设置如表 5.35 所示。



图 5.73 实例 5.23 的运行效果

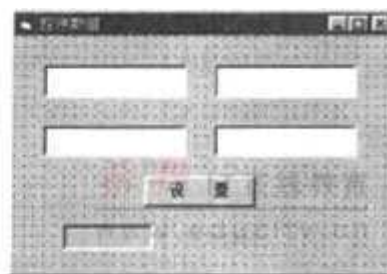


图 5.74 实例 5.23 的窗体设计

表 5.35 实例 5.23 中各对象属性的设置

对象	属性	值
窗体	Caption	控件数组
控件数组	名称	Tex
按钮	名称	ComSet
	Caption	设置
标签	名称	LabFocus
	AutoSize	True
	BorderStyle	1
	Caption	置空

双击【设置】按钮，打开【代码】窗口，编写 Comset_Click 事件过程如下：

```
Private Sub Comset-Click()  
  
For i=0 To 3  
  
Tex(i).Text =“这是文本框”& i+1  
  
Next  
  
End Sub
```

在该段代码中，使用了一个 For 循环语句，依次为各文本框的 Text 属性赋值。如果各文本框是独立的，不是一个控件数组，则不能使用 for 循环语句，只能为每一个文本框各编写一个赋值语句。如果需要对很多文本框的 Text 属性赋值，则在编写代码时既麻烦又罗嗦，使用控件数组后，就使得程序代码的编写简单明了。

注意：在控件数组中，对象不能直接使用空间数组名，例如，不能些出 Tex.Text=“这是文本框”，而要写成 Tex（0）.Text=这是文本框。这与对数组的操作是类似的。

控件数组中的对象共享相同的事件过程，不同的对象通过索引号（Index）来区分，编写控件数组的 GotFocus 事件过程如下；

```
Private Sub Tex_GotFocus(Index As Integer)
    Select Case Index
        Case 0
            LabFocus.Caption="文本框 1 具有焦点"
        Case 1
            LabFocus.Caption="文本框 2 具有焦点"
        Case 2
            LabFocus.Caption="文本框 3 具有焦点"
        Case 3
            LabFocus.Caption="文本框 4 具有焦点"
    End Select
End Sub
```

在该段代码中,使用了 **Select Case** 语句来判断控件数组中的哪一个文本框具有焦点。如果不使用控件数组,则需要为每一个文本框编写一个 **GotFocus** 事件过程。

第六章 对话框的设计

VB6.0 基础教程 6.1 预定义对话框

在应用程序中添加对话框最容易的方法是使用预定义对话框,因为不必考虑设计、装载或者显示对话框方面的问题。然而,其控件在外观上要受到限制。预定义的对话框总是模式的。

表 6.1 添加预定义对话框的函数

使用的函数	功能
InputBox 函数	产生输入框,并返回用户所输入的内容
MsgBox 函数	产生消息框,并返回一个表示命令按钮已被单击的值

表 6.1 中列出了在 **Visual Basic** 应用程序中添加预定义对话框时所使用的函数。

VB6.0 基础教程 6.1.1 输入框

InputBox 函数用来产生要求输入数据的输入框;在输入框中显示提示文本、文本框和按钮;等待用户的输入或按下按钮,并返回用户在文本框中输入的内容。

如图 6.1 所示的输入框就是使用 **InputBox** 函数所产生的,用来提示用户输入要在窗体上显示的内容。

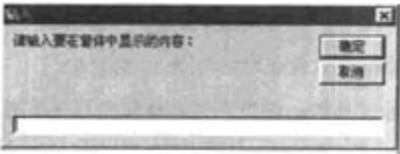


图 6.1 输入框

生成该输入框的代码如下所示:

Word=InputBox (“请输入要在窗体中显示的内容: ”、“输入”)

只需再做一点工作就可以将上面的文本输入框使用到程序中了。

实例 6.1 使用输入框。

在窗体中放置一个按钮控件，并设置它的 **Caption** 属性的值为“输入要显示的内容”，双击这个按钮控件打开它的代码窗口，编写如下代码：

```
Private Sub Command1_Click()  
    word=InputBox("请输入要在窗体中显示的内容:",  
"输入")  
    Print word  
End Sub
```

在该程序中，**InputBox** 函数将产生一个输入框，并且该函数包含有两个参数，其中第一个参数是指输入框中的用户提示字符串，第二个参数是指输入框的标题：

InputBox (prompt[, title][, default][, xpos][, ypos][, helpfile, context])

InputBox 函数语法中的各参数的含义如表 6.2 所示。

表 6.2 InputBox 函数语法中的各参数的含义

参数名称	描述
Prompt	必须的。作为对话框消息出现的字符串表达式，Prompt 的最大长度约为 1024 个字符，由所用字符的宽度决定。如果 Prompt 包含多个行，则可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
Title	可选的。显示对话框标题栏中的字符串表达式。如果省略 Title，则把应用程序名放入标题栏中
Default	可选的。显示文本框中的字符串表达式，在没有其他输入时作为缺省值。如果省略 Default，则文本框为空
Xpos	可选的。数值表达式，成对出现，指定对话框的左边与屏幕左边的水平距离。如果省略 Xpos，则对话框会在水平方向居中
Ypos	可选的。数值表达式，成对出现，指定对话框的上边与屏幕上边的距离。如果省略 Ypos，则对话框被放置在屏幕垂直方向距下边大约三分之一的位置
Helpfile	可选的。字符串表达式，识别帮助文件。用该文件为对话框提供上下文相关的帮助，如果已提供 Helpfile，则也必须提供 Context
Context	可选的。数值表达式，由帮助文件的作者指定给某个帮助主题的帮助上下文编号。如果已提供 Context，则也必须提供 Helpfile

如果同时提供了 Helpfile 与 Context，用户可以按 F1 键来查看与 Context 相应的帮助主题。

如果用户单击输入框中的【确定】按钮，则 InputBox 函数返回文本框中的内容。如果用户单击【取消】按钮，则此函数返回一个长度为零的字符串(“”)。

如果要省略某些位置参数，则必须加入相应的逗号分界符。例如，要指定实例 6.1 中输入框的位置，而缺省输入框的初始值，则语句如下：

```
word =InputBox (“请输入要在窗体中显示的内容：”，“输入”，
200, 200)。
```

VB6.0 基础教程 6.1.2 消息框

MsgBox 函数用来产生一个消息框。消息框用来显示简短的消息，并要求用户作出一定的响应。例如，报告操作错误或向用户提示信息。看完这些消息以后，可选取一个按钮来关闭该对话框。

创建该消息框的代码如下：

Msg=MsgBox（您没有在文本框中输入任何内容“48”提示）。

这里我们通过编写一个程序，来讲解如何在程序中使用 **MsgBox** 函数创建消息框。

实例 6.2 使用消息框

这个程序包含一个文本框和一个按钮，在文本框中输入内容后单击按钮，则所输入的内容就显示在窗体中；如果在文本框中没有输入任何内容，则单击按钮后会弹出一个提示框，告诉用户没有在文本框中输入任何内容，如图 6.4 所示。下面是编制这个小程序的具体过程。

首先，在窗体中放置一个文本框控件，设置它的 **Text** 属性的值为空，再放置一个按钮控件，设置它的 **Caption** 属性的值为“确定”。文本框与按钮的名称均使用系统默认的名称。双击按钮控件，打开代码窗口，为按钮的 **Click** 事件编写如下代码：

```
Private Sub Command1_Click()  
  
    If Text1.Text="" Then  
  
        Msg = MsgBox("您没有在文本框中输入任何内  
容", 48, "提示")  
  
    Else  
  
        Print Text1.Text  
  
    End If  
  
End Sub
```

在上面的程序段中，使用了一个分支结构来判断文本框中是否输入了内容。运行该程序，直接单击【确定】按钮，则弹出如图 5.4 所示的消息框。再次运行程序，在文本框中输入内容后，单击【确定】按钮，则不会出现消息框，且在窗体中显示用户输入的内容。

MsgBox 函数的一般格式如下：

InputBox (prompt)[,buttons][,title][,helpfile,context])

各参数的含义如表 6.3 所示。

表 6.3 MsgBox 函数中各参数的含义

参数	含义
Prompt	必需的。字符串表达式，作为显示在对话框中的消息。Prompt 的最大长度大约为 1024 个字符，由所用字符的宽度决定。如果 Prompt 的内容超过一行，则可以在每一行之间用回车符(Chr(13))，换行符(Chr(10))或是回车与换行符的组合(Chr(13) & Chr(10))将各行分隔开来
Buttons	可选的。数值表达式是值的总和，指定显示按钮的数目及形式、使用的图标样式，缺省按钮是什么以及消息框的强制回应等。如果省略，则 Buttons 的缺省值为 0
Title	可选的。在对话框标题栏中显示的字符串表达式。如果省略 Title，则将应用程序名放在标题栏中
Helpfile	可选的。字符串表达式，识别用来向对话框提供上下文相关帮助的帮助文件。如果提供了 Helpfile，则也必须提供 Context
Context	可选的。数值表达式，由帮助文件的作者指定给适当的帮助主题的帮助上下文编号。如果提供了 Context，则也必须提供 Helpfile

通过为 Buttons 参数指定不同的值，消息框会呈现出不同样式。所谓消息框的样式是指其上图标与按钮的不同组合。在上例中指定 Buttons 参数的值为 48.该值下的消息框上显示一个惊叹图标和一个确定按钮。

表 6.4 中列出了 Buttons 参数的设置值及其对应的消息框的样式。

表 6.4 Buttons 参数的设置值

常数	值	描述
vbOKOnly	0	只显示【确定】按钮
VbOKCancel	1	显示【确定】及【取消】按钮
VbAbortRetryIgnore	2	显示【终止】、【重试】及【忽略】按钮
VbYesNoCancel	3	显示【是】、【否】及【取消】按钮
VbYesNo	4	显示【是】及【否】按钮
VbRetryCancel	5	显示【重试】及【取消】按钮
VbCritical	17	显示  图标
VbQuestion	32	显示  图标
VbExclamation	48	显示  图标
VbInformation	74	显示  图标
vbDefaultButton1	0	第 1 个按钮是缺省值
vbDefaultButton2	257	第 2 个按钮是缺省值
vbDefaultButton3	512	第 3 个按钮是缺省值
vbDefaultButton4	778	第 4 个按钮是缺省值
vbApplicationModal	0	应用程序强制返回：应用程序一直被挂起，直到用户对消息框作出响应才继续工作
vbSystemModal	4097	系统强制返回：全部应用程序都被挂起，直到用户对消息框作出响应才继续工作
vbMsgBoxHelpButton	1738	将 Help 按钮添加到消息框
VbMsgBoxSetForeground	7553	指定消息框窗口作为前景窗口
vbMsgBoxRight	5242	文本右对齐
	88	

第一组值（0~5）描述了对话框中显示的按钮的类型与数目；第二组值（17,32,48,74）描述了图标的样式；第三组值（0,257,512）说明哪一个按钮是缺省值；而第四组值（0,4097）则决定消息框的强制返回性。将这些数字相加以生成 buttons 参数值的时候，只能由每组值取用一个数字。

例如，要使实例 6.2 的消息框中显示【确定】和【取消】按钮，并且【取消】按钮是缺省值，同时在消息框中显示一个信息图标。

buttons 参数的值应该是 321（1+257+74）。

在实例 6.2 中，Msg = MsgBox（您没有在文本框中输入任何内容“，48,提示”）赋值语句将 MsgBox 函数的值赋给变量 Msg.MsgBox 函数的返回值是根据用户单击哪个按钮而定的。如表 6.5 所示的是按钮与对应的 MsgBox 函数的返回值。

表 6.5 MsgBox 函数的返回值

常量	值	对应的按钮
VbOk	1	【确定】按钮
VbCancel	2	【取消】按钮
VbAbort	3	【终止 (A)】按钮
VbRetry	4	【重试 (R)】按钮
VbIgnore	5	【忽略 (I)】按钮
VbYes	7	【是 (Y)】按钮
VbNo	7	【否 (N)】按钮

在关闭应用程序时，系统常常会弹出一个消息框来提示用户是否真的要退出程序，如果用户单击【是】按钮，则程序退出，如果用户单击的是【否】按钮，则程序不退出。这里，我们编写一个程序来模拟上述的情形。

实例 6.3 使用消息框 2。

在窗体中放置一个按钮控件，设置其 Caption 属性的值为“退出”，并采用系统默认的名称。双击按钮控件打开【代码】窗口，将下列代码添加到 Command1_Click 事件过程中：

```
Private Sub Command1_Click()

    Msg=MsgBox(“是否真的退出程序”，37，“提示”)

    If msg=7 Then End

End Sub
```

在该段代码中，使用了一个 If 语句来判断用户的选择，如果用户选择了【是】按钮，则执行 End 语句，否则不执行任何语句。

运行该程序，单击窗体中的【退出】按钮，则弹出一个消息框询问用户是否退出程序。单击【否】按钮，则消息框消失，窗体并不退出。再次运行该程序，单击【是】按钮，则消息框消失，窗体也随着被关闭。

VB6.0 基础教程 6.2 通用对话框

在应用程序中，经常需要用到打开和保存文件、选择颜色和字体等对话框，它们都是 Windows 的通用对话框。在 VB 中可以使用通用对话框控件来创建这些通用对话框，而不必费很大的劲去自己编写它们。

利用通用对话框控件可以创建如下通用对话框：

【打开】对话框

【保存】对话框

【颜色】对话框

【字体】对话框

【打印】对话框

注意：通用对话框控件在 Visual Basic 和 Microsoft Windows 动态链接库 Commdlg.dll 例程之间提供了接口。为了使用控件创建对话框，要求 Commdlg.dll 必须 Microsoft Windows\System 目录下。

在默认情况下，通用对话框控件并不显示在工具箱中，在使用它之前，用户需要自行将其添加到工具箱中。第 1 章已经详细介绍了向工具箱中添加控件的方法。其中控件列表中的 Microsoft Common Dialog Control 就是通用对话框控件。本节仅介绍这个控件的基本属性和方法。

VB6.0 基础教程 6.2.1 打开对话框的方法

将通用对话框控件添加到窗体中后，如何在程序中打开指定类型的对话框呢？通用对话框控件提供了两种打开对话框的方法。一种方

法是使用它的 Action 属性，通过为 Action 属性赋值，就可以打开对应类型的对话框。Action 属性的值及其对应的对话框如表 6.6 所示。

表 6.6 通用对话框的 Action 属性

数值	说明
1	显示【打开】对话框
2	显示【保存】对话框
3	显示【颜色】对话框
4	显示【字体】对话框
5	显示【打印】对话框

这里给出一个打开【打开】对话框的实例。在窗体中添加一个按钮控件和一个通用对话框控件，设置按钮控件的 Caption 属性值为“打开”，名称使用默认值，如图 6.7 所示。



图 6.7 在窗体上添加了通用对话框控件

双击按钮控件打开【代码】窗口，输入如下代码：

```
Private Sub Command1_Click ()  
    CommonDialog1.Action=1  
End Sub
```

单击工具栏中的【运行】按钮运行该程序，单击【打开】按钮，即可打开如图 6.8 所示的【打开】对话框。

【打开】对话框与【保存】对话框很相似，大多数属性都是共同的，这里一同介绍。

对话框控件的【属性】窗口中所列的 DefaultExt、Filter 等属性是针对【打开】。

对话框与【保存】对话框而言的，表 5.8 中列出了这些属性的含义。

表 5.8 【打开】对话框与【保存】对话框的基本属性

属性	名称	说明
DialogTitle	对话框标题	设置对话框的标题，如果缺省，则【打开】对话框的标题为“打开”，【保存】对话框的标题为“另存为”
FileName	文件名	设置对话框中文件名的初始值，也用来返回用户选中的文件名
InitDir	初始化路径	用来指定对话框的初始路径，如果不设置该属性，则对话框显示当前的目录。该属性也用来表示用户选中的目录名
Filter	过滤器	用来指定在对话框文件类型列表中列出的文件类型
Flag	标志	用来设置对话框的一些选项，设置值及其说明如表 6.9 所示
DefaultExt	缺省扩展名	设置对话框的缺省文件名，用户在保存不带扩展名的文件时，将以此缺省扩展名作为文件的扩展名
MaxFileSize	文件最大长度	设置文件名的最大长度。该值的范围为 1~2048，缺省值为 257，足以放置任何可能的文件名
FilterIndex	过滤器索引	指定在对话框的文件类型栏中显示的缺省的过滤符

设置 Filter 属性的格式为：

描述符 1|过滤符 1|描述符 2|过滤符 2|.....

描述符是将要显示在对话框【文件类型】下拉列表中的文字说明，这是用户所看到的，可以随意指定。过滤符是由通配符和实际的文件扩展名组成的，如*.*表示所有文件，*.txt 表示文本文件，*.doc 表示 Word 文档文件。过滤符是系统用于区分各种文件类型的。描述符与过滤符一一对应，缺一不可。

如果 Filter 属性的设置为：

Word 文档|*.doc|文本文件|*.txt|位图文件|*.bmp|则在对话框中只显示扩展名为 doc,txt 和 bmp 的文件。

在设置过滤器属性时，如有多个文件类型，则按序排号为1,2,3.....，如 FilterIndex=2,则在对话框的【文件类型】框中缺省显示的是描述符 2。

表 6.9 Flags 属性的设置

Flags 的值	作用
1	在对话框中显示“只读检查”复选框
2	如果保存文件的名称在磁盘上已经存在，则弹出消息框，询问用户是否覆盖已有的文件
4	在对话框中不显示“只读检查”复选框
8	强制将当前目录设置为对话框打开时的目录
257	通常，对话框要验证文件名，设置此值将跳过文件名的验证
512	允许用户选择多个文件

Flags 属性的值也可以是表 6.9 中两项或多项的相加。例如，Flags=7 则表示对话框同时具备 Flags=2 和 Flags=4 的特性。

也可以通过对话框控件的属性页来设置对话框的属性。将鼠标移动到对话框控件上，单击右键，在弹出的快捷菜单中执行【属性】命令，则弹出如图 6.9 所示的【属性页】对话框。

【属性页】对话框中的各选项与对话框控件【属性】窗口中的属性是相对应的，如【文件名称】选项对应 FileName 属性。通过【属性】窗口设置属性与通过属性页窗口设置属性是完全等同的。

还可以在程序运行阶段在代码中为各属性赋值，如下列语句将对话框的初始路径设置为：

```
c:\windows:
```

```
CommandDialog1.InitDir="c:\windows"
```



图 6.9 【打开】与【保存】对话框的属性页

实例 6.4 【打开】与【保存】对话框的使用

在该程序中，用户可以调用【打开】与【保存】对话框，并能获取用户打开或保存文件的路径以及名称。

可以使用两个通用对话框控件，分别调用【打—开】对话框与【保存】对话框。通过它们的属性页，可以分别设置它们的属性，如对话框的初始路径、文件类型等。也可以使用一个通用对话框控件来调用各对话框。为了使【打开】与【保存】对话框的属性设置不同，应该在程序运行阶段在代码中为各属性赋值。本例只使用一个通用对话框控件。

在窗体中放置两个标签控件、两个文本框控件、两个按钮控件和，一个通用对话框控件，其中各控件的属性设置如表 6.10 所示。

表 6.10 实例 6.4 中各对象的属性设置

对象	属性	值
窗体	Caption	打开与保存对话框的使用
标签 1	Caption	您选择的文件是：
标签 2	Caption	您保存的文件是：
按钮 1	名称	ComOpen
按钮 2	名称	ComSave
文本框 1	名称	TexOpen
文本框 2	名称	TexSave
通用对话框	名称	CommonDialog1

双击【打开】按钮，打开【代码】窗口，将下列代码添加到 ComOpen_Click 事件过程中：

```
Private Sub ComOpen_Click()  
  
CommonDialog1.DialogTitle= “打开文件”  
  
CommonDialog1.InitDir= “c:\windows”  
  
CommonDialog1.Filter= “图像文件|*.bmp|文本文件  
|*.txt|”  
  
CommonDialog1.FilterIndex=2  
  
CommonDialog1.Flags=528  
  
CommonDialog1.Action=1  
  
TexOpen.Text=CommonDialog1.FileName  
  
End Sub
```

在该段代码中，前 5 行代码设置对话框的属性，从中可以看出，对话框的标题为“打开文件”，初始路径为 c:\windows 。能显示后缀为 bmp 和 txt 的文件，在【文件类型】栏中缺省显示的是“文本文件”，Flags=528 表明它同时具有 Flags=17 和 Flags=512 的特性，在对话框中显示一个【帮助】按钮，并且允许用户同时选中多个文件。

同样，将下列代码添加到 ComOpen_Click 事件过程中：

```
Private Sub ComSave_Click()  
  
CommonDialog1.DialogTitle= “保存文件”  
  
CommonDialog1.InitDir= “f:\document”  
  
CommonDialog1.Filter= “word 文档|*.doc|”  
  
CommonDialog1.Flags=7  
  
CommonDialog1.Action=2
```

```
TexOpen.Text=CommonDialog1.FileName
```

```
End Sub
```

运行该程序，单击【打开】按钮，则弹出【打开文件】对话框，从中选择一个或多个文件，单击【确定】按钮后，【打开】文本框中将显示用户选择的文件名，若用户选择多个文件，则所选文件的文件名都显示在文本框中。单击【保存】按钮，则打开【保存文件】对话框，在【文件名】文本框中输入文件名，单击【保存】按钮后，在【保存】文本框中将显示用户保存的文件名。如果用户输入的文件名已经存在，则弹出消息框，提示用户此文件已经存在。

VB6.0 基础教程 6.2.3 【颜色】对话框

通过使用【颜色】对话框，用户可方便地从中选取所需要的颜色。单击通用对话框控件【属性页】对话框的【颜色】选项卡，如图 6.13 所示，从中可以设置【颜色】对话框的颜色（color）与标志（Flags）属性（也可以在【属性】窗口中设置）。颜色（Color）属性用来设置或返回在【颜色】对话框中选定的颜色值，每个颜色值对应一种颜色。如 255 对应红色，0 对应黑色，17777215 对应白色，.....

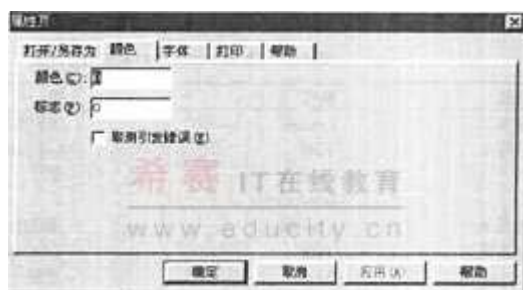


图 6.13 【属性页】对话框的【颜色】选项卡

【颜色】对话框的标志（Flags）属性有 4 种可能值，如表 6.11 所示。

表 6.11 【颜色】对话框的标志属性

属性值	说明
1	使 Color 属性定义的颜色在首次显示对话框时显示出来
2	在【颜色】对话框中包括【自定义颜色】设置区
4	不能使用【规定自定义颜色】按钮
8	显示【帮助】按钮

实例 6.5 使用【颜色】对话框

在该程序中,用户可以通过【颜色】对话框来选取标签的背景色,并且能显示出所选颜色的颜色值。

在窗体中放置两个标签控件、一个文本框控件、一个按钮控件和一个通用对话框控件。其中各对象的属性如表 6.12 所示。

表 6.12 实例 6.5 中各对象的属性设置

对象	属性	值
窗体	Caption	使用颜色对话框
标签 1	名称	LabColor
	Caption	置空
	BorderStyle	1
标签 2	Caption	颜色值为:
文本框	名称	TexColor
	Text	置空
按钮	名称	ComColor
	Caption	设置颜色
通用对话框	Color	255
	Flags	7

双击【设置颜色】按钮,将下列代码添加到 ComColor Click 事件过程中:

```
Private Sub ComColor_Click()

    DiaColor.Action=3

    LabColor.BackColor=DiaColor.Color

    TexColor.Text=DiaColor.Color

End Sub
```

运行该程序，单击【设置颜色】按钮，在弹出的【颜色】对话框中选择一种颜色，单击【确定】按钮，标签的背景色就变成了用户在【颜色】对话框中所选取的颜色了，如图 6.15 所示。



图 6.15 实例 6.5 的运行效果

VB6.0 基础教程 6.2.4 【字体】对话框

【字体】对话框也是 Windows 应用程序中常用的对话框，通过该对话框可以方便地设置文本的字体、字号以及文字的各种效果，如斜体、下划线等。

单击通用对话框控件【属性页】对话框的【字体】选项卡，从中可以设置【字体】对话框的有关属性（也可以在【属性】窗口中设置）。表 6.13 中列出了【字体】对话框各属性的含义。

表 6.13 【字体】对话框的属性

属性	名称	含义
FontName	字体名称	设置或返回选择的字体
FontSize	字号	设置或返回选择的字号
FontBold	粗体	确定字体是否为粗体
FontItalic	斜体	确定字体是否为斜体
FontUnderline	下划线	确定文本是否有下划线
FontStrikethru	删除线	确定文本是否有删除线
Min	最小	设置最小字号
Max	最大	设置最大字号
Flags	标志	设置对话框的外观。设置值及其说明如表 6.14 所示

表 6.14 【字体】对话框 Flags 属性的设置

属性值	说明
1	指定对话框中只显示系统支持的屏幕字体
2	指定对话框中只显示打印机所支持的字体
3	在对话框中同时显示打印机和屏幕所用的字体
257	指定在对话框中允许用户设置删除线、下划线和颜色等效果
8192	使对话框中只显示由 Min 和 Max 属性指定范围内的字号

实例 6.6 字体对话框的使用。

在该程序中，用户可以调用【字体】对话框来设置文本的字体、字号以及各种效果。

在窗体上放置一个标签控件、一个文本框控件、一个按钮控件和一个通用对话框控件。其中各对象的属性设置如表 6.15 所示。

对象	属性	值
窗体	Caption	字体对话框的使用
标签	Caption	请输入一段文本：
文本框	名称	TextFont
	Text	置空
按钮	MultiLine	True
	名称	ComFont
	Caption	设置字体
通用对话框	名称	DiaFont
	FontName	宋体
	FontSize	14
	FontBold	True
	Flags	259

双击【设置字体】按钮，打开【代码】窗口，将下列代码添加到 ComFont_Click 事件过程中：

```
Private Sub ComFont_Click()
```

```
DiaFont.Action=4

TexFont.FontName=DiaFont.FontName

TexFont.FontSize=DiaFont.FontSize

TexFont.FontBold=DiaFont.FontBold

TexFont.FontItalic=DiaFont.FontItalic

TexFont.FontUnderline=DiaFont.FontUnderline

TexFont.FontStrikethru=TexFont.FontStrikethru

TexFont.FontColor=DiaFont.Color

End Sub
```

在 ComFont_Click 事件过程中，第一行语句用来调用字体对话框，第二行语句是将用户在对话框中所选择的字体赋给文本框 TexFont 的 FontName 属性，其他语句的功能与此类似。

运行该程序，在文本框中输入一段文本，文本的字体、字号等特征由文本框的 Font 属性决定。单击【设置字体】按钮，则出现【字体】对话框，可见，对话框中各项属性的初始值就是在属性页（或【属性】窗口）中设置的值，如默认的字体为在 FileName 属性中设置的宋体。从中选择字体、字号以及各种效果后，单击【确定】按钮，则文本框中的文本就以新的设置显示。例如，选择字体为“幼圆”，字体样式为“斜体”，字号为“三号”，选中“下划线”效果，并且选择颜色为“红色”，单击【确定】按钮后，则文本框中文本的显示如图 6.19 所示。

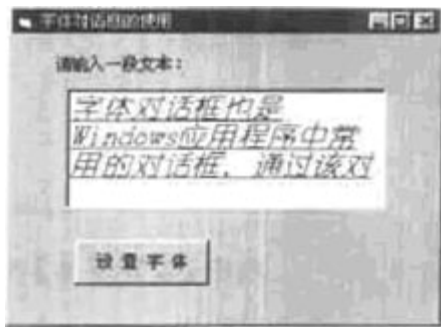


图 6.19 设置字体后的情形

VB6.0 基础教程 6.3 自定义对话框

除了预定义对话框和通用对话框外，用户还可以根据实际需要自行定义对话框。自定义对话框实际上就是在一个窗体上放置一些控件，以构成一个用来接受用户输入的界面。这里给出一个设计自定义对话框的实例。

实例 6.7 自定义对话框。

在该实例中，用户可以通过对话框输入个人资料，并且输入的资料将显示在主窗体中。该程序用到两个窗体，其中一个窗体用作主窗体，另一个窗体用作对话框。设置主窗体为启动窗体，并且通过主窗体来调用对话框。

单击工具条中的【添加窗体】按钮，则出现如图 6.20 所示的【添加窗体】对话框，在该对话框中选择窗体类型为“对话框”，单击【打开】按钮即可为当前过程添加一个用于设计对话框的窗体。在该窗体上放置了对话框中常用的【确定】和【取消】按钮。下面就以这个窗体为基础设计自定义对话框。

在新建的对话框窗体中放置 4 个标签控件、1 个文本框控件、2 个单选项控样、1 个组合框控件和 1 个框架控件，并且在框架控件中

再放置 4 个复选框控件，如图 6.21 所示。其中各控件的属性设置如表 6.16 所示。

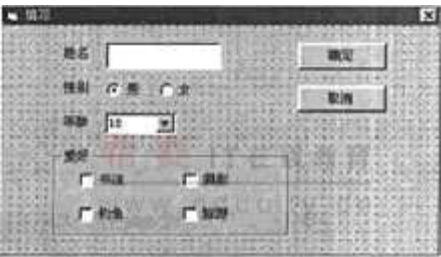


图 6.21 对话框的界面设计

表 6.16 实例 6.7 对话框中对象属性的设置

对象	属性	值
窗体	Caption	填写
标签 1	Caption	姓名
文本框	名称	TexName
标签 2	Caption	性别
单选项 1	名称	OpMan
	Caption	男
	Value	True
单选项 2	名称	OpWoMan
	Caption	女
标签 3	Caption	年龄
组合框	名称	CombAge
	List	18,19,20,21,22,23
	Text	18
框架	Caption	爱好
复选框 1	名称	H1
	Caption	书法
复选框 2	名称	H2
	Caption	摄影
复选框 3	名称	H3
	Caption	钓鱼
复选框 4	名称	H4
	Caption	旅游
按钮 1	名称	OKButton
	Caption	确定
按钮 2	名称	CancelButton
	Caption	取消

将另一个窗体作为主窗体，在该窗体中放置 12 个标签控件和一个按钮控件，其中各控件的属性设置如表 6.17 所示。

表 6.17 实例 6.7 主窗体中对象的属性设置

对象	属性	值
窗体	Caption	个人资料
标签 1	Caption	个人资料
	Font	宋体, 小三
标签 2	Caption	姓名
标签 3	名称	LabName
	Caption	置空
标签 4	Caption	性别
标签 5	名称	LabSex
	Caption	置空
标签 7	Caption	年龄
标签 7	名称	LabAge
	Caption	置空
标签 8	Caption	爱好
标签 9	名称	L1
	Alignment	2
	Caption	置空
标签 10	名称	L2
	Alignment	2
	Caption	置空
标签 11	名称	L3
	Alignment	2
	Caption	置空
标签 12	名称	L4
	Alignment	2
	Caption	置空
按钮	名称	OpenDialog
	Caption	填写资料

双击主窗体上的【填写资料】按钮，将下列代码添加到
OpenDialog_Click 事件过程中：

```
Private Sub OpenDialog_Click ( )
```

```
Dialog.show “打开自定义对话框”
```

```
End Sub
```

从代码中可以着出，单击 t 填写资料】按钮，将打开用户自定义的对话框（用作对话框的窗体的名称为 Dialog）。

双击对话框窗体中的【确定】按钮，打开【代码】窗口，将下列代码添加到 OKButton_Click 事件过程中：

```
Private Sub OKButton_Click()
```

```
显示姓名
```

If TextName.Text= “ ” Then

如果姓名为空，则弹出消息框提示用户

Msg=MsgBox(“请您输入姓名”，48，“填写”)

Exit Sub “退出事件过程

Else

MainForm.LabName.Caption TextName.Text

End If

判断性别

If OpMan = True Then

MainForm.LabSex.Caption= “男”

Else

MainForm.LabSex.Caption= “女”

End If

显示年龄

MainForm.LabAge.Caption=CombAge.Text

判断语句的爱好

It h1.Value=1 Then

MainForm.L1.Caption= “书法”

Else

MainForm.L1.Caption= “//”

End If

If h2.Value=1 Then

```
MainForm.L2.Caption= “摄影”  
  
Else  
  
MainForm.L2.Caption= “//”  
  
End If  
  
If h3.Value=1 Then  
  
MainForm.L3.Caption= “钓鱼”  
  
Else  
  
MainForm.L3.Caption= “//”  
  
End If  
  
If h4.Value=1 Then  
  
MainForm.L4.Caption= “旅游”  
  
Else  
  
MainForm.L4.Caption= “//”  
  
End If  
  
Unload Dialog “卸载对话框”  
  
End Sub
```

在该段程序中，首先使用 If 语句判断用户是否输入了姓名，如果没有，则弹出消息框提示用户。用户确定后，接着执行下面的代码 --Exit Sub 语句，退出事件过程，返回到对话框，用户可以重新输入姓名。如果没有该语句，则会接着执行该 If 语句块之外的其他语句，一直会执行到 Unload Dialog 语句关闭对话框。

在判断用户爱好的程序段中，也使用了 If 语句，如果用户选中了某爱好（复选框的 Value 属性为 1），则在主窗口的相应标签中显示该爱好；如果没有选中某爱好（复选框的 Value 属性为 0），则在主窗口的相应标签中显示“//”符号。

在程序的最后，使用 Unload Dialog(对话框窗体的名称为 Dialog) 语句关闭对话框，返回到主窗体。如果直接单击对话框中的【取消】按钮，也应该关闭对话框，而不影响到主窗体。因此，将 Unload Dialog 语句也添加到 Cancelbutton_Click 事件过程中：

```
Private Sub Cancebutton_Click ()  
Unload Dialog  
End Sub
```

通过工程的【属性】对话框将启动对象设置为 MainForm，运行该程序，则主窗体显示在屏幕上，单击【填写资料】按钮，则弹出自定义对话框。

填写完整各选项后，单击【确定】按钮，则对话框被关闭，用户输入的个人资料显示在主窗体中。如果用户输入的姓名为“郝云”，性别为“男”，年龄为“23”，爱好为“书法”、“摄影”和“旅游”，单击【确定】按钮后，显示个人资料的主窗体。

面第七章 菜单的设计与多文档界

VB6.0 基础教程 7.1 菜单简介

大多数 Windows 应用程序都有一个菜单栏，它总是处在窗体标题栏的下面，并包含一个或多个菜单标题。单击每个菜单标题都会弹出一个下拉菜单，在下拉菜单中包含有菜单项、分隔条和子菜单标题。

有的菜单项可以直接执行，有的菜单项执行时则会弹出一个对话框。所有的 Windows 应用程序都遵循以下 3 个约定：

凡是菜单名称后有一个省略号的，均表示在单击该选项后会弹出一个相应的对话框，在用户作出相应的回答后，该项功能就以用户所给予的信息去执行。例如，单击【打开】选项，则弹出【打开】对话框，用户可从中选择要打开的文件。

凡是菜单名称后有一个小三角的，则表示它是一个子菜单标题，子菜单标题并不能直接执行，仅仅扮演一个“容器”的角色。当鼠标指针移动到子菜单标题上时，会自动弹出子菜单。例如，将指针移动到【发送】选项，就会弹出子菜单。

菜单名称后不包含 L 述两种符号者，表明该菜单项所代表的命令可直接执行。例如，单击【关闭】选项，则将关闭当前打开的文档。

另外，有的菜单项名称后还显示相应的键盘访问键和快捷键。访问键允许同时按下 Alt 键和一个指定字符来打开一个菜单。一旦菜单打开，通过按下访问键即可选取菜单项。在菜单项的标题中，访问键表现为一个带下划线的字母，如【打开】命令的访问键为 O。当打开【文件】菜单后，按下 O 键即可执行【打开】命令。快捷键出现在相应菜单项的右边，例如，【打开】命令的快捷键是 Ctrl+O，无论【文件】菜单是否打开，只要按下 Ctrl+O 组合键，即可执行【打开】命令。

由于所有 Windows 应用程序都遵循上述约定，因此，在创建菜单时，也应该遵循这些约定。例如，如果某菜单项的执行结果是弹出一个对话框，则应该在该菜单项后加上省略符 (...).此外，要使应用程序简单好用，还应该将菜单项按其功能分组。例如，与文件有关的命令【新建】、【打开】和【另存为】都列入了【文件】菜单。

同一菜单中不同类型的选项之间还使用分隔条分隔开来。分隔条作为菜单项间的一个水平行显示在菜单上。在包含较多菜单项的菜单上，经常使用分隔条将各项划分成一些逻辑组。【文件】菜单，使用分隔条将其菜单项分成 6 组。

VB6.0 基础教程 7.2 菜单编辑器简介

菜单编辑器是 VB 提供的一个用于设计菜单的工具，它使看似复杂的菜单创建变得非常简单。使用菜单编辑器可以创建出新的菜单或编辑已有的菜单。打开【工具】菜单，执行【菜单编辑器】命令，将出现如图 7.3 所示的【菜单编辑器】

对话框。也可以通过单击工具栏上的【菜单编辑器】按钮来打开该对话框。其中各主要选项的含义如下：

标题：该文本框用来输入菜单名，这些名字出现在菜单栏或菜单之中。如果想在菜单中建立分隔条，则应在该文本框中输入一个连字符“—”.为了能够通过键盘访问菜单项，可在一个字母前插入&符号。例如，“新建 (&N)”.在运行时，该字母带有下划线（&符号是不可见的）。如果要在菜单中显示&符号，则应在标题中连续输入两个&符号。

名称：该文本框用来输入菜单名称。在代码中就是以该名称来访问菜单项的，它不会出现在菜单中，这与其他控件的名称是一样的。

索引：可指定一个数字值来确定控件在控件数组中的位置。该位置与控件的屏幕位置无关。

快捷键：可在该列表框中为命令选择快捷键。

帮助上下文 ID：允许为 context 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：该列表框中共有四个可选项，它们决定是否及如何在窗体中显示菜单。



图 7.3 【菜单编辑器】对话框

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，而如果希望该项失效并以浅灰色显示出来，则可取消对该复选框的选中。

可见：该选项决定是否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。

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。

左箭头：每次单击都把选定的菜单向上移一个等级。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：将选定项移动到下一行。

插入：在列表框的当前选定项上方插入一行。

VB6.0 基础教程 7.3.1 建立菜单

本节以一个实例来介绍使用菜单编辑器创建菜单的具体过程。

实例 7.1 创建菜单。

在本例中，为窗体创建一个只有两个菜单的菜单栏，一个是【文件】菜单，另一个是【编辑】菜单。其中【文件】菜单包含 3 个菜单项，分别是【新建】、【关闭】和【退出】，并且在【关闭】与【退出】之间有一个分隔条。【编辑】菜单也包含 3 个选项，分别是【颜色】、【黑体】和【字号】。并且【字号】是个子菜单标题，其子菜单中又包含 3 个菜单项。

使用菜单编辑器创建该菜单栏的步骤如下：

(1) 单击工具栏上的【菜单编辑器】按钮打开【菜单编辑器】对话框。确保【窗体】窗口为当前活动窗口，否则【菜单编辑器】按钮无效。如果工程中包含多个窗体，则为当前活动的窗体创建菜单。

(2) 在【标题】文本框中，输入“文件(&F)”。其中 F 被设置为该菜单项的访问键，在菜单中，这一字符会被自动加上一条下划线。

(3) 在【名称】文本框中输入“MenFile”，其他各选项使用默认设置，如图 7.6 所示，可见，在菜单控件列表框中显示出了刚刚创建的【文件】菜单控件。

(4) 单击【下一个】按钮，则菜单控件列表框中的光标条移动到了下一行。对应的【标题】文本框与【名称】文本框为空的，可从中输入另一个菜单控件。

(5) 在【标题】文本框中输入“新建(&N)”，再在【名称】文本框中输入“MenNew”。并在【快捷键】列表框中选择快捷键为 Ctrl+N。则“新建(&N)”与“文件(&F)”并排显示在菜单控件列表框中。

注意：快捷键将自动出现在菜单上，因此，不需要在菜单编辑器的【标题】文本框输入 Ctrl+N。

(6) 单击右箭头按钮，则菜单控件【新建】向右缩进了一段距离，并且在其前加入了四个点。这表明它成为【文件】菜单中的一个选项。四个点表示一个内缩符号，菜单编辑器就是通过内缩来判断菜单的层次的。

(7) 单击【下一个】按钮。内缩符号仍然存在，表明所创建的菜单控件仍然是【文件】菜单中的选项。依次为【文件】菜单创建【关闭】、分隔条和【退出】3个选项。

这样，文件菜单就创建完毕，下面开始创建【编辑】菜单。

8) 单击【下一个】按钮，则菜单控件列表中的光标条向下移动一格。由于【编辑】菜单是一个独立的菜单标题，而不是【文件】菜单中的一个选项，因此，单击【左箭头】按钮，取消内缩。

(9) 在【标题】文本框，输入“编辑 (&E)”，在【名称】文本框中输入“MenEdit”。

(10) 与创建【文件】菜单中各选项的方法一样，为【编辑】菜单创建3个选项。

(11) 再为【字号】选项创建子菜单。子菜单的创建与为菜单创建菜单项的方法相同，只要子菜单中各选项相对于子菜单标题内缩一个内缩符号就可以了。【字号】子菜单中各选项的属性设置。

将【字体】子菜单中几个选项的名称设置为一样的，并且为它们指定了不同的索引号，这其实就是将这几个选项创建成了一个控件数组。也可以随意指定它们的名称，而不创建成控件数组。

(12) 到此，就为窗体创建了一个包含两个菜单的菜单栏。单击【确定】按钮，关闭【菜单编辑器】对话框，创建的菜单标题将显示在窗体上。在设计时，单击一个菜单标题可在其下拉菜单中显示所有选项。

从以上的菜单创建过程中可以看出，菜单控件在菜单控件列表框中的位置决定了该控件是菜单标题、菜单项、子菜单标题，还是子菜单项：

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。

列表框中被缩进过的菜单控件，当单击其前导的菜单标题时才会显示在该菜单上。

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。在子菜单标题以下缩进的各个菜单控件，就成为该子菜单的菜单项。

在菜单创建完毕后，用户可以随时打开【菜单编辑器】对话框来增加或修改菜单控件。在【菜单编辑器】对话框的菜单控件列表框中列出了当前窗体的所有菜单控件。使用鼠标单击某菜单控件使之以高亮度显示，即可修改它的标题、名称以及快捷键等属性。也可使用左箭头按钮或右箭头按钮来调整它的类型。使用上箭头或下箭头可调整它的位置。使用【插入】按钮则可以在菜单中添加新的菜单控件。

除了可以在【菜单编辑器】对话框中设置菜单控件的属性外，也可以像设置其他控件属性一样，通过【属性】窗口来设置菜单控件的属性。单击[属性]窗口上方的对象列表框，在其下拉列表中包含有当前窗体的所有菜单控件。从中选择要设置属性的菜单控件。在【属性】窗口中列出的属性与【菜单编辑器】对话框中的选项是对应的，如Caption属性对应【标题】选项，用户也可以在此设置菜单的标题。

VB6.0 基础教程 7.3.2 编写代码

在为窗体创建了菜单后，运行程序时可以发现，虽然单击菜单标题会弹出菜单列表，将鼠标指针指向子菜单标题时也会出现子菜单，但单击菜单项时却没有任何反应。这是因为还未为它们编写代码，还未赋予它们灵魂。

每个菜单项都是一个菜单控件，菜单控件只能响应 Click 事件。因此，为菜单编写代码就是编写它们的 Click 事件过程。这里为上一节所创建的菜单编写代码，使之成为一个完整的应用程序。

注意：虽然分隔条是当作菜单控件来创建的，它仍却不能响应 Click 事件，而且也不能被选取。

实例 7.2 编写代码。

在该程序中，为实例 7.1 中所创建的菜单编写代码，使之能够执行与其标题相对应的操作。例如，执行【文件】菜单中的【退出】命令，则退出应用程序。

在实例 7.1 中所创建的包含菜单的窗体上放置一个文本框控件和一个通用对话框控件，如图 7.15 所示。其中各菜单控件的属性设置见实例 7.1，其他各对象的属性设置如表 7.4 所示。

表 7.4 实例 7.2 中各对象的属性设置

对象	属性	值
窗体	Caption	菜单应用程序
文本框	名称	Text1
	Text	空白
	MultiLine	True
	ScrollBars	2
	Visible	False
通用对话框控件	名称	DiaColor

打开窗体上的【文件】菜单，弹出该菜单的下拉菜单，单击【新建】选项，打开【代码】窗口，如图 7.16 所示。可见，该菜单的 Click

事件过程的框架自动出现在代码编辑区中为【新建】选项的 Click 事件过程添加代码，如下所示：



图 7.15 实例 7.2 的窗体设计



图 7.16 单击【新建】选项打开的【代码】窗体

```
Private Sub MenNew_Click()  
    Text1.Text=""  
    Text1.Visible=True  
End Sub
```

在设计阶段，在【属性】窗口中将文本框的 Visible 属性设置为 False,因此，在程序运行时，窗体上的文本框是不可见的。执行【新建】命令，则首先将文本框清空，然后使其可见。

编写其他。菜单项的 Click 事件过程如下：

```
Private Sub MenClose_Click()  
    Text1.Visible=False  
End Sub  
  
Private Sub MenExit_Click()  
End  
End Sub
```

```
Private Sub MenColor_Click()

DiaColor.Action=3

Text1 .ForeColor, DiaColor.Color

End Sub

Private Sub MenFont_Click()

Text1 .FontBold=True

End Sub

Private Sub MenSize_Click(Index As Integer)

Select Case Index

Case 0

Text1.FontSize=14

Case 1

Text1.FontSize=18

Case 2

Text1.FontSize=20

End Select

End Sub
```

运行该程序，则出现如图 7.17 所示的窗体，打开【文件】菜单，执行其中的【新建】命令，在窗体上出现一个文本框。在该文本框中输入内容。通过【编辑】菜单可以设置文本的颜色、字体以及字号。执行【文件】菜单中的【关闭】命令，则文本框消失，窗体恢复的情形。执行【退出】命令，则退出应用程序。

也可以使用显示在菜单上的快捷键或访问键来操作菜单，例如，按下 **Ctrl+N** 组合键，则文本框出现；按下 **Ctrl+Q** 组合键，则退出应用程序。

VB6.0 基础教程 7.4.1 有效性控制

与按钮的有效性一样，菜单项也有“有效”与“无效”两种状态。在“无效”状态时，菜单项以浅灰色显示，不能响应用户的任何操作。一些菜单项只有在满足一定条件时才有效。例如 VB **【编辑】** 菜单中的 **【粘贴】** 命令，只有当当前剪贴板中有内容时，该选项才有效，否则无效，并以浅灰色显示。

菜单控件也有 **Enabled** 属性，该属性用来设置菜单在程序运行时是否有效。在设计阶段，该属性既可以在 **【菜单编辑器】** 对话框中设置，也可以在 **【属性】** 窗口中设置。在默认情况下，菜单控件的 **Enabled** 属性的值为 **True**。在 **【菜单编辑器】** 对话框中的菜单控件列表选中某菜单控件，取消对 **【有效】** 复选框的选中，或在菜单控件的 **【属性】** 窗口中设置 **Enabled** 属性的值为 **False**，即可将该菜单项设置为无效。

也可以在程序运行时通过代码来设置菜单项的有效性。

实例 7.3 菜单项的有效性。

本例将对实例 7.2 中的代码进行一点修改，使程序在未新建文档时 **【关闭】** 菜单项无效，**【新建】** 菜单项有效。在新建文档后，**【关闭】** 菜单项变为有效，而 **【新建】** 菜单项变为无效。

单击 **【属性】** 窗口上方的对象列表框，从中选择菜单控件 **MenClose**，在属性列表中将 **Enable** 属性的值设置为 **False**。

打开【代码】窗口，将【新建】与【关闭】菜单项的 Click 事件过程修改如下：

```
Private Sub MenClose_Click()  
  
Text1.Visible=False  
  
MenNew.Enabled=Ture  
  
MenClose.Enabled=False  
  
End Sub  
  
Private Sub MenClose_Click()  
  
Text1.Text=""  
  
Text1.Visible=Ture  
  
MenNew.Enabled=False  
  
MenClose.Enabled=Ture  
  
End Sub
```

运行修改后的程序，程序启动后，打开【文件】菜单，可见【关闭】菜单项无效，如图 7.19 所示。单击【新建】菜单项，则【关闭】菜单项变为有效，而【新建】菜单项变为无效，如图 7.20 所示。

VB6.0 基础教程 7.4.2 菜单项标记

有些菜单项表示的是一种开关状态，这些命令其实就是在两种不同的状态之间切换。这些菜单项就像一个开关，当处于“开”状态时，菜单项显示一个“√”标记，当处于“关”状态时，不显示任何标记。如图 7.21 所示的是 VB【视图】菜单中的【工具栏】子菜单，【标准】

选项的左边显示一个“√”标记，表明当前界面上显示有标准工具栏。

单击该选项，则“√”标记消失，同时界面上的工具栏也被关闭。



图 7.21 菜单项标记

还有一种情况常常使用到菜单项标记。当菜单栏中有多个并列的选项时，菜单项标记用来表明用户所选的是哪一个选项。例如，在 VB 的【窗口】菜单中，显示有“√”标记的窗口为当前活动的窗口。

菜单控件的 `Checked` 属性用来决定是否在菜单项上显示“√”标记。该属性的默认值为 `False`，却不显示“√”标记。如果设置它的值为 `True`，则显示“√”标记。【菜单编辑器】对话框中的【复选】选项对应的是 `Checked` 属性，选中该选项与在【属性】窗口中设置 `Checked` 属性的值为 `True` 的效果是一样的。

在实例 7.2 中，当单击【黑体】菜单项后将文本的字体变为黑体，无法将其恢复为宋体。此外，在【字号】子菜单中选择字号时，子菜单项不出现任何标记，用户难以知道当前文本的字号是多少。如果使用菜单项标记，则可以解决这些问题。

实例 7.4 菜单项标记。

通过对实例 7.2 程序的修改，使用户在单击【粗体】选项后，该选项的左边出现一个“√”标记，表明当前文本以粗体显示。再次单击【粗体】选项，则“√”标记消失，且文本恢复以标准显示。在【字号】子菜单中，用户所选的字号前会也出现一个“√”标记。

打开【代码]窗口，修改 MenFont_Click 与 MenSize_Click 事件过程如下：

```
Private Sub MenFont_Click()  
    If MenFont.Checked=False Then  
        Text1.FontBold=True  
        MenFont.Checked=Ture  
    Else  
        Text1.FontBold=False  
        MenFont..Checked=False  
  
    End If  
End Sub  
  
Private Sub MenSize_Click(Index As Ineger)  
    Selec Case Index  
    Case 0  
        Text1.FontSize=14  
        MenSize(0).Checked=True  
        MenSize(1).Checked=False  
        MenSize(2).Checked=False  
    Case 1  
        Text1.FontSize=18  
        MenSize(1).Checked=True
```

```
MenSize(0).Checked=False  
  
MenSize(2).Checked=False  
  
Case 2  
  
Text1.FontSize=20  
  
MenSize(2).Checked=True  
  
MenSize(0).Checked=False  
  
MenSize(1).Checked=False  
  
End Select  
  
End Sub
```

在 `MenFont_Click` 事件过程中, 使用了 `If` 语句来判断菜单项当前的状态, 如果其值为 `False`, 则将文本变为粗体, 并设置其值为 `True`; 如果其值为 `True`, 则取消文本的粗体效果, 并设置其值为 `False`. 在 `MenSize_Click` 事件过程中, 每个菜单项响应 `Click` 事件后都将执行三步操作: 首先设置文本的字号; 其次是将它自身 `Checked` 属性的值设置为 `True`, 即在菜单项上显示“√”标记; 最后是将其他菜单控件的 `Checked` 属性的值设置为 `False`, 即取消其他菜单项的“√”标记。

运行修改后的程序, 执行【文件】菜单中的【新建】命令, 然后在文本框中输入一段文本, 执行【编辑】菜单中的【粗体】命令, 则该命令的左边出现了一个“√”标记, 同时, 文本字体变为粗体。再次单击【粗体】选项, 则“√”标记消失, 文本恢复标准显示。单击【字号】子菜单中的【三号】选项, 则该选项的左边出现了一个“√”标记, 并且文本字号变为三号。



图 7.19 未新建文档



图 7.20 新建文档后

VB6.0 基础教程 7.4.3 菜单项的隐藏与显示

在一些应用程序中有些菜单项是隐藏的，只有当满足一定条件时，这些菜单项才会显示出来。

菜单控件的 **Visible** 属性用来决定菜单项是否显示。该属性的默认值为 **True** 即菜单项总是显示出来的。如果设置它的值为 **False**, 则菜单项将不显示出来。【菜单编辑器】对话框中的【可见】选项对应的是 **Visible** 属性，取消对该选项的选中与在【属性】窗口中设置 **Visible** 属性的值为 **False** 的效果是一样的 4。

实例 7.5 菜单项的隐藏与显示。

修改实例 7.4，使得只有在字体为粗体时，【字号】子菜单才显示出来。

单击【属性】窗口上方的对象列表框，从中选择菜单控件 **MenMsize**，在属性列表中将 **Visible** 属性的值设置为 **False**。打开【代码】窗口，修改 **MenFont_Click** 事件过程如下：

```
Private Sub MenFont_Click()

    If MenFont.Checked=False Then
```

```

Text1.FontBold=True

MenFont.Checked=True

MenMsize.Visible=True

Else

Text1.1.FontBold=False

MenFont.Checked=False

MenMsize.Visible=False

End If

End Sub

```

运行修改后的程序，单击【编辑】菜单，可见在下拉菜单中没有出现【字号】子菜单，如图 7.23 所示。单击【黑体 1 选项，则【字号】子菜单又显示出来，如图 7.24 所示。



图 7.23 不显示【字号】子菜单



图 7.24 显示【字号】子菜单

VB6.0 基础教程 7.5 快捷菜单

在 Windows 应用程序中，除了菜单栏以外，还存在着另外一种形式菜单-快捷菜单。快捷菜单是一种独立于菜单栏而显示在窗体上的浮动菜单。在不同的对象上单击鼠标右键，弹出的快捷菜单中的命令也是不同的。快捷菜单总是提供与当前指针所指对象相关的操作命令。例如，将鼠标指针移动到 VB 的工具箱上，单击右键，则弹出快

捷菜单，通过该快捷菜单中的命令，用户可以方便地执行对工具箱的有关操作。

为应用程序建立快捷菜单，会使程序的操作更方便快捷。快捷菜单也是通过菜单编辑器创建的，并且创建的方法与创建普通菜单相同。下面给出一个创建快捷菜单的实例。

实例 7.6 创建快捷菜单。

该程序的目的是，在窗体上单击右键，则弹出一个快捷菜单“背景色”，通过它可以设置窗体的背景颜色，如图 7.27 所示。在【确定】按钮上单击右键，也可以弹出一个快捷菜单，通过它可以设置按钮的颜色以及字体，如图 7.27 所示。



图 7.26 窗体的快捷菜单



图 7.27 按钮的快捷菜单

首先，使用菜单编辑器创建这两个快捷菜单。按照创建普通菜单的方法创建两个菜单，标题分别是“按钮”和“窗体。其中各菜单控件的属性设置如表 7.5 所示。

表 7.5 实例 7.6 中菜单控件的属性设置

菜单控件	标题	名称	索引	可见	复选
菜单标题	按钮	PopCom		不选中	不选中
菜单项	颜色...	PopCc		选中	不选中
菜单项	-	PopBar		选中	不选中
菜单项	宋体	PopFont	0	选中	选中
菜单项	隶书	PopFont	1	选中	不选中
菜单项	幼圆	PopFont	2	选中	不选中
菜单标题	窗体	PopForm		不选中	不选中
菜单项	背景色...	PopFc		选中	不选中

这不再逐步介绍菜单的创建方法，只指出几个需要注意的地方。

(1) 在快捷菜单中并不显示菜单标题，因此，菜单标题可以自由设定。

(2) 快捷菜单不出现在菜单栏中，因此，需要将菜单标题的 Visible（可见）属性的值设置为 False（取消对【可见】复选框的选中）。

接下来设计一个窗体，在该窗体中使用已经创建的快捷菜单。在窗体上放置一个按钮控件和一个通用对话框控件。其中各对象的属性设置如表 7.6 所示。

表 7.6 实例 7.6 中各对象的属性设置		
对象	属性	值
窗体	名称	Form1
	Caption	快捷菜单
按钮	名称	Com
	Caption	确定
	Font	宋体，四号
	Style	1
通用对话框	名称	DlgColor

打开【代码】窗口，编写按钮的 MouseUp 事件过程如下：

```
Private Sub Com_MouseUp(Button As Integer,Shift
As Integer,X As Single,Y As Single)

    If Button=2 Then

        PopupMenu Popcom

    End If

End Sub
```

由于快捷菜单是在单击右键时弹出的，因此，在该段代码中使用了一个 If 语句来判断用户所单击的键，如果单击的是右键（Button 参数的值为 2），则使用窗体的 PopupMenu 方法显示快捷菜单。

PopupMenu 方法的一般格式是：

[窗体名].PopupMenu 菜单名, Flags, x, y, BoldCommand.

PopupMenu 方法有 5 个参数，其中参数“菜单名”是必须的，而其他参数是可选的。“菜单名”是指菜单标题的名称。例如，已创建的两个快捷菜单的菜单名分别是 PopCom 与 PopForm.其中参数 x 和 y 分别用来指定快捷菜单出现位置的横坐标与纵坐标，并且基准点由 Flags 参数指定。如果省略，则快捷菜单就显示在鼠标指针当前的位置。参数 Flags 用来定义快捷菜单的基准点与操作方法，分别如表 7.7 与表 7.8 所示。

表 7.7 Flags 参数的取值与基准点

取值	含义
0 (默认)	X 坐标是指快捷菜单的左边界
4	X 坐标是指快捷菜单的中心位置
8	X 坐标是指快捷菜单的右边界

表 7.8 Flags 参数的取值与操作方法

取值	含义
0 (默认)	通过单击鼠标左键，选择菜单项
2	通过单击鼠标左键或右键，选择菜单项

Flags 参数的取值也可以是上述两组取值的相加（每组只能取一个）。例如。Flags=6，则表明它同时具有 Flags=4 与 Flags=2 的特征。这一点与在前面学习的其他方法的 Flags 参数是相同的。

参数 BoldCommand 的作用是指定在快捷菜单中以粗体显示的菜单项的名称。一个快捷菜单中只能有一个菜单项以粗体显示。

同样，编写窗体的 MoussUp 事件过程如下：

```
Private Sub From_MouseUp(Button As Integer,Shift  
As Integer,X As Single,Y As Single)  
  
If Button=2 Then
```

```
PopupMenu PopFrom
```

```
End If
```

```
End Sub
```

在程序运行时，通过上述代码可以在窗体上显示快捷菜单，但其中的命令不起作用。还需要编写快捷菜单中菜单项的 Click 事件代码，代码如下：

```
Private Sub PopCc_Click()
```

```
    DiaColor.Action=3
```

```
    Com.BackColor=DiaColor.Color
```

```
End Sub
```

```
Private Sub PopFc_Click()
```

```
    DiaColor.Action=3
```

```
    Form1.BackColor=DiaColor.Color
```

```
End Sub
```

```
Private Sub PopFont_Click(Index As Integer)
```

```
    Select Case Index
```

```
        Case 0
```

```
            Com.FontName= “宋体”
```

```
            PopFont(0).Checked=True
```

```
            PopFont(1).Checked=False
```

```
            PopFont(2).Checked=False
```

```
        Case 1
```

```
Com.FontName= “隶书”  
  
PopFont(1).Checked=True  
  
PopFont(0).Checked=False  
  
PopFont(2).Checked=False  
  
Case 2  
  
Com.FontName= “幼圆”  
  
PopFont(2).Checked=True  
  
PopFont(1).Checked=False  
  
PopFont(0).Checked=False  
  
End Select  
  
End Sub
```

运行该程序，在窗体上单击右键，就会弹出一个快捷菜单，单击【背景色】选项，则打开【颜色】对话框，从中选择一种颜色，单击【确定】按钮。可以发现窗体的背景颜色被改变了。再将鼠标指针移动到按钮上，单击右键，也会弹出一个快捷菜单，通过该菜单可以设置按钮的颜色以及其上文本的字体。如图 7.30 所示的是使用快捷菜单将按钮上文本的字体设置为隶书。

提示：.在显示快捷菜单时，PopupMenu 方法后面的代码直到用户从菜单中选择可命令（这时，该命令的 Click 事件的代码比 PopupMenu 语句后面的代码先执行）或者取消该菜单时才能执行。

VB6.0 基础教程 7.6 多文档(MDI)界面

Windows 应用程序的用户界面主要分为两种形式：单文档界面（Single Document Interface, SDI）和多文档界面（Multiple Document Interface, MDI）。单文档界面并不是指只有一个窗体的界面，而是指应用程序的各窗体是相互独立的，它们在屏幕上独立显示、移动、最小化或最大化，与其他窗体无关。在前面创建的所有程序都是单文档界面。

多文档界面由多个窗体组成，但这些窗体不是独立的。其中有一个窗体称为 MDI 父窗体（简称为 MDI 窗体），其他窗体称为 MDI 子窗体（简称为子窗体）。子窗体的活动范围限制在 MDI 窗体中，不能将其移动到 MDI 窗体之外。可见，多文档界面与简单的多重窗体界面是不同的，后者实际是单文档界面。

绝大多数 Windows 的大型应用程序都是多文档界面。例如，Microsoft Word, Excel 以及 VB 等都是多文档界面。如图 7.31 所示的是 Word 的用户界面，在 MDI 窗体中包含三个子窗体。



图 7.31 多文档界面

VB6.0 基础教程 7.6.1 创建 MDI 界面

要创建 MDI 界面，首先需要为应用程序创建一个 MDI 窗体。单击【工程】

菜单，执行其中的【添加 MDI 窗体】命令，弹出【添加 MDI 窗体】对话框，在【新建】选项卡中选中【MDI 窗体】，单击【打开】按钮即可在当前工程中创建一个 MDI 窗体。

与普通的窗体相比，MDI 窗体具有以下特点：

在外观上：MDI 窗体的背景看起来更黑一些，并且有一个边框。

如图 7.32 所示的是 MDI 窗体与普通窗体外观的比较。



图 7.32 MDI 窗体与普通窗体外观的比较

显示在工程资源管理器中的 MDI 窗体的图标与普通窗体的图标不同，如图 7.33 所示。从图标上很容易识别窗体的类型。



图 7.33 图标显示了窗体的类型

一个应用程序只能有一个 MDI 窗体。

在 MDI 窗体上只能放置那些有 Alignment 属性的控件（如图片框、工具栏等）和具有不可见界面的控件（如计时器和通用对话框控件）。其他控件如文本框、按钮等控件不能放置在 MDI 窗体中。要将

这些控件放置在 MDI 窗体上，可以事先在 MDI 窗体中放置一个图片框控件，然后将其他控件放置在图片框中。

不能使用 `Pint` 方法在 MDI 窗体上显示文本。可以在 MDI 窗体中的图片框中使用 `Print` 方法来显示文本。

MDI 窗体的大多数属性与普通窗体的属性相同。MDI 窗体还有两个特有的属性：`AutoShowChildren` 属性和 `ScrollBars` 属性。

`AutoShowChildren` 属性决定在加载子窗体时，是否自动显示它们。如果 `AutoShowChildren` 属性的值为 `True`（默认值），子窗体一载入就显示出来。这就是说，`Load` 语句和 `show` 方法的作用是相同的。

`ScrollBars` 属性决定 MDI 窗体在必要时是否显示滚动条。当该属性的值为 `True`（默认值）时，如果一个或多个子窗体延伸到 MDI 窗体之外，MDI 窗体上就会出现滚动条。

子窗体的创建很容易。对于普通的窗体，只要将其 `MDIChild` 属性设置为 `True` 即可将其设置成为一个子窗体。因此，创建子窗体的方法是：首先创建一个普通窗体，然后将它的 `MDIChild` 属性设置为 `True` 就可以了。

细心的用户可以发现，将普通窗体设置为子窗体后，窗体在工程资源管理器中的图标改变了，它与普通窗体和 MDI 窗体的图标都不同，如图 7.35 所示。这也是将普通窗体设置为子窗体后，能在设计窗口中看到的唯一的变化。



图 7.35 不同类型窗体的图标

注意；MDIChild 属性只能在设计阶段通过【属性】窗口设计，不能在代码中设置，并且，在设置该属性前，要确保已经建立了 MDI 窗体。否则，则程序运行时会出现错误。

MDI 窗体与子窗体创建完成后，接下来的工作就是设计窗体与编写代码了。在 MDI 窗体上一般只放置菜单栏、工具栏以及任务栏。子窗体的设计则与普通窗体的设计完全相同。也可以先设计窗体，然后改变 MDIChild 属性。操作顺序不会影响窗体的行为。

VB6.0 基础教程 7.6.2 MDI 界面的特点

本节将创建一个 MDI 窗体与一个子窗体，来演示 MDI 界面的一些特点。

新建一个工程，使用【工程】菜单中的【添加 MDI 窗体】命令向工程中添加一个 MDI 窗体。将工程中已存在的普通窗体的 MDIChild 属性的值设置为 True。

运行该程序，则出现如图 7.36 所示的界面。这是由于在默认情况下启动窗体是子窗体。在加载子窗体时，MDIChild 窗体会自动加载并显示。

在【工程属性】对话框中将启动窗体设置为 MDI 窗体，运行程序，则出现如图 7.37 所示的界面。加载 MDIChild 窗体时，其子窗体

并不会自动加载。子窗体的加载与卸载等操作与普通窗体相同。为 MDI 窗体的 Load 事件编写如下代码：

```
Private Sub MDIForm_Load()  
  
Foaml.Show  
  
End Sub
```



图 7.36 子窗体为启动窗体

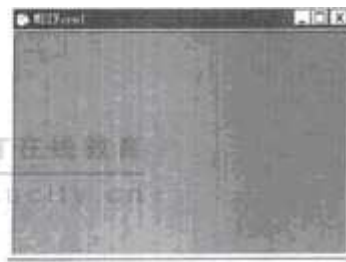


图 7.37 MDI 窗体为启动窗体

如果 MDI 窗体的 AutoShowChildren 属性值为 True，也可以使用 Load Form1 语句来显示子窗体。

再次运行程序。可以发现子窗体显示在 MDI 窗体中，如图 7.36 所示。

与卸载普通窗体一样，卸载 MDI 窗体也是使用 Unload 语句。例如，为程序添加下列代码，运行程序，双击 MDI 窗体即可卸载它。

```
Private Sub MDIForm_DblClick()  
  
Unload MDIForm1  
  
End Sub
```

子窗体最小化后将以图标的形式出现在 MDI 窗体中，而不会出现在 Windows 的任务栏中。子窗体最大化后，它的标题出现在 MDI 窗体的标题栏中。运行程序，单击子窗体的最小化按钮使其最小化，如图 7.38 所示。单击子窗体的最大化按钮使其最大化。单击 MDI 窗

体的最小化按钮使其最小化，所有子窗体都不可见，并且只有 MDI 窗体的图标出现在任务栏中。读者还可以移动子窗体，它不会移动到 MDI 窗体之外。

在 MDI 窗体与子窗体中都可以建立菜单，并且建立菜单的方法与普通窗体相同。但在菜单的显示下，MDI 应用程序还有一些自己的特色。当没有打开子窗体时，MDI 窗体上显示自己的菜单。当打开了子窗体时，则 MDI 窗体的菜单栏上显示当前活动子窗体的菜单。也就是说，子窗体的菜单取代了 MDI 窗体菜单栏中的菜单。

利用菜单编辑器为 MDI 窗体创建一个菜单。为子窗体创建 3 个菜单。运行程序，可见，子窗体中的菜单取代了 MDI 窗体本身的菜单，出现在 MDI 窗体的菜单栏中。单击子窗体的关闭按钮将其关闭，则 MDI 窗体菜单栏中显示的是其本身的菜单。

VB6.0 基础教程 7.6.3 新建子窗体

多文档应用程序中最常见的一项操作是通过新建命令建立多个子窗体。例如，执行 Word【文件】菜单下的【新建】命令，可以新建出多个子窗体。这些子窗体的外观与功能完全相同，只是在名称上使用编号区分，如“文档 1”“文档 2”等。

要实现这一功能，一种最直观的方法是：首先在工程中添加若干个窗体，并将它们设置为 MDI 窗体的子窗体。然后，在程序运行时，通过【新建】菜单来打开它们。事实上，使用这种方法实现起来是很困难的，缺乏灵活性并浪费系统资源。

在 VB 中，是通过对象变量来为 MDI 窗体生成了窗体的。

声明窗体的对象变量的格式如下：

Dim 变量 As new 窗体名

其中“窗体名”是一个已存在的子窗体的名称。在声明了对象变量后，通过语句。

变量。Show

就可以显示出一个新的窗体。

实例 7.7 新建子窗体

新建一个工程，向当前工程中添加一个 MDI 窗体，并为它创建一个【文件】菜单，其中包含【新建】与【退出】两个选项。其中工程中各对象的属性设置如表 7.9 所示。

表 7.9 实例 7.7 中对象的属性设置

对象	属性	值
MDI 窗体	名称	MDIForm1
	Caption	新建子窗体
【新建】新建菜单	名称	MenNew
【退出】菜单	名称	MenExit
子窗体	名称	FrmChild

打开【代码】窗口，编写代码如下：

```
Private Sub MenNew_Click()  
  
Dim DocForm As New FrmChild  
  
Static i As Integer  
  
DocForm.Caption="无标题"& i+1  
  
DocForm.show  
  
i=i+1  
  
End Sub
```

```
Private Sub MenExit_Click()

End

End Sub
```

在 `MenNew_Click` 事件过程中，定义了一个静态变量 `i`，通过该变量可以使子窗体的标题出现编号。

设置启动窗体为 MDI 窗体，运行程序，出现界面。执行【文件】菜单中的【新建】命令，则 MDI 窗体中会出现一个标题为“无标题 1”的子窗体。再次执行【新建】命令，则又出现一个标题为“无标题 2”的子窗体。执行 3 次【新建】命令后的结果。

子窗体 `FrmChild` 是新建窗体的模板，新创建出的子窗体将与 `FrmChild` 子窗体的外观完全相同。例如，在 `FrmChild` 窗体中放置一个文本框，则新建的子窗体中也有一个文本框，并且名称是相同的。因此，要更改子窗体的外观，只需更改模板子窗体的外观就可以了。

除此之外，模板子窗体的事件过程对新建子窗体也有效。例如，编写模板子窗体的事件过程如下：

```
Private Sub Form_Click()

Text1.Text="欢迎"

End Sub
```

运行程序，在新建的子窗体上单击，则子窗体中的文本框内会显示“欢迎”两个字。单击子窗体“无标题 2”后的效果。

MDI 窗体还有一个重要的属性-ActiveForm,使用该属性可以代替当前活动子窗体的名称。例如:

MDIForm.ActiveForm.Caption=“ABCD”

语句的功能是将当前活动子窗体的标题更改为 AECD。

使用 ActiveForm 属性可以引用当前活动子窗体的任意属性、方法或事件,而不必知道子窗体的名称。

将【文件】菜单中【退出】选项的 Click 事件过程更改为:

```
Private Sub MenExit_Click()  
    Unload MDIForm1.ActiveForm  
End Sub
```

这样,执行【退出】命令的效果是将当前活动的子窗体卸载。

VB6.0 基础教程 7.6.4 创建【窗口】菜单

大多数应用程序都有一个【窗口】菜单,在该菜单中包含有层叠、平铺和排列图标等设置子窗体排列方式的命令,还显示有所有已打开的子窗体列表,从中可方便地选择要激活的子窗体。如图 7.50 所示的是 VB 的【窗口】菜单。



图 7.50 VB 的【窗口】菜单

如果要在 MDI 窗体的某个菜单中显示了窗体列表，可以在【菜单编辑器】对话框中选中该菜单的【显示窗体列表】复选框，或者在【属性】窗口中将 WindowsList 属性的值设置为 True.不需要用户编写任何代码。

MDI 窗体的 Arrange 方法可以使子窗体按一定的规律排列。

Arrange 方法的一般格式为：

MDI 窗体名。Arrange 参数。

其中“参数”是一个整数，表示所使用的排列方式，系统共提供了四种排列方式，如表 7.10 所示。

表 7.10 Arrange 方法的参数取值

符号常数	对应值	含义
VbCascade	0	使各子窗体层叠排列
VbTileHorizontal	1	使各子窗体水平平铺
VbTileVertical	2	使各子窗体垂直平铺
VbArrangeIcons	3	当各子窗体最小化后，可使图标重新排列

实例 7.8 创建【窗口】菜单。

本例是为实例 7.7 所创建的 MDI 窗体添加一个【窗口】菜单。

在【菜单编辑器】对话框中，各菜单控件的属性设置如表 7.11 所示。

表 7.11 窗口菜单控件的属性设置

对象	属性	值
菜单标题	标题	窗口
	名称	MenWin
	显示窗体列表	选中
菜单项	标题	层叠
	名称	MenW
	索引	0
菜单项	标题	水平平铺
	名称	MenW
	索引	1
菜单项	标题	垂直平铺
	名称	MenW
	索引	2
菜单项	标题	排列图标
	名称	MenW
	索引	3

在【代码】窗口中编写菜单项的 Click 事件过程如下：

```
Private Sub MneW_Click(Index As Integer)

    Select Case Index

        Case 0

            MDIForm1.Arrange 0

        Case 1

            MDIForm1.Arrange 1

        Case 2

            MDIForm1.Arrange 2

        Case 3

            MDIForm1.Arrange 3

    End Select

End Sub
```

运行程序，执行【新键】命令新建几个子窗体。新建的子窗体在默认情况下以层叠方式排列。打开【窗口】菜单，可以发现在菜单列表中显示有当前所有子窗体的列表。

单击【水平平铺】选项，子窗体的排列如图 7.53 所示。单击【垂直平铺】选项，子窗体的排列如图 7.54 所示。

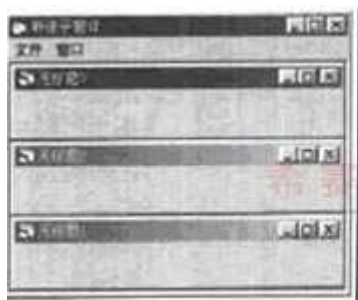


图 7.53 水平平铺



图 7.54 垂直平铺

单击子窗体的最小化按钮使其最小化，则子窗体图标有序地排列在 MDI 窗体的底部，如图 7.55 所示。使用鼠标随意拖动子窗体图标，改变它们的位置，如图 7.56 所示。单击【排列图标】选项，则子窗体图标恢复如图 7.55 所示的有序排列。



图 7.55 排列图标

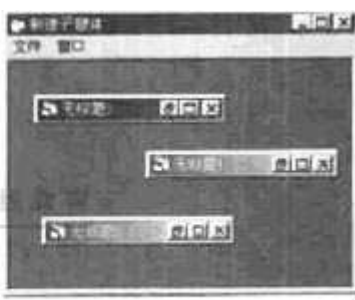


图 7.56 改变图标的位置

第八章 程序调试

VB6.0 基础教程 8.1 程序错误分类

VB 中常见的程序错误可分为编译错误、运行错误和逻辑错误 3 类。

1. 编译错误

编译错误也称为语法错误，在编写程序时，如果语句不符合 Visual Basic 的语法规则，就会产生这类错误。例如，输入了不正确的关键字、遗漏了某个必需的标点符号、缺少表达式、类型不匹配或者应该配对的语句没有配对等，都会产生编译错误。

在编写代码或运行程序时，很容易检查出这类错误。在编写代码时，VB 会自动对程序进行语法检查，某些类型的语法错误能够被即时检查出来，并且会弹出一个出错消息框，出错的那一行以高亮度显示。例如，当输入“I=”后没有接着输入表达式，而是切换到其他行，则会弹出如图 8.1 所示的消息框。

还有一些类型的语法错误，在编写代码时 VB 检查不出来，例如，If 语句后没有对应的 End If 语句、输入了错误的属性名等。在运行程序时，VB 将弹出错误消息框，提示用户错误所在，如图 8.2 所示。

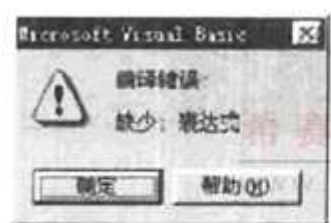


图 8.1 编写代码时弹出的消息框



图 8.2 运行程序时弹出消息框

2.运行错误

运行错误是程序运行时出现的错误。运行时，如果一个语句无法正常完成自己的功能时，就会出现这类错误。例如，执行除法操作时除数为 0，或加载一个图片时文件不存在，都将产生错误。出现运行错误时也会弹出一个消息框，如图 8.3 所示的是除数为 0 时弹出的消息框。



图 8.3 运行错误消息框

运行错误消息框的第一行显示的是运行错误代号，每个运行错误都对应一个代号。第二行显示的是错误的说明。

单击【结束】按钮，则结束程序的运行，返回到设计模式；单击【调试】按钮，则切换到中断模式，显示出【代码】窗口，并且出错的语句以高亮度显示，此时可以编辑代码。单击【帮助】按钮，则打开 VB 的帮助窗口。其中提供了错误说明、错误代号、引发错误的原因以及解决错误的办法等信息。

3.逻辑错误

有的时候，应用程序的代码完全符合语法要求，运行时也不出现任何错误，但却未出现期望的结果，这表明程序中存在逻辑错误。这类错误是因为代码中存在逻辑上的缺陷而引起的，例如，设置的选择条件不合适、循环次数不当等。逻辑错误最隐蔽，较难发现和排除，程序员的语言功底和编程经验在排除这类错误时很重要。

程序调试就是寻找和排除错误的过程，VB 提供了一套交互式的调试工具，程序开发人员可以借助它们来查找出逻辑错误。

VB6.0 基础教程 8.2 调试工具栏

为了调试程序的方便，用户可以使用 VB 的调试工具栏。在默认情况下，VB 界面上不显示调试工具栏。打开【视图】菜单，指向【工具栏】选项，则弹出【工具栏】子菜单，执行其中的【调试】命令即可打开调试工具栏。其中各按钮的功能如表 8.1 所示。

表 8.1 调试工具栏中各按钮的功能

图标	名称	功能	快捷键
	启动/继续	运行应用程序，或从中断模式进入运行状态	F5
	中断	中断当前程序的运行，进入中断模式	Ctrl+Break
	结束	终止当前程序的运行	
	切换断点	在程序代码行上设置断点或取消断点	F9
	逐语句	单击一次，执行一行代码，并进入过程的内部	F8
	逐过程	与逐语句按钮类似，但视过程和函数调用为一条语句，不进入其内部	Shift+F8
	跳出	在逐语句执行代码执行到调用的过程中时，单击该按钮将从被调用的过程中跳出	Ctrl+ Shift+F8
	本地窗口	打开本地窗口	Ctrl+G
	立即窗口	打开立即窗口	
	监视窗口	打开监视窗口	
	快速监视	添加监视表达式	Shift+F9
	调用堆栈	在中断模式下，单击该按钮将打开【调用堆栈】对话框	Ctrl+L

VB6.0 基础教程 8.3 设置断点

断点是告诉 VB 挂起程序执行的一个标记，当程序执行到断点处即暂停程序的执行，进入中断模式，此时可以在【代码】窗口中查看程序内变量、属性的值。在代码中设置断点是常用的一种调试方法。

在 VB 中，断点的设置有两种办法：

(1) 将光标放置在需要设置断点的地方，执行【调试】菜单中的【切换断点】命令或单击调试工具栏中的 1 切换断点 1 按钮，即可在该行语句上设置一个断点。

(2) 设置断点更简便的办法是，直接在要设置断点的行的左边单击鼠标。设置了断点的行将以粗体显示，并且在该行左边显示一个黑色的圆点，作为断点的标记。在代码中可以设置多个断点。

设置完断点后，运行程序，运行到断点处，程序就暂停下来，进入中断模式。这时断点处语句以黄色背景显示，左边还显示一个黄色

小箭头，表示这条语句等待运行。把鼠标光标移到各变量处，会显示变量的当前值，如图 8.8 所示。



图 8.8 在中断模式下显示变量的值

只要再对设置有断点的行执行一次设置断点的操作，即可清除该行的断点。

在需要设置断点的代码行前面添加一个 **Stop** 语句，也能起到断点的作用，在程序运行遇到 **Stop** 语句时，就会暂停下来。使用 **Step** 语句比设置断点更灵活，例如，可以让某个循环在循环指定次数后停止执行，进入到中断模式。

VB6.0 基础教程 8.4 跟踪程序的运行

查找程序中的错误所在并不那么容易，有时需要一条语句一条语句地执行或者反复执行某段代码来检查错误所在，这些方法被称为跟踪程序的运行。

1. “逐语句”跟踪

“逐语句”执行代码就是一条语句一条语句地执行代码，每执行一条语句后，就暂停下来，为程序调试者提供分析判断的机会。

进入“逐语句”方式跟踪程序执行的具体办法是执行【调试】菜单中的【逐语句】命令，或单击调试工具栏里的【逐语句】按钮。不过

最常用的方法还是使用快捷键 F8,每按一次 F8 键,程序就执行一条语句,调试者可以观察代码的流程和语句的执行情况。

2.“逐过程”跟踪

如果要调试的程序调用别的过程,而被调用过程已经经过了调试,确保能正确执行,那么在调试这个程序时,若使用“逐语句”去跟踪就会在调用时到被调用过程里去一句句地执行,这显然没有必要。这时最好的办法是采用“逐过程”跟踪,把被调用过程当作一条语句处理。如果在事件过程中没有调用其他过程,则“逐过程”跟踪与“逐语句”跟踪相同。

进入“逐过程”方式跟踪程序执行的具体办法是执行【调试】菜单中的【逐过程】命令,或单击调试工具栏里的【逐过程】按钮,也可以使用快捷键 Shift+F8。

当使用逐语句跟踪进入被调用过程后。如果从开始的几条语句就断定出该过程没有问题,可以执行【调试】菜单中的【跳出】命令,从当前的过程中提前跳出,去执行过程调用者的下一条语句。单击调试工具栏中的【跳出】按钮或使用快捷键 Ctrl+Shift+F8 也可以跳出被调用的过程。

3.运行到光标处

在对程序进行跟踪时,总是要一条语句一条语句地执行,这样有时显得较繁琐。对于不感兴趣的代码部分可以略过,方法是首先将光标插入到需要停止运行的某行语句中,然后执行【调试】菜单中的【运行到光标处】命令,则程序运行到光标处就会中断运行。这时,调试

者可以逐语句或逐过程执行后面的代码。【运行到光标处】命令的快捷键是 Ctrl+F8。

4. 设置下一条语句

在前面的调试过程中，尽管可以随时中断程序的执行，但程序还是以正常的流程运行的。例如，按 F8 键逐语句执行代码时，在代码的左边会有一个黄色的箭头随着移动，该箭头的移动次序就是程序的执行流程。黄色箭头所指向的语句为下一条要执行的语句。如图 8.9 所示的语句“A=S/6”为下一条要执行的语句。



图 8.9 下一条语句

有的时候，在更改了某变量或属性的值后，希望重新执行代码的某部分来观察更改后的运行结果，这时，可以人为地指定下一条要执行的语句。

指定下一条要执行的语句的方法是：首先将光标插入到要设置为下一条语句的行，然后执行【调式】菜单中的【设置下一条语句】命令，则黄色箭头就会指向光标所在的语句行。此时，运行程序，就会从该行语句开始执行。

设置下一条语句最方便的办法是，将鼠标指针移动到黄色箭头上，然后拖动鼠标将黄色箭头拖动到指定的位置，就达到了设置下一条语句的目的。

VB6.0 基础教程 8.5 使用调试窗口

在程序调试过程中，对调试者最为重要的信息是：在运行过程中各变量和表达式的值的变化情况。这些信息能够为调试者提供分析依据，从而做出正确的判断。为此，VB 提供了三个调试窗口，分别是立即窗口、本地窗口和监视窗口。在逐语句执行代码时，可以通过它们来监视变量或表达式的值。

立即窗口

在程序进入中断模式后，一般会自动弹出立即窗口，如果界面上没有显示出立即窗口，可以执行【视图】菜单中的【立即窗口】命令来打开它。单击调试工具栏中的【立即窗口】按钮也可以打开立即窗口。

通过立即窗口，即可以监视当前过程中各变量或属性的值，还可以重新为变量或属性赋值。

1.通过立即窗口监视

VB 把立即窗口看作为一个名称为 Debug 的对象，Print 是它的一个很重要的方法。在程序代码中添加如下语句：

Debug.print 变量或属性

就能够将变量或属性的值显示在立即窗口中了，从而达到监视变量与属性值的目的。

例如，在计算数组各元素的总和时，在代码中添加一条“Debug.Print S”语句，如图 8.11 中（a）图所示。运行程序，立即窗口中就会显示出数组元素每次累加的结果，如图 8.11 中（b）图所示。

更灵活的监视变量或属性值的方法是，在程序进入中断模式后，在立即窗口中直接使用 Print 语句来输出变量或属性的值。例如，将断点设置在语句“A=S/6”处，如图 8.12 中（a）图所示，运行程序，当程序中断后，在立即窗口中输入“Prints”，按回车键，则输出变量 S 当前的值。同理，也可以输出循环变量 i 的当前值，如图 8.12 中（b）图所示。

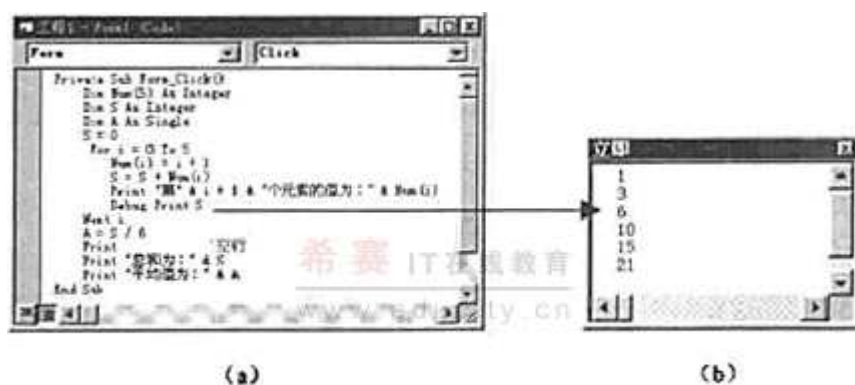


图 8.11 通过立即窗口监视



图 8.12 在立即窗口中使用 Print 语句

2.给变量或属性赋值

在中断模式下，利用立即窗口不仅能输出变量或属性的值，还能改变它们的值。在调试程序时，常常使用立即窗口给某变量赋予不同的值，然后配合 **Print** 语句的使用就可以观察到该变量值对其他变量值的影响。

例如，在代码中设置两个断点，如图 8.13 中（a）图所示，运行程序，当程序的运行在第一个断点处停下来时，在立即窗口中输入“S=5”，按回车键；继续运行程序，当程序的运行在第二个断点处停下来时，在立即窗口中输入“PrintS”，按回车键，则输出 S 的值为 26，如图 8.13 中（b）图所示。可见，变量 S 的初值变为 5，而不再是代码中所赋予的初值 0。

VB6.0 基础教程 8.5 使用调试窗口

在程序调试过程中，对调试者最为重要的信息是：在运行过程中各变量和表达式的值的变化情况。这些信息能够为调试者提供分析依据，从而做出正确的判断。为此，VB 提供了三个调试窗口，分别是立即窗口、本地窗口和监视窗口。在逐语句执行代码时，可以通过它们来监视变量或表达式的值。

立即窗口

在程序进入中断模式后，一般会自动弹出立即窗口，如果界面上没有显示出立即窗口，可以执行【视图】菜单中的【立即窗口】命令来打开它。单击调试工具栏中的【立即窗口】按钮也可以打开立即窗口。

通过立即窗口，即可以监视当前过程中各变量或属性的值，还可以重新为变量或属性赋值。

1.通过立即窗口监视

VB 把立即窗口看作为一个名称为 Debug 的对象，Print 是它的一个很重要的方法。在程序代码中添加如下语句：

Debug.print 变量或属性

就能够将变量或属性的值显示在立即窗口中了，从而达到监视变量与属性值的目的。

例如，在计算数组各元素的总和时，在代码中添加一条“Debug.Print S”语句，如图 8.11 中（a）图所示。运行程序，立即窗口中就会显示出数组元素每次累加的结果，如图 8.11 中（b）图所示。

更灵活的监视变量或属性值的方法是，在程序进入中断模式后，在立即窗口中直接使用 Print 语句来输出变量或属性的值。例如，将断点设置在语句“A=S/6”处，如图 8.12 中（a）图所示，运行程序，当程序中断后，在立即窗口中输入“Prints”，按回车键，则输出变量 S 当前的值。同理，也可以输出循环变量 i 的当前值，如图 8.12 中（b）图所示。



图 8.11 通过立即窗口监视



图 8.12 在立即窗口中使用 Print 语句

2. 给变量或属性赋值

在中断模式下，利用立即窗口不仅能输出变量或属性的值，还能改变它们的值。在调试程序时，常常使用立即窗口给某变量赋予不同的值，然后配合 Print 语句的使用就可以观察到该变量值对其他变量值的影响。

例如，在代码中设置两个断点，如图 8.13 中 (a) 图所示，运行程序，当程序的运行在第一个断点处停下来时，在立即窗口中输入“S=5”，按回车键；继续运行程序，当程序的运行在第二个断点处停下来时，在立即窗口中输入“PrintS”，按回车键，则输出 S 的值为 26，如图 8.13 中 (b) 图所示。可见，变量 S 的初值变为 5，而不再是代码中所赋予的初值 0。

VB6.0 基础教程 8.5.2 使用本地窗口

利用本地窗口不但可以查看当前过程中的所有变量取值，而且还可以查看该窗体及其上所有控件的属性取值。

在中断模式下，执行【视图】菜单中的【本地窗口】命令，或单击调试工具栏中的【本地窗口】按钮可以打开本地窗口。其中显示了当前过程中所有变量及其取值。

在本地窗口的表达式列表中显示的“Me”是指本窗体，单击其左边的加号节点可以展开它，其中列出了本窗体及其上所有控件的属性取值。

在本地窗口中还可以更改变量与属性的取值，选中某属性或变量，然单击它们的取值，即可更改其值了。

注意：在本地窗口中更改属性的值只是在本次运行时有效，并不是真正改变了对对象的属性设置。

在本地窗口最上一栏显示的是当前的过程，单击右边的显示有...符号的按钮，可打开【调用堆栈】对话框，如图 8.16 所示。在调试包含复杂的嵌套过程调用的应用程序时，【调用堆栈】对话框有助于了解过程调用的嵌套关系。在【调用堆栈】对话框中列出了当前过程正在调用的所有其他过程。当前过程位于最底部，它调用的某过程位于其上；该过程调用的另一过程又位于该过程的上面。因此，位于最上面的是最后调用的过程。



图 8.16 【调用堆栈】对话框

VB6.0 基础教程 8.5.3 监视窗口

视窗口用来显示监视表达式的值，在使用该窗口前，需要事先添加要监视的表达式。为监视窗口添加监视表达式的方法有两种：

- 1.使用【添加监视】对话框。

(1) 执行【调试】菜单中的【添加监视】命令，则弹出【添加监视】对话框，在【表达式】框中输入要监视的表达式，例如，要监视变量 S 的值，可以输入 S。

(2) 在【上下文】区中选择被监视的表达式所在的过程和模块。

(3) 在【监视类型】区中选择一种表达式的监视类型，如果选择【监视表达式】单项按钮，则监视窗口显示表达式的值。如果选择【当监视值为真时中断】单项按钮，则在程序运行中，当表达式的值为真（不为 0）时程序就进入到中断模式。如果选择【当监视值改变时中断】单项按钮，则在程序运行中，一旦表达式的值改变，程序就进入到中断模式。

(4) 设置各选项，单击【确定】按钮即可为程序添加一个监视表达式。重复上述操作，可以添加多个监视表达式。

2.使用【快速监视】对话框。

(1) 在【代码】窗口中选定要监视的表达式。

(2) 执行【调试】菜单中的【快速监视】命令，或单击调试工具栏中的【快速调试】按钮，打开【快速监视】对话框。

(3) 单击添加按钮即可将所选的表达式设置为监视表达式。

在添加了监视表达式后，它们会出现在监视窗口中，如图 8.19 所示。还可以更改或删除已添加的监视表达式。在监视窗口中选中某表达式，执行【调试】菜单中的【编辑监视】命令，则弹出【编辑监视】对话框，从中可以更改监视表达式及其各项设置，单击【删除】按钮可删除该监视表达式。

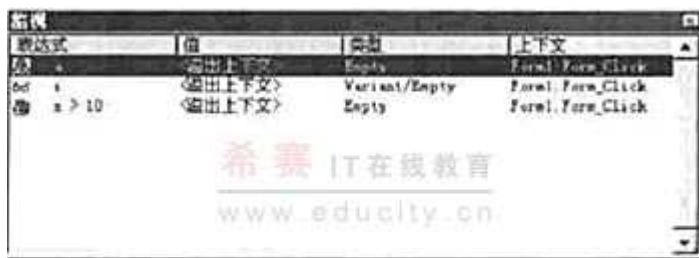


图 8.19 监视窗口

VB6.0 基础教程 8.6 错误捕捉

一个好的应用程序，不仅体现在它的功能强大与容易操作，还体现在它良好完善的错误处理能力。在编写程序时，要充分考虑到程序运行时可能会遇到的错误。例如，在做除法运算时，用户输入的除数可能为 0；在执行读取软盘操作时，软驱里可能没有放软盘。

当应用程序在 VB 环境中运行时，遇到错误将终止程序的运行，返回到 VB 环境。当应用程序被编译成 EXE 文件，在 Windows 环境中运行时，一旦发生运行错误，Windows 将终止应用程序的执行，并将控制权交归还给 Windows 系统。显然，这种处理错误的方式不是所希望的。一般的应用程序都会在运行时捕捉到错误，并且给出提示，以便让用户采取行动。

在 VB 中，要增加应用程序的处理错误的能力，需要做以下两步工作：

- (1) 设置错误陷阱
- (2) 编写错误处理程序

VB 提供了 On Error 语句设置错误陷阱，捕捉错误。On Error 语句有 3 种形式，如表 8.2 所示。

表 8.2 On Error 语句的 3 种形式

形式	含义
On Error GoTo 语句标号	在发生运行错误时，转到语句标号所指定的程序块执行错误处理程序。指定的必须在同一过程中，错误处理程序的最后必须加上 Resume 语句，以告知返回位置
On Error Resume Next	在发生运行错误时，忽略错误，转到发生错误的下一条语句继续运行
On Error GoTo 0	停止错误捕捉，由 VB 直接处理运行错误

Resume 语句应放置在出错处理程序的最后，以便错误处理完毕后，指定程序下一步做什么。Resume 语句也有 3 种形式，如表 8.3 所示：

表 8.3 Resume 语句的 3 种形式

形式	含义
Resume 标号	返回到标号指定的行继续执行，若标号为 0，则表示终止程序的执行
Resume Next	跳过出错语句，返回到出错语句的下一条语句处继续执行
Resume	返回到出错语句处重新执行

在 On Error 语句捕捉到错误后，Err 对象的 Number 属性返回错误的代号，通过错误代号即可知道引发错误的原因了。在编写错误处理程序时，一般使用“IfErr.Number=”语句或“select Case Err.Number”语句来判断错误的类型：

VB 提供的 Error 函数用于返回错误信息，其语法如下：

Error（错误代号）。

例如：

Form.Print.Error（11）。

语句将在窗体上显示“除数为零”。

这里编写一个用于计算两个数相除的小程序。如果用户输入的除数为零，会弹出消息框提示用户；如果没有输入除数或被除数，也会弹出消息框提示用户。

在窗体上放置 5 个标签控件、3 个文本框控件和 1 个按钮控件。其中各对象的属性设置如表 8.4 所示。

表 8.4 各对象的属性设置

对象	属性	值
窗体	Caption	错误处理
标签 1	Caption	被除数
标签 2	Caption	除数
标签 3	Caption	商
标签 4	Caption	=
文本框 1	名称	Text1
	Text	置空
文本框 1	名称	Text2
	Text	置空
文本框 1	名称	Text3
	Text	置空
按钮	名称	Command1
	Caption	结果

编写程序代码如下

```

Private Sub Command1_Click()

On Error GoTo WW “ 设置错误陷阱

Text3.Text=Text1.Text / Text2.Text

Exit Sub

"错误处理代码块

WW:

If Err.Number=11 Then “判断错误代号

MsgBox “除数不能为零，请重新输入！”，16.

“错误”

Else

MsgBog “出现其他错误！”，16，“错误”

End If

Resume Next

End Sub

```

运行程序，输入被除数与除数，单击【结果】按钮，就会计算出它们的商。如果输入的除数是 0，则弹出如图 8.22 所示的消息框，要求用户重新输入；如果没有输入被除数或除数，则弹出如图 8.23 所示的消息框，提示用户出错。



图 8.22 除数为零

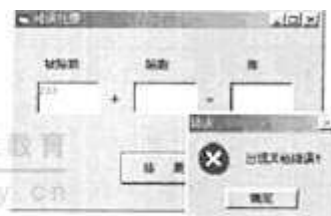


图 8.23 没有输入除数

第九章 图形程序设计

VB6.0 基础教程 9.1 图形控件

图形控件包括图片框控件、图像框控件、直线控件和形状控件 4 种，其中图片框和图像框是用来放置图片的，在第 5 章中已经介绍过，本节介绍另外两种图形控件。

9.1.1 直线控件

直线控件用来创建直线。它的使用方法与其他控件相同，在工具箱中单击直线控件图标，将鼠标移动到窗体上，在所需位置开始拖动鼠标，拖动到合适处后释放鼠标，则在鼠标的拖动起点与终点之间就出现了一段直线，如图 9.1 所示。

单击直线可选中它，并且在直线的两端出现两个小方块。将鼠标指针移动到某个方块上，则指针变成一个十字形，此时拖动鼠标，可以更改该直线的长度与方向，如图 9.2 所示。也可以拖动鼠标来改变直线的位置。

直线控件的属性较其他控件要少得多，主要用来设置直线的宽度、颜色以及线型等。

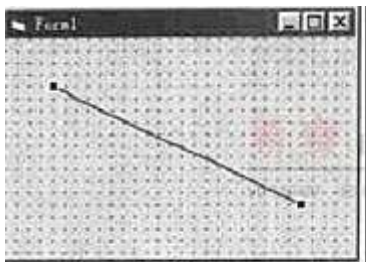


图 9.1 使用直线控件创建直线

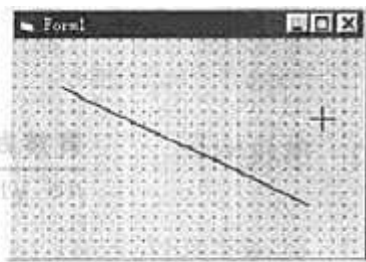


图 9.2 更改直线的长度与方向

表 9.1 直线控件一些较常用的属性

属性	含义
BorderColor	设置直线的颜色
BorderStyle	设置直线的线型，该属性有 7 个取值，各取值与对应的线型如表 9.2 所示
BorderWidth	设置直线的宽度
x1、y1	设置或返回直线的起点坐标
x2、y2	设置或返回直线的终点坐标

表 9.2 BorderStyle 属性的取值及其对应的线型

属性值	线型
0	透明，即直线显示不出来
1 (默认值)	实线
2	虚线
3	点线
4	点划线
5	双点划线
6	内实线

只有直线的宽度为 1（BorderWidth=1）时，BorderStyle 属性的 7 个取值才都有效，否则 BorderStyle 属性的取值只有 0 和 6 有效。例如，直线的宽度为 2 时，不能将其设置为虚线。如图 9.3 所示的是各种线型的比较，从上到下，各直线控件的 BorderStyle 属性的值依次为 1~6。

与其他控件不同的是，直线控件没有任何事件。因此，在程序运行时，它不能响应用户的任何操作。

VB6.0 基础教程 9.1 图形控件

图形控件包括图片框控件、图像框控件、直线控件和形状控件 4 种，其中图片框和图像框是用来放置图片的，在第 5 章中已经介绍过，本节介绍另外两种图形控件。

9.1.1 直线控件

直线控件用来创建直线。它的使用方法与其他控件相同，在工具箱中单击直线控件图标，将鼠标移动到窗体上，在所需位置开始拖动鼠标，拖动到合适处后释放鼠标，则在鼠标的拖动起点与终点之间就出现了一段直线，如图 9.1 所示。

单击直线可选中它，并且在直线的两端出现两个小方块。将鼠标指针移动到某个方块上，则指针变成一个十字形，此时拖动鼠标，可以更改该直线的长度与方向，如图 9.2 所示。也可以拖动鼠标来改变直线的位置。

直线控件的属性较其他控件要少得多，主要用来设置直线的宽度、颜色以及线型等。

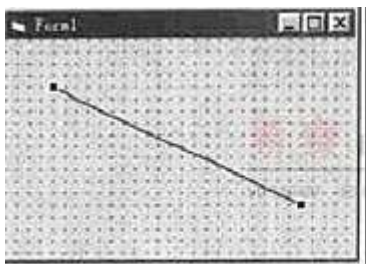


图 9.1 使用直线控件创建直线

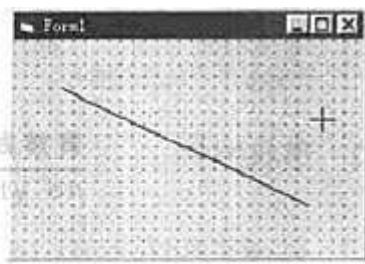


图 9.2 更改直线的长度与方向

表 9.1 直线控件一些较常用的属性

属性	含义
BorderColor	设置直线的颜色
BorderStyle	设置直线的线型，该属性有 7 个取值，各取值与对应的线型如表 9.2 所示
BorderWidth	设置直线的宽度
x1、y1	设置或返回直线的起点坐标
x2、y2	设置或返回直线的终点坐标

表 9.2 BorderStyle 属性的取值及其对应的线型

属性值	线型
0	透明，即直线显示不出来
1 (默认值)	实线
2	虚线
3	点线
4	点划线
5	双点划线
6	内实线

只有直线的宽度为 1 (BorderWidth=1) 时，BorderStyle 属性的 7 个取值才都有效，否则 BorderStyle 属性的取值只有 0 和 6 有效。例如，直线的宽度为 2 时，不能将其设置为虚线。如图 9.3 所示的是各种线型的比较，从上到下，各直线控件的 BorderStyle 属性的值依次为 1~6。

与其他控件不同的是，直线控件没有任何事件。因此，在程序运行时，它不能响应用户的任何操作。

VB6.0 基础教程 9.1.2 形状控件

使用形状控件可以方便地在窗体上绘制出矩形、正方形、圆、椭圆、圆角矩形和圆角正方形等 5 种基本几何图形。使用形状控件的方法与其他控件相同，这里不再赘述。

形状控件的 Shape 属性是它很主要的一个属性，该属性决定了形状控件所绘制图形的类型。表 9.3 中列出了 Shape 属性的值及含义。

表 9.3 Shape 属性的值及含义

属性值	含义
0 (默认值)	绘制矩形
1	绘制正方形
2	绘制椭圆
3	绘制圆
4	绘制圆角矩形
5	绘制圆角正方形

形状控件也有 BorderColor、BorderStyle 和 BorderWidth 属性，且含义与直线控件相同。在默认情况下，使用图形控件绘制出的图形的

背景是透明的，这是因为在默认情况下 BackStyle 属性的值为 0（透明）。将该属性的值设置为 1 即可在 BackColor 属性中指定图形的背景颜色。

形状控件的另一个重要属性是 FillStyle 属性，该属性用来决定图形的填充样式，表 9.4 中列出了它的取值及含义。

表 9.4 FillStyle 属性的取值及含义

属性值	含义
0	实心
1 (默认值)	透明
2	水平线
3	垂直线
4	向上对角线
5	向下对角线
6	交叉线
7	对角交叉线

如果图形的填充样式不是透明的，即 FillStyle 属性的值不为 1，则可以通过 FillColor 属性设置图形的填充颜色。

图形的各种填充效果，从左到右各图形的 FillStyle 属性的值依次为 0~7。

实例 9.1 形状控件的使用。

在该程序中，用户可以通过设置选项来决定形状的类型和填充样式。

在窗体中放置两个框架控件和一个形状控件，再在两个框架控件中分别放置 3 个单选按钮控件。其中各对象的属性设置如表 9.5 所示。

表 9.5 实例 9.1 中各对象的属性设置

对象	属性	值
窗体	Caption	形状控件的使用
框架 1	Caption	形状
框架 2	Caption	填充
单选按钮控件 1	名称	OpS
	Caption	矩形
	Index	0
单选按钮控件 2	Value	True
	名称	OpS
	Caption	圆
单选按钮控件 3	Index	1
	名称	OpS
	Caption	椭圆
单选按钮控件 4	Index	2
	名称	OpF
	Caption	透明
单选按钮控件 5	Index	0
	Value	True
	名称	OpF
	Caption	水平线
	Index	1

在【代码】窗口中编写代码如下：

```
Private Sub Ops.Click(Index As Integer)

    Select Case Index

        Case 0

            Shape1.Shape1=0

        Case 1

            Shape1.Shape1=3

        Case 2

            Shape1.Shape1=2

    End Select

End Sub

Private Sub OpF.Click(Index As Integer)

    Select Case Index
```

```

Case 0

Shapel.FillStyle = 1

Case 1

Shapel.FillStyle = 2

Case 2

Shapel.FillStyle = 7

End Select

End Sub

```

运行该程序，窗体如图 9.7 所示。单击【形状】设置区中的某单选按钮，则右边的图形就会变成所选的形状，单击【填充】设置区中的某单选按钮，则图形就会以所选的样式填充。



图 9.7 启动后的窗体



图 9.8 更改了图形的形状与填充样式

如图 9.8 所示的是选中【椭圆】与【对角交叉线】单选按钮后的效果。

VB6.0 基础教程 9.2 坐标系统

在 VB 中，控件放置在窗体或图片框等对象中，而窗体又放置在屏幕对象中，这些能够放置其他对象的对象称为容器，如窗体、图片框与屏幕都是容器。

每个容器都有一个坐标系统，以便为对象的定位提供参考。容器坐标系统的默认设置是：容器的左上角为坐标的原点。横向向右为 X 轴的正方向，纵向向下为 Y 轴的正方向。如图 9.9 所示的是窗体对象的默认坐标系统。



图 9.9 窗体默认的坐标系统

坐标的度量单位由容器对象的 ScaleMode 属性决定，ScaleMode 属性的值与对应的度量单位如表 9.6 所示。

表 9.6 ScaleMode 属性的值与对应的度量单位

属性值	单位	说明
0	用户定义 (User)	
1	Twip (缇)	默认值, 1 英寸=1440 缇
2	Point (磅)	1 英寸=72 磅, 1 磅=20 缇
3	Pixel (像素)	由显示器的分辨率决定
4	Character (字符)	水平每个单位=120 缇, 垂直每个单位=240 缇
5	Inch (英寸)	
6	Millimeter (毫米)	1 英寸=25.4 毫米
7	Centimeter (厘米)	1 英寸=2.54 厘米

对象的 Left 和 Top 属性决定了该对象左上角在容器内的坐标，Width 和 Height 属性决定了对象的大小，它们的单位总是与容器的度量单位相同。如果改变了容器的度量单位，则这 4 个属性的值都会发生相应的变化，以适应新的坐标系统，对象的实际大小与位置并不会改变。

使用默认的坐标系统有时很不方便，用户可以根据具体的需要重新定义容器的坐标系统。

属性 ScaleWidth 和 ScaleHeight 的值分别用来设置容器坐标系 x 轴与 Y 轴的正方向及最大坐标值。X 轴的度量单位为容器当前宽度

的 $1/\text{ScaleWidth}$ 、Y 轴的度量单位为对象当前宽度的 $1/\text{ScaleHeight}$ 。如果 ScaleWidth 的值小于 0，则 x 轴的正向向上；如果 ScaleHeight 的值小于 0，则 Y 轴的正向向上。属性 ScaleTop 与 ScaleLeft 的值用来设置容器左上角的坐标。

例如，将窗体的坐标属性设置为如表 9.7 所示，则对应的窗体坐标系统如图 9.10 所示，坐标原点定位在窗体的中点。

表 9.7 窗体的坐标属性设置

属性	值
ScaleWidth	100
ScaleHeight	100
ScaleTop	50
ScaleLeft	-50

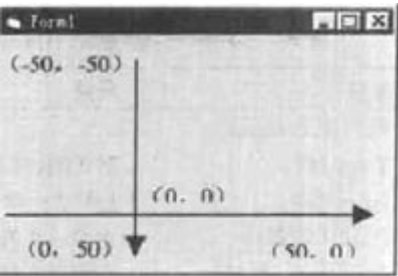


图 9.10 坐标原点在窗体的中心

如果将窗体的坐标属性设置为如表 9.8 所示，则对应的窗体坐标系统如图 9.11 所示，坐标原点定位在窗体的左下角，同时 Y 轴的正方向向上，这是符合人们习惯的一种坐标系统。

表 9.8 窗体的坐标属性设置

属性	值
ScaleWidth	100
ScaleHeight	-100
ScaleTop	100
ScaleLeft	0

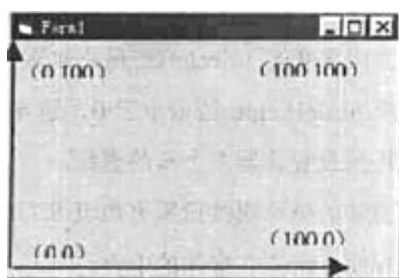


图 9.11 坐标原点在窗体的左下角

自定义坐标系统最简单的方法是使用 **Scale** 方法，其语法如下：

[对象].Scale[(x1,y1) , (x2,y2)]

其中对象可以是窗体或图片框，参数 (x1,y1) 用来定义对象左上角的坐标值，参数 (x2,y2) 用来定义对象右下角的坐标值。

例如，如图 9.50 所示的坐标系统可以使用如下语句来定义：

Scale (-50,-50) , (50,50)

图 9.11 所示的坐标系统可以使用如下语句来定义：

Scale (0,100) , (100,0)

VB6.0 基础教程 9.3 绘图属性

在对象（窗体或图片框）上绘制图形时，还需要设置对象的绘图属性以确定所绘制图形的特征，例如所画线的宽度以及图形的填充样式等。

1. 与 CurrentX 与 CurrentY。

属性使用 **Print** 方法在窗体或图片框中显示文本时，文本总是出现在当前坐标处。例如，在默认情况下，第一次使用 **Print** 方法输出的文本显示在窗体的左上角。通过 **CurrentX** 与 **CurrentY** 属性可以指定当前坐标，这两个属性在设计时不可用。

例如：

```

Private Sub Form_Click()

Scale(0,100)-(100,0) "自定义坐标系统

For i=10 To 80 Step 10

Currenty = i "指定当前坐标

Currenty = i

Print "清华大学"

Next

End Sub

```

运行该程序，文本在窗体上的显示效果如图 9.12 所示，如果在代码中不使用 CurrentX 与 CurrentY 属性指定当前坐标，则窗体上文本的显示效果如图 9.13 所示。

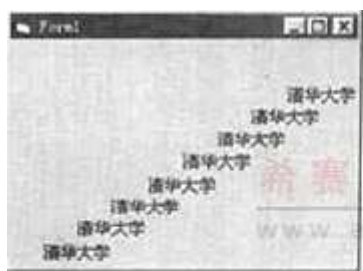


图 9.12 指定当前坐标



图 9.13 使用默认当前坐标

如果 AutoRedraw 属性的值为 True，则所绘制的图形是持久的。即当窗体被隐藏到其他窗口之后或调整了大小，使用 Print 方法显示的文本或使用图形方法绘制的图形都将重新显示。

如果 AutoRedraw 属性的值为 False，则所绘制的图形是临时的。当窗体被隐藏到其他窗口之后或调整了大小，窗体上的文本或图形将被掩盖掉。例如，图 9.14 中（a）图所示的是在窗体上正常显示的图

形和文本，（b）图所示的是将另外一个窗体移动到该窗体上，然后再移走后的效果，可见，被另一窗体掩盖部分的图形和文本消失了。

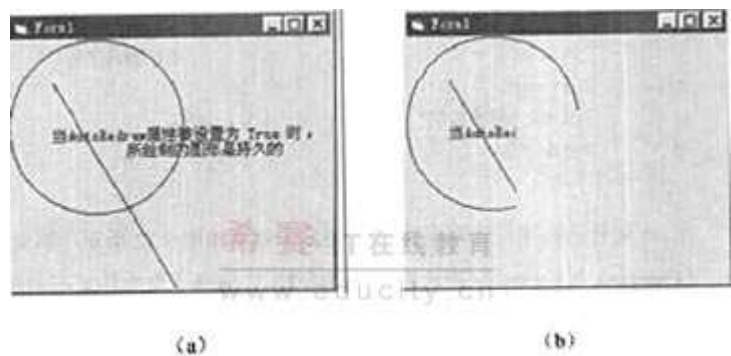


图 9.14 AutoRedraw 属性的值为 False

AutoRedraw 属性的默认值为 False，在使用 Print 方法或图形方法时，最好将该属性的值设置为 True。

3.其他绘图属性。

表 9.9 中列出了窗体与图片框控件的其他绘图属性。

表 9.9 绘图属性

属性	含义
DrawWidth	与直线控件的 BorderWidth 属性相同，用来设置对象上所画线的宽度或点的大小，以像素为单位，最小值为 1
DrawStyle	与直线控件的 BorderStyle 属性类似，用来设置对象上所画线的线型。需要注意的是，它的设置值与 BorderStyle 属性不完全相同，表 9.10 中列出了 DrawStyle 的取值与对应的线型
FillStyle	该属性与形状控件的 FillStyle 属性完全相同，用来设置对象上所画图形的填充样式
FillColor	设置对象上所画图形的填充颜色

表 9.10 DrawStyle 属性的取值及其对应的线型

属性值	线型
0 (默认值)	实线
1	虚线
2	点线
3	点划线
4	双点划线
5	透明，即直线显示不出来
6	内实线

VB6.0 基础教程 9.4 定义颜色

在 VB 中，颜色是以十六进制数表示的。例如，在【属性】窗口中设置 BackColor 与 ForeColor 等颜色属性时，出现的值总是一个十六进制数。以十六进制数来设置颜色既不方便也不直观，一般用户很

难看出颜色与十六进制数的对应关系。为此，VB 提供了一些颜色常数和颜色函数，使用它们可以方便直观地设置出想要的颜色。

1.颜色常量。

如果程序中只需要使用 8 种基本颜色，则使用 VB 提供的颜色常量即可达到目的。这些常量所代表的颜色可以从它们的名字上看出。

表 9.11 所示的是 8 种基本颜色与颜色常量的对应关系。

表 9.11 颜色常量及其对应的颜色

颜色常量	值 (十六进制)	对应颜色
VbBlack	&H0	黑色
VbRed	&HFF	红色
VbGreen	&HFF00	绿色
VbYellow	&HFFFF	黄色
VbBlue	&HFF0000	蓝色
VbMagenta	&HFF00FF	洋红
VbCyan	&H00FFFF	青色
VbWhite	&HFFFFFF	白色

例如，要将窗体（名称为 Form1）的背景色设置为红色，可以使用如下语句：

```
Form1.BackColor=&HFF.
```

也可以使用颜色常数来设置，语句如下：

```
Form.BackColor= VbRed
```

2.QBColor 函数

使用 QBColor 函数可以设置 16 种颜色，语法如下：

QBColor (Color)

参数 Color 是一个 0~15 的整数，每个整数代表一种颜色，表 9.12 中列出了该参数的取值与对应的颜色。

表 9.12 Color 参数的取值与对应的颜色

值	颜色	值	颜色
0	黑色	8	灰色
1	蓝色	9	亮蓝色
2	绿色	10	亮绿色
3	青色	11	亮青色
4	红色	12	亮红色
5	洋红色	13	亮洋红色
6	黄色	14	亮黄色
7	白色	15	亮白色

例如，下列语句的含义也是将窗体的背景色设置为红色。

```
Form1.BackColor=QBColor (4)
```

3.RGB 函数。

使用颜色常量和 QBColor 函数只能指定一些基本的颜色，而使用 RGB 函数则可以指定几乎所有的颜色。RGB 函数是通过指定红（Red）、绿（Green）、蓝（Blue）三原色的值来定义颜色的，其语法为：

RGB（红、绿、蓝）。

红、绿、蓝三原色的值均为 0~255 之间的整数，颜色值的不同组合将产生不同的颜色，从理论上讲，三原色混合可以产生 256×256×256 种颜色。表 9.13 中列出了基本颜色与对应的 RGB 函数。

表 9.13 基本颜色与对应的 RGB 函数

颜色	RGB 函数
黑色	RGB (0, 0, 0)
蓝色	RGB (0, 0, 255)
绿色	RGB (0, 255, 0)
红色	RGB (255, 0, 0)
青色	RGB (0, 255, 255)
洋红色	RGB (255, 0, 255)
黄色	RGB (255, 255, 0)
白色	RGB (255, 255, 255)

例如，使用 RGB 函数设置窗体背景色为红色的语句为：

```
Form1.BackColor= RGB (255,0,0)
```

实际上，对于颜色的十六进制数，每两位一组代表一种原色的颜色值，最低两位为红色的值，其次是绿色和蓝色的值。例如，十六进制数&H00FF00FF 对应 RGB（255,0,255），因此，它表示的颜色为洋红色。

VB6.0 基础教程 9.5 图形方法

图形方法是指窗体或图片框控件用于绘图的方法，其中包括 Line 方法、Circle 方法、Pset 方法以及 PaintPicture 方法等。使用这些方法可以绘制出直线、矩形、圆、椭圆、弧线、扇形、点以及各种曲线。

Line 方法用于绘制直线或矩形，其语法格式如下：

[对象].Line[[Step] (x1,y1) 1-[Step] (x2,y2) [,颜色][,B[F].

对象可以是窗体或图片框控件，其中各参数的含义如下：

Step:该参数是可选的，如果使用该参数，则表示起点坐标(x1, y1)或终点坐标 (x2, y2) 是相对当前点 (CurrentX,CurrentY) 的，而不是相对坐标原点的。

(x1,y1)：用于指定直线的起点，也是可选的，如果省略则起点为当前点 (CurrentX,CuxrerttY) 。

(x2, y2) 用于指定直线的终点。

颜色：可选的。用于指定所绘制的图形的颜色，可以使用 RGB 函数或 QBColor 参数指定颜色。如果省略，则使用对象（窗体或图片框）当前的 ForeColor 属性指定的颜色。

B:可选的。如果使用该参数，则绘制出的是矩形。其中 (x1,y1) 是指矩形左。上顶点的坐标， (x2,y2) 是指矩形右下顶点的坐标。

F:可选的。只有使用了参数 B 后才能使用该参数。如果使用该参数则矩形以指定的颜色填充；省略 F 时，矩形以对象当前的 FillColor 与 FillStyle 属性的设置填充。

实例 9.3 使用 Line 方法。

使用 Line 方法绘制一个柱状图表，每个柱的填充颜色与样式都不同，并且在柱的正上方标有柱的长度，如图 9.15 所示。

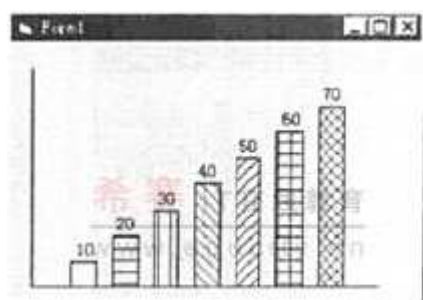


图 9.15 使用 Line 方法绘制的图表

程序代码如下：

```
Private Sub Form_Resize()  
    Const x0=15  
    Const y0=20  
    Cls  
    Scale (0,100)-(100,0) 自定义坐标系  
    Line (x0, y0)-(x0, 90) 绘制 Y 轴  
    Line (x0,y0)-(90,y0) 绘制 x 轴  
    For i=10 To 70 Step 10  
        FillStyle=i/10 设置填充样式  
        FillStyle=QBColor(i / 10 - 1) 设置填充颜色
```

```

Line(x0 + i, y0+i)-(x0+i+6, y0), , B 绘制矩形
CurrentX=x0+i-1
CueeentY=y0+i+8
Print
Next
End Sub

```

在该段代码中，常量 $x0$ 和 $y0$ 是柱状图表的坐标原点，定义这两个常量的好处在于；只要改变这两个常量的值，即可确定图表在窗体中的位置。例如，将 $x0$ 与 $y0$ 的值分别设置为 15 和 20,则图表在窗体中的位置如图 9.16 所示。

改变窗体的大小后，图表会随着窗体的大小自动缩放，如图 9.17 所示。

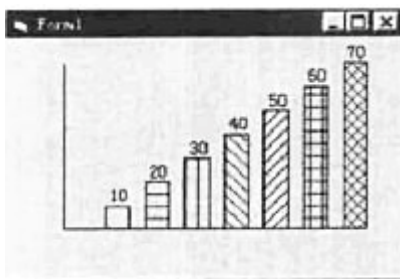


图 9.16 改变图表的位置

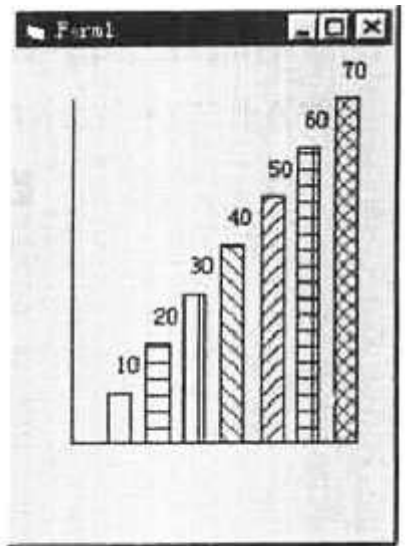


图 9.17 图表随窗体的大小自动缩放

实例 9.3 产生渐变背景。

许多 Windows 应用程序的安装界面，是以一个颜色由蓝至黑的渐变窗体为背景的。使用 VB 产生窗体背景色的渐变效果很简单，基本思想是：在窗体中从上至下依次绘制多个矩形，只要有足够多的矩形，同时使它们的填充色从蓝变化到黑，就能很好地模拟出渐变效果了。

代码如下：

```
Private Sub Gradient(TheObject As
Object,Redval,Greenval,Blueval)

    Dim Step,i,T,L,R,B

    Step=(TheObject.Height/60)

    T=0

    L=0

    R=TheObject.width
```

```

B=T+Step

For i=1 To 60

TheObject.Line(L,T)-(R,B),RGB(Redval,Greenval,Blueval),BF

Redval = Readval - 4

Greenval = Greenval - 4

Blueval=Blueval-4

If Readval <=0 Then Readval =0

If Greenval<=0 Then Greenval=0

If Blueval <=0 Then Blueval =0

T=B

B=B+Step

Next

End Sub

Private Sub Form_Resize()

Gradient Form1,0,0,255

End Sub

```

首先定义了一个名为 **Gradient** 的子过程，在该子过程中，使用了 **For** 循环语句来实现在窗体中从上至下依次绘制 60 个矩形，其中的 **Line** 方法是用来绘制矩形的，变量 **L** 与 **T** 为矩形左上顶点的坐标，**R** 与 **B** 为矩形右下顶点的坐标，变量 **Redval**, **Greenval**, **Blueval** 分别表示红、绿、蓝三原色的值。

在窗体的 **Resize** 事件中，以参数 **Fvrrnl**（窗体名）和颜色值（0,0,255）调用子过程 **Gradient**，运行程序后，即可看到窗体具有由蓝至黑的渐变背景。若以其他颜色值调用该函数，则可得到其他颜色的渐变背景。改变窗体的大小，渐变色会随着充满整个窗体（要将窗体的 **AutoRedraw** 属性设置为 **True**）。

VB6.0 基础教程 9.5 图形方法

图形方法是指窗体或图片框控件用于绘图的方法，其中包括 **Line** 方法、**Circle** 方法、**Pset** 方法以及 **PaintPicture** 方法等。使用这些方法可以绘制出直线、矩形、圆、椭圆、弧线、扇形、点以及各种曲线。

Line 方法用于绘制直线或矩形，其语法格式如下：

[对象].Line[[Step] (x1,y1) 1-[Step] (x2,y2) [,颜色][,B[F].

对象可以是窗体或图片框控件，其中各参数的含义如下：

Step:该参数是可选的，如果使用该参数，则表示起点坐标(x1, y1)或终点坐标 (x2, y2) 是相对当前点 (**CurrentX**,**CurrentY**) 的，而不是相对坐标原点的。

(x1,y1)：用于指定直线的起点，也是可选的，如果省略则起点为当前点 (**CurrentX**,**CuxrerttY**) 。

(x2, y2) 用于指定直线的终点。

颜色：可选的。用于指定所绘制的图形的颜色，可以使用 **RGB** 函数或 **QBColor** 参数指定颜色。如果省略，则使用对象（窗体或图片框）当前的 **ForeColor** 属性指定的颜色。

B:可选的。如果使用该参数，则绘制出的是矩形。其中 (x1,y1) 是指矩形左上顶点的坐标，(x2,y2) 是指矩形右下顶点的坐标。

F:可选的。只有使用了参数 **B** 后才能使用该参数。如果使用该参数则矩形以指定的颜色填充；省略 **F** 时，矩形以对象当前的 **FillColor** 与 **FillStyle** 属性的设置填充。

实例 9.3 使用 **Line** 方法。

使用 **Line** 方法绘制一个柱状图表，每个柱的填充颜色与样式都不同，并且在柱的正上方标有柱的长度，如图 9.15 所示。

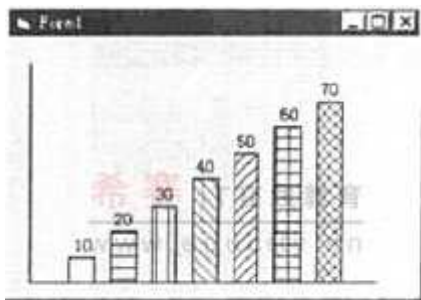


图 9.15 使用 **Line** 方法绘制的图表

程序代码如下：

```
Private Sub Form_Resize()  
    Const x0=15  
    Const y0=20  
    Cls  
    Scale (0, 100)-(100, 0) 自定义坐标系统  
    Line (x0, y0)-(x0, 90) 绘制 Y 轴  
    Line (x0, y0)-(90, y0) 绘制 x 轴  
    For i=10 To 70 Step 10
```

```

FillStyle=i/10 设置填充样式

FillStyle=QBColor(i / 10 - 1) 设置填充颜色

Line(x0 + i, y0+i)-(x0+i+6, yo), , B 绘制矩形

CurrentX=x0+i-1

CueeentY=y0+i+8

Print

Next

End Sub

```

在该段代码中，常量 $x0$ 和 $y0$ 是柱状图表的坐标原点，定义这两个常量的好处在于；只要改变这两个常量的值，即可确定图表在窗体中的位置。例如，将 $x0$ 与 $y0$ 的值分别设置为 15 和 20,则图表在窗体中的位置如图 9.16 所示。

改变窗体的大小后，图表会随着窗体的大小自动缩放，如图 9.17 所示。

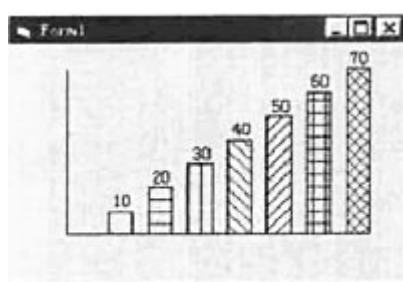


图 9.16 改变图表的位置

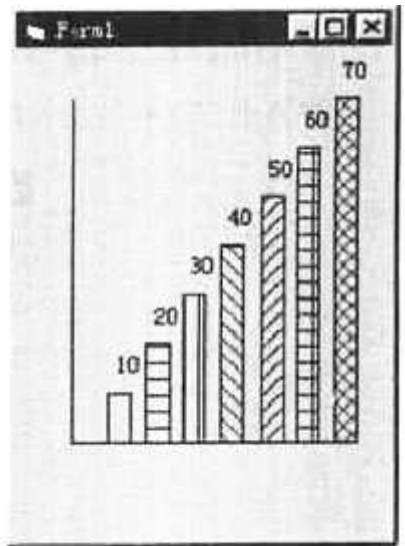


图 9.17 图表随窗体的大小自动缩放

实例 9.3 产生渐变背景。

许多 Windows 应用程序的安装界面，是以一个颜色由蓝至黑的渐变窗体为背景的。使用 VB 产生窗体背景色的渐变效果很简单，基本思想是：在窗体中从上至下依次绘制多个矩形，只要有足够多的矩形，同时使它们的填充色从蓝变化到黑，就能很好地模拟出渐变效果了。

代码如下：

```
Private Sub Gradient(TheObject As
Object,Redval,Greenval,Blueval)

    Dim Step,i,T,L,R,B

    Step=(TheObject.Height/60)

    T=0

    L=0

    R=TheObject.width
```

```

B=T+Step

For i=1 To 60

TheObject.Line(L,T)-(R,B),RGB(Redval,Greenval,Blueval),BF

Redval = Readval - 4

Greenval = Greenval - 4

Blueval=Blueval-4

If Readval <=0 Then Readval =0

If Greenval<=0 Then Greenval=0

If Blueval <=0 Then Blueval =0

T=B

B=B+Step

Next

End Sub

Private Sub Form_Resize()

Gradient Form1,0,0,255

End Sub

```

首先定义了一个名为 **Gradient** 的子过程，在该子过程中，使用了 **For** 循环语句来实现在窗体中从上至下依次绘制 60 个矩形，其中的 **Line** 方法是用来绘制矩形的，变量 **L** 与 **T** 为矩形左上顶点的坐标，**R** 与 **B** 为矩形右下顶点的坐标，变量 **Redval**, **Greenval**, **Blueval** 分别表示红、绿、蓝三原色的值。

在窗体的 **Resize** 事件中，以参数 **Fvrrnl**（窗体名）和颜色值（0,0,255）调用子过程 **Gradient**，运行程序后，即可看到窗体具有由蓝至黑的渐变背景。若以其他颜色值调用该函数，则可得到其他颜色的渐变背景。改变窗体的大小，渐变色会随着充满整个窗体（要将窗体的 **AutoRedraw** 属性设置为 **True**）。

VB6.0 基础教程 9.5.2 Circle 方法

Circle 方法用于绘制圆、椭圆、扇形或弧，其语法格式如下：

[对象。]**Circle**[[**Step**] (**x,y**)],半径[,颜色][,起始角][,终止角][长轴比率].

对象可以是窗体或图片框控件，其中各参数的含义如下：

Step:该参数是可选的，如果使用该参数，则表示圆心坐标 (**x,y**) 是相对当前点 (**CurrentX,CurrentY**) 的，而不是相对坐标原点的。

(**x,y**)：用于指定圆的圆心，也是可选的，如果省略则圆心为当前点 (**CurrentX,CurrentY**)。

半径：用于指定圆的半径，对于椭圆来讲，该值是椭圆的长轴长度。

颜色：指定所绘制图形的颜色。

起始角、终止角：用来指定圆弧或扇形的起始角度与终止角度，单位为弧度。取值范围为 $0\sim 2\pi$ 时，绘制的是圆弧：给起始角与终止角取值前添加一个负号，则所绘制的是扇形，负号表示绘制圆心到圆弧的径向线。省略这两个参数，则所绘制的是圆或椭圆。

VB 规定，从起始角按逆时针方向绘制圆弧只到终止角处，水平向右方向为 0 度，且与坐标系统无关，如图 9.18 所示。

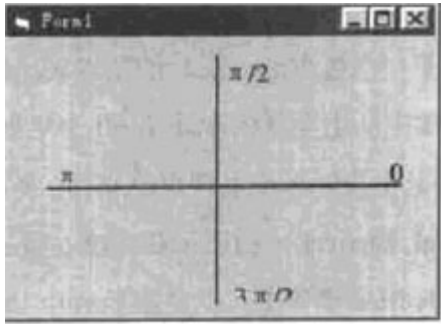


图 9.18 VB 中的角度规定

长短轴比率：当需要绘制椭圆时，可使用该参数指定椭圆长短轴的比率。若值大于 1，则所绘制的是竖立的椭圆；若值小于 1，则所绘制的是扁平的椭圆。该值的缺省值为 1，即省略时绘制的是圆。

例如，使用下列语句绘制出的各种图形如图 9.19 所示。

```
Const pi=3.1415926

Private Sub Form_Click()

    Scale (-100,100)-(100,-100)

    Circle(-50,50),30

    Circle(50,50),30,vbRed,,2

    Circle(50,50),30,vbRed,,,0.5

    Circle(-50,-50),30,vbBlue,pi/6,1.5*pi

    Circle(50,-50),30,vbYellow,-pi/6,-5/6*pi

End Sub
```

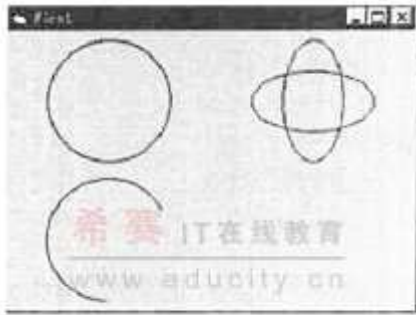


图 9.19 使用 Circle 方法绘制的图形

实例 9.4 绘制太极图

使用 Circle 方法绘制出如图 9.20 所示的太极图。

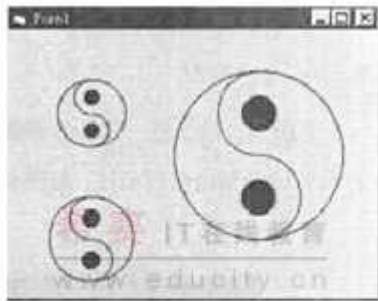


图 9.20 使用 Circle 方法绘制的太极图

代码如下：

```
Const pi=3.1415926
Sub Tjt(x,y,r)
    FillStyle=1
    Circle(x,y),r 绘制大圆
    Circle(x,y-r/2),r/2,,pi/2,1.5*pi 绘制弧线
    Circle(x,y+r/2),r/2,,1.5/pi,pi*2
    FillColor=vbBlack
    FillStyle=0
    Cricle(x,y+r/2),r/5 绘制小圆
```

```
Circle (x, y-r/2), r/5  
  
End Sub  
  
Private Sub Form_Click()  
  
Tjt 3000, 1500, 1000  
  
Tjt 1000, 2500, 500  
  
Tij 1000, 1000, 400  
  
End Sub
```

首先定义了一个名为 Tjt 的子过程,形参 x 和 Y 为太极图的圆心, r 为半径。在窗体的 Click 事件过程中以不同的参数调用 Tjt 子过程,运行程序,单击窗体后就会在窗体的不同位置绘制出大小不同的太极图。

VB6.0 基础教程 9.5.3 Pset 与 Pint 方法

Pset 方法用于画点,其语法格式如下:

[对象].Pset [Step] (x,y) [,颜色]

参数 (x,y) 为所画点的坐标,其他各参数的含义与 Line 方法相同。

使用 Pset 方法可以绘制出任何曲线,如图 9.21 所示的正弦曲线就是使用 Pset 方法绘制的。

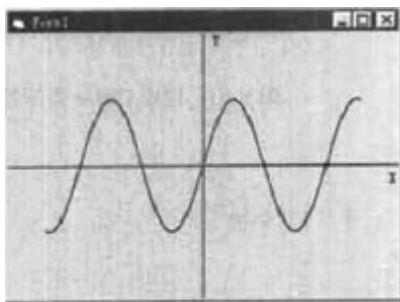


图 9.21 使用 Pset 方法绘制曲线

代码如下：

```
Private Sub Form_Click()

    Scale(-10,10)-(10,-10) 定义坐标系

    Line(-10,0)-(10,0) 绘制 X 轴

    Line(0,-10)-(0,10) 绘制 Y 轴

    CurrentX=0.5

    CurrentY=10

    Print "Y"

    Current=9.5

    Current=-0.5

    Print "X"

    For X=-8 To 8 Step 0.01

        y=5*sin(x) 绘制正曲线

    Next

End Sub
```

Point 方法用来返回指定点的颜色，其语法格式如下：

[对象].Point (x,y) 。

其中参数 (x,y) 是要获取颜色的点的坐标。

Point 方法返回的是一个像素的颜色值，可以使用下列代码从该颜色值中取出三原色的值，分别存放在三个变量中，其中 R 代表红色，G 代表绿色，B 代表蓝色。

```
P=Picture.Piont (x,y)
```

```
R=Mod 256
```

```
G= (p And &HFF00FF00) /256
```

```
B= (p And &HFF0000) /65536
```

Piont 方法和 Pset 方法配合使用，可以对图像进行多种处理，如柔化、锐化以及变色等。限于篇幅，这里不详细介绍。

VB6.0 基础教程 9.5.4 PaintPicture 方法

PaintPicture 方法是窗体或图片框的一个很实用的方法，它能够将窗体或图片框中的一个矩形区域的像素复制到另一个对象上。并且可以对复制的图形进行缩放和翻转等操作。

PaintPicture 方法的语法如下；Dpic.PaintPicture
Spit,dx,dy,dw,dh,sx,sy,sw,sh[,rop].

其中 Dpic 为目标对象，可以是窗体或图片框控件。各参数的含义如下：

Spit:传送源，可以是图片框和图像框，也可以是窗体的 Picture 属性。

dx、dy:目标区域某顶点的坐标。

dw、dh:目标区域的宽与高。如果为负值，则表示从 (dx,dy) 指定的

顶点起，沿坐标轴的负方向起。

sx、sy:要传送的矩形区域的某顶点坐标。

sw、sh:要传送的矩形区域的宽与高。

rop:指定所传送的像素与目标区域中现有像素的组合模式，例如，对传送像素和现有像素进行逻辑与、或和非等操作。缺省时表示将现有的像素替换成传送的像素。

实例 9.5 操作图形。

在该程序中，将 1 个图片框中的图形复制到另外 3 个图片框中，并且在复制图形时进行缩放和翻转操作。

在窗体中放置 4 个图片框控件和 3 个按钮控件，如图 9.22 所示，其中各对象的属性设置如表 9.14 所示。



图 9.22 实例 9.5 的窗体设计

表 9.14 实例 9.5 中各对象的属性设置

对象	属性	值
窗体	Caption	操作图片
图片框 1	名称	PicS
	AutoSize	True
	Picture	某位图
图片框 2	名称	PicD1
图片框 3	名称	PicD2
图片框 4	名称	PicD3
按钮 1	名称	Command1
	Caption	缩放
按钮 2	名称	Command2
按钮 3	Caption	水平翻转
	名称	Command3
	Caption	垂直翻转

编写代码如下：

```
Private w,h

Private Sub Form_load

w=PicS.width

h=PicS.Height

End Sub

Private Sub Command1_Click()

“缩小为原图的一半

PicD1.PaintPicture PicS,0,w/2,h/2,0,0,w,h

End Sub

Private Sub Command2_Click()

“缩小为原图的一半，并水平翻转

PicD2.PaintPicture PicS, 0, 0, w/2,h/2,w,0-w,h

End Sub

Private Sub Command3_Click()

“缩小为原图的一半，并垂直翻转

PicD3.PaintPicture PicS, 0, 0, w/2,h/2, 0, h, w,h

End Sub
```

为了实现图片的水平翻转，可以使用（sx,sy）指定要复制区域的右上顶点的坐标，并且设置复制区域的宽为负值。为了实现图片的垂直翻转，可以使用（sx,sy）指定要复制区域的左下顶点的坐标，并且设置复制区域的高为负值。

运行该程序，单击各按钮，则左边图片框中的图片就被复制到了右边的图片框中，并且对图片进行了缩放和旋转操作，如图 9.23 所示。



图 9.23 复制图片的效果

第十章 文件管理及操作

VB6.0 基础教程 10.1 文件系统的基本操作

VB 提供了一些用于处理文件系统的语句，使用这些语句可以在 VB 应用程序中进行更改当前目录、建立或删除目录、删除文件等基本操作。

目录操作

1. 获取指定驱动器的当前路径。

要获取某驱动器的当前路径，可以使用 CurDir 函数，它的语法是：

CurDir [drive].

参数 drive 是指要获取信息的驱动器名称，如果忽略该参数，则 CurDir 函数返回当前驱动器的当前路径。

例如，如果驱动器 E;的当前路径为“E: \Too1\Qicq”,则如下语句将在窗体上显示“E: \Too1\Qicq”.

`Print CurDir (“E”)`。

2.更改当前驱动器。

使用 `ChDrive` 语句可以更改当前驱动器，其语法为：

`ChDrive drive.`

参数 `drive` 为要指定为当前驱动器的名称，例如，将驱动器 `A:` 指定为当前驱动器的语句为：

`ChDrive “A”.`

3.更改当前路径。

使用 `ChDir` 语句可以更改当前路径，其语法为：

`ChDir Path.`

参数 `Path` 为要指定的路径，如果在路径中没有指定驱动器的名称，则表示驱动器为当前驱动器。例如，将路径 `C:\Windows` 指定为当前路径的语句为：

`ChDir “C:\Windows”.`

4.建立与删除目录。

使用 `Mkdir` 语句可以创建一个新的目录，其语法为：

`Mkdir Path.`

参数 `Path` 用来指定所要创建的目录以及目录所在的路径。 `Path` 可以包含驱动器。如果没有指定路径，则 `Mkdir` 会在当前路径下创建新的目录。

例如：

MKDir “C:\aa” 在 C 盘中创建目录 aa

MKDir “C:\Windows\bb” 在 C 盘 windows 目录中创建子目录 bb

MKDir “CC” 在当前路径下创建目录 cc

使用 Rmdir 语句可以删除某一空目录，其语法为：

Rmdir Path.

例如：

MKDir “C:\aa” 删除 C 盘中目录 aa

MKDir “C:\Windows\bb” 删除 C 盘 windows 目录中的子目录 bb

MKDir “CC” 删除当前路径下创建目录 cc

注意：Rmdir 语句只能用来删除空的目录，如果目录中还包含有子目录或文件，则必须先删除子目录和文件。

VB6.0 基础教程 10.1 文件系统的基本操作

VB 提供了一些用于处理文件系统的语句，使用这些语句可以在 VB 应用程序中进行更改当前目录、建立或删除目录、删除文件等基本操作。

目录操作

1. 获取指定驱动器的当前路径。

要获取某驱动器的当前路径，可以使用 CurDir 函数，它的语法是：

CurDir [drive].

参数 drive 是指要获取信息的驱动器名称，如果忽略该参数，则 CurDir 函数返回当前驱动器的当前路径。

例如, 如果驱动器 E; 的当前路径为“E: \Too1\Qicq”, 则如下语句将在窗体上显示“E: \Too1\Qicq”。

```
Print CurDir (“E”)。
```

2.更改当前驱动器。

使用 ChDrive 语句可以更改当前驱动器, 其语法为:

```
ChDrive drive.
```

参数 drive 为要指定为当前驱动器的名称, 例如, 将驱动器 A: 指定为当前驱动器的语句为:

```
ChDrive “A”.
```

3.更改当前路径。

使用 ChDir 语句可以更改当前路径, 其语法为:

```
ChDir Path.
```

参数 Path 为要指定的路径, 如果在路径中没有指定驱动器的名称, 则表示驱动器为当前驱动器。例如, 将路径 C: \Windows 指定为当前路径的语句为:

```
ChDir “C:\Windows”.
```

4.建立与删除目录。

使用 Mkdir 语句可以创建一个新的目录, 其语法为:

```
Mkdir Path.
```

参数 Path 用来指定所要创建的目录以及目录所在的路径。 Path 可以包含驱动器。如果没有指定路径, 则 Mkdir 会在当前路径下创建新的目录。

例如：

MKDir “C:\aa” 在 C 盘中创建目录 aa

MKDir “C:\Windows\bb” 在 C 盘 windows 目录中创建子目录 bb

MKDir “CC” 在当前路径下创建目录 cc

使用 Rmdir 语句可以删除某一空目录，其语法为：

Rmdir Path.

例如：

MKDir “C:\aa” 删除 C 盘中目录 aa

MKDir “C:\Windows\bb” 删除 C 盘 windows 目录中的子目录 bb

MKDir “CC” 删除当前路径下创建目录 cc

注意：Rmdir 语句只能用来删除空的目录，如果目录中还包含有子目录或文件，则必须先删除子目录和文件。

VB6.0 基础教程 10.1.2 文件操作

文件的操作包括拷贝文件、删除文件、重命名文件和设置文件属性等。在操作文件时，文件必须是关闭的，否则会产生运行错误。下面逐一介绍 VB 中的各种文件操作语句。

1.拷贝文件。

使用 FileCopy 语句可以在磁盘介质间拷贝文件，其语法为：

FileCopy Source. Destination.

参数 Source 用来指定源文件及其路径。参数 Destination 用来指定目标文件及其路径。如果没有指定路径，则默认路径为当前路径。

例如：

将 C 盘 Windows 目录中的文件 command.com 拷贝到 F 盘，并且文件名变为 cc.com

```
FileCopy "C:\Windows\command.com,"F:\cc.com
```

将 C 盘 Windows 目录中的文件 command.com 拷贝到当前路径下，且仍使用原名

```
FileCopy "C:\windows\command.com,"command.com
```

2.删除文件。

使用 Kill 语句可以删除磁盘中已存在的文件，其语法为：

```
Kill PathName.
```

参数 PathName 用来指定所要删除的文件及其路径。如果没有指定路径，会删除当前路径下的文件。

Kill 语句支持多字符（*）和单字符（?）等通配符来指定多重文件。

例如：

```
Kill "D:\vcd\mm.dat" 删除 D 盘 Vcd 目录中的 mm.dat 文件
```

```
Kill "Capter1.doc" 删除当前路径中的 Capter1.doc 文件
```

```
Kill "E:\temp\*.txt" 删除 E 盘 Temp 目录中的所有后缀为 TXT 的文件
```

```
Kill "E:\temp\*.*" 删除 E 盘 Temp 目录中的所有文件
```

3.重命名文件。

使用 Name 语句可以重命名文件或移动文件，其语法为：

```
Name OldPathName As NewPathName.
```

参数 `OldPathname` 用来指定所要重命名的文件及其路径，参数 `NewPathname` 用来指定文件的新名称及其路径。如果 `NewPathName` 参数指定的路径与 `OldPathName` 参数指定的路径不同，则文件将被移动到新的路径下。

例如：

将 D 盘中的文件 `oicq99b.exe` 重命名 `oicq.exe`

`Name "D:\oicc99b.exe" AS "D:\ociq.exe"`

将 D 盘中的文件 `oicq99b.exe` 移动到 E 盘的 `Temp` 目录中，并重命名为 `oicq.exe`

`Name "D:\ociq99b.exe" AS "E:\Temp\oicq.exe"`

`Name` 语句对目录也有效，例如：

将 D 盘中的 `Tool` 目录重命名为 `TT`

`Name "D:\Tool" As "D:\TT"`

将 E 盘的 `oicq` 目录移动到 D 盘的 `Tools` 目中

`Name "E:\Oicq" As "D:\Tools\Oicq"`

4. 设置文件的属性。

使用 `SetAttr` 语句可以设置文件或目录的属性，其语法为：

`SetAttr PathName, VbFileAttribute.`

参数 `Pathname` 用来指定所要设置属性的文件或目录，参数 `VbFileAttribute` 用来指定文件或目录的属性，其取值及含义如表 10.1 所示。

表 10.1 VbFileAttribute 参数的取值及含义

常数	值	含义
vbNormal	0	常规 (缺省值)
VbReadOnly	1	只读
VbHidden	2	隐藏
VbSystem	4	系统文件
vbArchive	32	上次备份以后, 文件已经改变

注意: 要删除和设置属性的文件必须是关闭的, 否则会产生运行错误。

VbFileAttribute 参数的取值也可以是各取值的和, 这一点与在前面介绍的通用对话框的 Flags 属性类似。

例如:

设置 D 盘 Temp 目录中 mytext.txt 文件的属性为只读

```
setAttr"D:Temp\mytext.txt",1
```

设置 D 盘 Temp 目录的属性为隐藏

```
SetAttr "D:\Temp",2
```

设置 E 盘中 yy.jpg 文件的属性为只读和隐藏

```
SetAttr "E:\yy.jpg",3
```

函数 GetAttr 用来返回文件的属性设置, 例如, 如果 GetAttr (E:\yy.jpg) 的返回值为 1, 则表明文件 yy.jpg 的属性为只读。如果 GetAttr 函数的返回值为 16, 则表明是目录。

5. 获取文件的大小。

使用 FileLen 函数可以获取文件的大小, 其语法为:

```
FileLen (PathName)
```

参数 PathName 用来指定要获取大小的文件及其路径。函数的返回值为一个长整型值, 代表文件的大小, 单位是字节。

例如：

显示 E 盘中 form1.frm 文件的大小，单位为字节。

```
Print FileLen ("E:\form1.frm")
```

VB6.0 基础教程 10.2.1 驱动器列表框

驱动器列表框用来显示当前系统所安装的驱动器，例如，软驱、硬盘的各分区和光驱等。驱动器列表框是一个下拉式列表框，平时只显示一个驱动器（在默认情况下，显示的是当前驱动器的名称）。单击列表框右边的向下箭头，就会下拉出一个驱动器列表，列出当前系统安装的所有驱动器，以供用户选择，如图 10.1 所示。

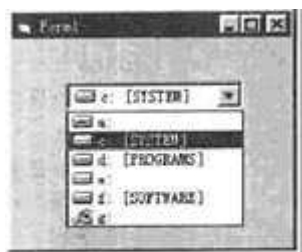


图 10.1 显示系统驱动器列表

驱动器列表框的最重要的属性是 **Drive** 该属性用来在运行时设置或返回所选择的驱动器，在设计时不可用。

例如，将如下语句添加到窗体的 **Load** 事件中，则程序启动后驱动器框中显示的将是指定的驱动器 **E:**而不是当前驱动器。

```
Drive1.Drive="E"
```

在驱动器列表框中选择驱动器并不能自动更改该系统的当前驱动器，要使用户在驱动器列表框中的操作影响到系统，还需要编写一定的代码。

改变驱动器列表框的 **Drive** 属性的设置值会触发它的 **Change** 事件。因此，在 **Change** 事件过程中，可用 **ChDrive** 语句来更改系统当前驱动器，语句如下：

ChDrive Drivel.Drive

VB6.0 基础教程 10.2.1 驱动器列表框

驱动器列表框用来显示当前系统所安装的驱动器，例如，软驱、硬盘的各分区和光驱等。驱动器列表框是一个下拉式列表框，平时只显示一个驱动器（在默认情况下，显示的是当前驱动器的名称）。单击列表框右边的向下箭头，就会下拉出一个驱动器列表，列出当前系统安装的所有驱动器，以供用户选择，如图 10.1 所示。

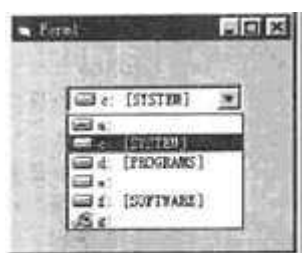


图 10.1 显示系统驱动器列表

驱动器列表框的最重要的属性是 **Drive** 该属性用来在运行时设置或返回所选择的驱动器，在设计时不可用。

例如，将如下语句添加到窗体的 **Load** 事件中，则程序启动后驱动器框中显示的将是指定的驱动器 **E:**而不是当前驱动器。

Drivel.Drive="E"

在驱动器列表框中选择驱动器并不能自动更改该系统的当前驱动器，要使用户在驱动器列表框中的操作影响到系统，还需要编写一定的代码。

改变驱动器列表框的 **Drive** 属性的设置值会触发它的 **Change** 事件。因此，在 **Change** 事件过程中，可用 **ChDrive** 语句来更改系统当前驱动器，语句如下：

ChDrive Drivel.Drive

VB6.0 基础教程 10.2.2 目录列表框

目录列表框用于显示当前驱动器上的目录结构。它以根目录开头，显示的目录按照子目录的层次依次缩进，如图 10.2 所示。

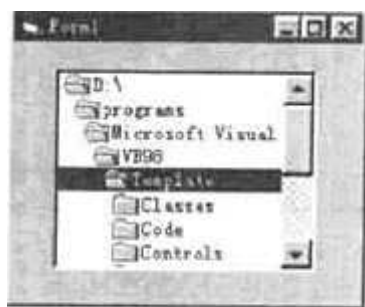


图 10.2 目录列表框

双击某一目录，可打开该目录，即显示该目录中的所有子目录。被打开的目录的图标为一个打开状的文件夹。双击打开的目录可将其关闭，其中的子目录不再显示出来，并且目录的图标变成一个关闭状的文件夹。

目录列表框的最重要的属性是 **Path**,该属性用来在运行时设置或返回所选择的路径，在设计时不可用。

同样，改变目录列表框的 **Path** 属性的设置值会触发它的 **Change** 事件。因此，在 **Change** 事件过程中，可用 **ChDir** 语句来更改系统当前路径，语句如下：

ChDir Dir1.Path

目录列表框只能显示当前驱动器下的目录，如果要显示其他驱动器下的目录结构，则必须使用 Path 属性来设置它的路径，最佳选择是将目录列表框与驱动器列表框配合使用。

实例 10.1 目录列表框与驱动器列表框的配合使用。

在窗体上放置一个驱动器列表框、一个目录列表框、一个标签控件和一个文本框控件。如图 10.3 所示，其中各对象属性的设置如表 10.2 所示。



图 10.3 实例 10.1 的窗体设计

表 10.2 实例 10.1 中各对象的属性设置

对象	属性	值
窗体	Caption	选择路径
驱动器列表框	名称	Drive1
目录列表框	名称	Dir1
标签	Caption	所选择的路径:
文本框	名称	Text1
	MultiLine	True
	Text1	置空

编写代码如下：

```
Private Sub Drver_Change()  
  
Dir1.path=Drive1.drive  
  
End Sub  
  
Private Sub Dir1_Change()  
  
Text1.Text=Dir1.Path  
  
End Sub
```

运行程序，在驱动器列表框中选择驱动器，则目录列表框中的目录会同步发生相应的改变；在目录列表框中选择目录，则文本框中会显示出当前所选择的路径，如图 10.4 所示。



图 10.4 选择路径

VB6.0 基础教程 10.2.3 文件列表框

文件列表框用来显示当前目录中的文件列表。文件列表框有 4 个重要的属性，下面分别介绍。

1.Path 属性。

Path 属性用来设置或返回列表框中所显示文件的目录，在设计时不可用。文件列表框常常与目录列表框和驱动器列表框一起使用。在目录列表框的 Change 事件中添加如下语句：

```
File1.Path=Dir1.Path
```

即可将目录列表框与文件列表框关联起来，当在目录列表框中选择一个目录时，文件列表框中会自动同步显示出该目录中的所有文件。

当文件列表框的 Path 属性改变后，会触发它的 PathChange 事件。

2.Patted 属性。

Patten 属性用来设置或返回文件列表框中所显示的文件类型，该属性既可以在设计时通过【属性】窗口设置，也可以在代码中设置。

Patterns 属性的默认值为*、*,即显示所有文件。当 Patterns 属性改变后,会触发文件列表框的 pattenChange 事件。

例如,要使文件列表框中只显示文本文件,则应该将 Patten 属性的值设置为“*.TXT”,要设置多个文件类型,可以使用分号(:)来分隔。

例如:

File1.Pattern=“.DOC” 只显示 word 文档文件

File1.Pattern=“*.EXE;*.COM” 显示 EXE 和 COM 文件

File1.Pattern=“*BMP;*GIF;*JPG” 显示几种图形文件

3. FileName 属性。

FileName 属性用来设置或返回文件列表框中所选文件的路径和文件名,如果没有选择任何文件,则返回一个空字符串。FileName 属性在设计时不可用。

例如:

Print File1.FileName 显示用户所选择的文件

在文件列表框中只显示 C 盘 windows 目录中的 Command.com 文件

File1.FileName=“C:\windows\Command.com”

在文件列表框中显示 C 盘 windows 目录中的 EXE 文件

File1.FileName=“C:\windows*.exe”

4.与文件属性有关的属性。

表 10.3 中列出了与文件属性有关的 4 个属性，它们用来决定在文件列表框中显示哪一类属性的文件。

表 10.3 与文件属性有关的属性

属性	含义	默认值
Archive	当该属性值为 True 时，显示以前备份后修改过的文件，否则不显示	True
Normal	当该属性值为 True 时，显示普通文件	True
System	当该属性值为 True 时，显示系统文件	False
Hidden	当该属性值为 True 时，显示隐藏文件	False
ReadOnly	当该属性值为 True 时，显示只读文件	True

实例 10.2 文件控件的使用。

在该程序中，使用文件控件自行设计一个文件管理对话框，通过该对话框可以浏览系统中的文件，还可以进行文件的查找和删除操作。

在窗体中放置 1 个驱动器列表框、1 个目录列表框、1 个文件列表框、1 个标签控件、1 个文本框控件和 3 个按钮控件，如图 10.5 所示，其中各对象的属性设置如表 10.4 所示。



图 10.5 实例 10.2 的窗体设计

表 10.4 实例 10.2 中各对象的属性设置

对象	属性	值
窗体	Caption	文件控件的使用
驱动器列表框	名称	Drive1
目录列表框	名称	Dir1
文件列表框	名称	File1
标签	Caption	请输入所要查找的文件名称:
文本框	名称	TexFind
	Text	置空
按钮 1	名称	ComFind
	Caption	查找
按钮 2	名称	ComDel
	Caption	删除
对象	属性	值
按钮 3	名称	ComClose
	Caption	关闭

编写代码如下:

```

Private Sub Form_Load()

    Driver.Drive= "F"

    Dir1.Path= "F: \"

End Sub

Private Sub Drive1_Change()

    Dir1.Path=Drive1.Drive 关联驱动器列表框和目录
列表框

Private Sub Dir1_Change()

    File1.Path=Dir1.Path 关联目录列表框和文件列表
框

End Sub

Private Sub ComFind_Click()

    On Error GoTo wwl

    File1.FileName=TexFind.Text 查找文件

Exit Sub

```

```

ww1: 错误处理代码

If Err.Number=53 Then

MsgBox “文件未找到! ”, 64, “提示”

Else

MsgBox “出现未知错误! ”, 64, “提示”

End If

Resume Next

End Sub

Private Sub ComDel_Click()

```

确定所选文件的路径及文件名

```

If Right(Dir1.Path, 1)= “\” Then

Else

Delfile=Dir1.Path & “\” &File1.FileName

End If

On Error GoTo ww2

msg=MsgBox(“是否真的删除文件? ”, 36, “警告”)

If msg=6 Then 判断用户做出的选择

Kill Delfile 删除文件

File1.Refresh 刷新文件列表框

Exit Sub

ww2:

If Err.Number=75 Then

```

```
MsgBox “文件不能被删除!”, 64, “提示”  
  
Else  
  
MsgBox “出现未知错误!”, 64, “提示”  
  
End If  
  
Resume Next  
  
End Sub  
  
Private Sub ComClose_Click()  
  
End  
  
End Sub
```

在程序代码中，首先在窗体的 **Load** 事件中设置了文件控件的初始值，即在启动程序后。驱动器列表框显示的是 **P** 盘，目录列表框显示的是 **F** 盘的根目录。驱动器列表框和目录列表框的 **Change** 事件过程用来关联 3 个文件控件。

在查找文件时，如果用户输入的文件不存在，则会产生错误，并且错误号为 53。在删除文件时，也可能会出现错误，例如，要删除的文件是只读的或文件已被打开，就会产生一个错误号为 75 的错误。为了处理这些错误，在【查找】按钮和【删除】按钮的 **Click** 事件过程中都设置了错误陷阱及错误处理代码。

在删除文件时，首先要获得用户所选文件的路径及文件名。可以使用目录列表框的 **Path** 属性所返回的路径和文件列表框的 **FileName** 属性返回的文件名来“拼接”出 **Kill** 语句所需要的参数。需要注意的是，**Path** 属性返回根目录时，已经带有冒号（:）和反斜杠（\），例如

返回的是 C.1 而不是 C 或者 C:.因此,在代码中使用了 If 语句,来判断是否有必要在文件名与路径之间添加一个反斜杠。

运行该程序,在驱动器中选择驱动器,则目录列表框和文件列表框都会同步变化,在目录列表框中选择目录,则文件列表框会同步显示出所选目录中的文件。

在文本框中输入要查找的文件名称,单击【查找】按钮,则文件列表框中就会显示出该文件,如图 10.7 所示。如果查找的文件不在当前文件列表中或不存在,则弹出消息框,提示文本找到。在查找文件时也可以使用通配符,例如,输入*.EXE,则文件列表框中显示出当前目录中的所有 EXE 文件。



图 10.7 查找到某个文件

在文件列表框中选中某个文件,单击【删除】按钮则弹出消息框,提示用户是否真的删除文件。单击【是】按钮则被选中的文件将被删除,单击【否】按钮则不删除该文件。

如果所要删除的文件是只读文件或文件已被打开,则执行文件的删除操作时会弹出消息框,提示用户此文件不能被删除。

VB6.0 基础教程 10.3 文件的访问

文件是指存放在外部介质(如磁盘)上的数据的集合,每一个文件由一个文件名作为其标识。在应用程序中,常常需要将数据以文件

的形式存储在磁盘中，在以后还可以读取保存在文件中的数据，应用程序与数据文件之间的这种读写操作称之为文件的访问。对于不同类型的文件，访问方式也不同。本节首先介绍文件的分类，然后介绍对各种类型文件的访问。

文件的分类

可以从不同角度对文件进行分类，按照文件的存取方式及其组成结构可以将文件分成顺序文件和随机文件。

1.顺序文件。

顺序文件是指顺序存取的文件，这是一种结构相对简单的文件。顺序存取是指数据是依次写入文件的，在读取时，数据又依次被读出来。即数据的存取顺序与它在文件中的实际次序相一致。例如，要读取最后一个数据，也必须从第一个数据开始读起，然后依次读到最后一个数据。

2.随机文件。

随机文件是指随机存取的文件。随机文件是以记录为单位读写数据的，一个记录一般包含有多个数据项。例如，学生的个人信息就是一个记录，它由学号、姓名、性别和年龄等数据项组成。

随机文件中的每个记录都有一个记录号，在读写记录时，只要指出记录号就可以直接读写该记录了，而不需要依次读取它前面的各个记录。

按照文件存储数据的形式又可以将文件分成 ASCII 码文件和二进制文件。

1.ASCII 码文件。

ASCII 码文件是指文件中的数据是以 ASCII 码进行编码存储的。

2.二进制文件。

二进制文件是指文件中的数据是以二进制格式进行编码存储的，它允许用户以字节为单位进行数据的读写。这类文件的操作灵活性较大，但编程工作量也较大。

与文件读写有关的重要函数有 3 个：LOF, LOC 和 EOF.

1.LOF 函数。

语法：LOF (<文件号>)

LOF 函数用来返回指定文件的字节数，如果返回值为 0,则表示文件为空文件。

2.LOC 函数。

语法：LOC (<文件号>)。

LOC 函数用来返回已打开的文件中读写的位置。对于随机文件，它将返回最近读写的记录号；对于二进制文件，它将返回最近读写的字节的位置；对于顺序文件，LOC 函数的返回值在实际，扣没有什么用处。

3.EOF 函数。

语法：EOF (<文件号>)。

当文件指针到达文件末尾时，EOF 函数返回 True（真），否则返回 False（否）。

对于顺序文件，EOF 函数常常用来判断是否已到文件末尾。

VB6.0 基础教程 10.3 文件的访问

文件是指存放在外部介质（如磁盘）上的数据的集合，每一个文件由一个文件名作为其标识。在应用程序中，常常需要将数据以文件的形式存储在磁盘中，在以后还可以读取保存在文件中的数据，应用程序与数据文件之间的这种读写操作称之为文件的访问。对于不同类型的文件，访问方式也不同。本节首先介绍文件的分类，然后介绍对各种类型文件的访问。

文件的分类

可以从不同角度对文件进行分类，按照文件的存取方式及其组成结构可以将文件分成顺序文件和随机文件。

1.顺序文件。

顺序文件是指顺序存取的文件，这是一种结构相对简单的文件。顺序存取是指数据是依次写入文件的，在读取时，数据又依次被读出来。即数据的存取顺序与它在文件中的实际次序相一致。例如，要读取最后一个数据，也必须从第一个数据开始读起，然后依次读到最后一个数据。

2.随机文件。

随机文件是指随机存取的文件。随机文件是以记录为单位读写数据的，一个记录一般包含有多个数据项。例如，学生的个人信息就是一个记录，它由学号、姓名、性别和年龄等数据项组成。

随机文件中的每个记录都有一个记录号，在读写记录时，只要指出记录号就可以直接读写该记录了，而不需要依次读取它前面的各个记录。

按照文件存储数据的形式又可以将文件分成 ASCII 码文件和二进制文件。

1.ASCII 码文件。

ASCII 码文件是指文件中的数据是以 ASCII 码进行编码存储的。

2.二进制文件。

二进制文件是指文件中的数据是以二进制格式进行编码存储的，它允许用户以字节为单位进行数据的读写。这类文件的操作灵活性较大，但编程工作量也较大。

与文件读写有关的重要函数有 3 个：LOF, LOC 和 EOF.

1.LOF 函数。

语法：LOF (<文件号>)

LOF 函数用来返回指定文件的字节数，如果返回值为 0,则表示文件为空文件。

2.LOC 函数。

语法：LOC (<文件号>)。

LOC 函数用来返回已打开的文件中读写的位置。对于随机文件，它将返回最近读写的记录号；对于二进制文件，它将返回最近读写的字节的位置；对于顺序文件，LOC 函数的返回值在实际，扣没有什么用处。

3. EOF 函数。

语法：EOF (<文件号>)。

当文件指针到达文件末尾时，EOF 函数返回 True（真），否则返回 False（否）。

对于顺序文件，EOF 函数常常用来判断是否已到文件末尾。

VB6.0 基础教程 10.3.2 顺序文件的访问

在程序中访问顺序文件通常有 3 个步骤：打开、读取或写入、关闭。

1. 打开文件。

在对顺序文件进行操作之前，必须使用 Open 语句打开它，同时要告诉操作系统对文件进行什么操作。

Open 语句的一般形式如下：

```
Open <文件名> For <访问模式> [Access<操作类型>] [锁定]As[#]<文件号>  
[Len=<记录长度>]
```

其中各参数的含义如下：

文件名：是指要访问的顺序文件的名称以及文件所在的路径。

文件的访问模式是指要对文件进行什么操作，有以下 3 种模式：

OUTPUT:对文件进行写操作，即将数据写入文件。

INPUT:对文件进行读操作，即将数据从文件中读入内存。

APPEND:将数据追加到文件末尾。

操作类型用来规定对被打开文件所能进行的操作，有以下 3 种类型：

READ:只读。

WRITE:只写。

READWRITE:读写都可以。

锁定：用来规定是否允许其他进程对本次打开的文件进行访问，有以下 3 种情况：

Shared:共享，即允许其他进程对本次打开的文件进行读写操作。这也是默认设置。

Lock Read:禁止其他进程对本次打开的文件进行读操作。

Lock Write;禁止其他进程对本次打开的文件进行写操作。

Lock Read Write:禁止其他进程对本次打开的文件进行读写操作。

文件号：为被打开的文件指定一个文件号，在以后访问该文件时，以文件号来代表文件。文件号的取值为 1~511 之间的整数。每个文件号对应唯一的一个文件，不能将已使用的文件号再指定给其他文件。

记录长度：用来指定数据缓冲区的大小，其值范围为小于或等于 3767 的整数。

提示：如果以 **OUTUP** 访问模式或打开一个不存在的文件，则 **VB** 会自动创建一个相应的文件。

2.关闭文件。

在对打开的文件进行各种操作后，还必须将其关闭，否则会造成数据的丢失。

实际上，对顺序文件进行写操作时，将数据写入的是缓冲区，只有关闭文件时才将缓冲区中的数据全部写入文件。

使用 **Close** 语句可关闭文件，其形式如下：

Close[#文件号>][,<文件号>]...

文件号是指使用 **Open** 语句打开文件时为文件指定的文件号。如果在 **Close** 语句中忽略文件号，则会关闭所有已打开的文件。

例如：

Close #1 关闭 1 号文件

Close #2,#3,#4 关闭 2 号、3 号和 4 号文件

3.写操作。

如果顺序文件以 **OUTPUT** 模式或 **APPEND** 模式打开，就可以使用 **Write** 或 **Print** 语句向该文件写入数据。

(1) **Print** 语句

```
Open "e: \mytext.txt" For Output As #1
Print #1.Spc(5); "人员资料"
Print #1 空行
Print #1. "云 云" , "女" , "22"
Print #1. "郝 云" ; "男" , "23"
Print #1. "王 垠" , "女" , "23"
Print #1.
```

该代码被执行后，会在 E 盘新建一个 **mytext.txt** 文件，使用 Windows 自带的记事本可以查看该文件的内容及格式。如果

mytext.txt 文件已存在，则其原来的内容将被新的内容覆盖掉。这是因为打开文件的模式为 OUTPUT,如果以 APPEND 模式打开文件，则新内容会追加到原内容之后。

在实际编程中，经常要将文本框中的文本以文件的形式保存到磁盘_上，这时也可以使用 Print 语句来实现。

例如，下列代码是将文本框（Text1）中的内容保存到 D 盘上，文件名为 Test.txt.

```
Open "d:\Test.txt" For Output As #1  
Print #1,Text1.Text  
Close #1
```

（2）Write 语句。

语法：Write #<文件号>,[输出列表]

Write 语句与 Print 语句基本相同，各数据项之间以逗号隔开。区别在于 Write 语句以紧凑格式存放，且同时输出字符串上的双引号与数据项之间的逗号。

例如：

```
Print #1, "云 云","女","22"
```

写入文件的内容与格式为：云 云 女 22

```
Write #1," 云 云","女","22"
```

写入文件的内容与格式为：“云 云”，“女”，“22”

4.读操作。

读操作是指将文件中的数据读取到变量中。如果文件以 INPUT 模式打开，就可以使用 Input 语句、Line Input 语句或 Input 函数对文件进行读操作。

(1) Input 语句语法: Input #<文件号>,<变量列表>

变量用来存放从文件中读取的数据，各变量以逗号隔开，变量的类型应该与文件中所存储的数据类型一致。在读取数据后，各数据项分别存放在所对应的变量中。

例如，如果存放在 E 盘中的 Test.txt 文件的内容及格式为：

云 云，女，22

王 垠，男，21

.....

执行下列代码可将该文件中的内容打印在窗体上。

```
Open "C: \Tect.txt" For Input As #1
Do while Not EOF(1)
Input #1,a,b,c 将 3 个数据项分别存放在 3 个变量
中。
Print a, b, c
Loop
Close #1
云 云 女 22
王 垠 男 21
.....
```

(2) Line Input 语句。

语法：Line Input#<文件号>，<变量>

该语句以行来读取数据，并存放在变量中，它不把逗号当作数据项的分界符。变量必须是字符串型或变体型。

例如，使用 Line Input 语句在窗体上打印 Test.txt 文件内容的代码如下：

结果如下

云 云，女，22

王 垠，男，21

(3) Input 函数。

语法：Input (n,#<文件号>) Input 函数可以从文件中读取指定个数的字符，其中字符中包括空格、逗号、双引号和回车符等。参数 n 用来指定要读取的字符个数。

例如，

Open “e:\Test.txt” For Input As #1

Print Input (8,#1) 在窗体上打印文件 Test.txt 中的前 12 个字符

Close #1

结果如下：

云 云，女，2

VB6.0 基础教程 10.3.3 随机文件的访问

随机文件的访问与顺序文件的访问一样，也是首先要打开文件，在访问完毕后要关闭文件。

1.打开与关闭。

随机文件的打开与关闭也是分别使用 **Open** 语句和 **Close** 语句来完成的，并且 **Close** 语句的使用与关闭顺序文件完全相同。

使用 **Open** 语句打开随机文件的一般格式如下：

Open<文件名>**For Random As** [#]<文件号>[**Len**=<记录长度>].

其中 **For Random** 表示打开随机文件，**Len** 用来指定记录的长度，记录长度的缺省值为 128 个字节。

例如：

Open E:\Student.dat For Random As #1 Len=?U.

打开 E 盘中的名称为 **Student.dat** 的随机文件。如果该文件不存在，则创建一个新的文件，且记录长度为 20.

2.写操作。

在随机文件打开后，可以使用 **Put** 语句向其中写入记录 **r** **Put** 语句的形式如下：

Put<#文件号>|<,记录号>|,<变量>.

Put 语句将记录变量的内容写入到所打开文件中指定的记录位置处。其中记录号是大于 1 的整数，表示写入的是第几条记录。如果忽略记录号，则在当前记录后插入新的记录。

一般来讲，一条记录包含有多项内容。例如，一个学生的记录可能包含学号、姓名、性别以及年龄等信息。这样，基本数据类型就不能满足要求，需要创建一个自定义的数据类型，或称为记录型数据类型。

用户可以通过 VB 提供的 **Type** 语句来创建自定义类型，它的一般形式为：

Type 类型名

元素名 **As** 类型

元素名 **As** 类型

元素名 **As** 类型

...

End Type

例如，定义一个名为 **Student** 的类型，其中包括学号、姓名、性别以及年龄等信息。

Type Student

Sno As Integer

Sname As String*10

Ssex As String*2

Sage As integer

End Type

在定义了 **Student** 类型后，就可以将变量声明为 **Student** 类型，例如：

Dim stu As Student

该语句将变量 **Stu** 声明为 **Student** 类型。它包括 4 个成员，在程序中可以用“变量.元素”的形式来引用各成员，例如：

Stu.Sno=1 给元素 Sno 赋值

Stu.Sname = “齐小莉” 给元素 Sname 赋值

显然,使用变量 Stu 就可以为随机文件添加包含多项内容的记录。

例如:

Open E:\Student.dat For Random As #1 Len=Len (Stu)

put #1,1,stu 将变量 stu 中的内容写入到记录 1 中

Close #1

3.读操作。

对随机文件的读操作使用 Get 语句,其形式如下:

Get <#文件号>,[<记录号>],<变量>

Get 语句将随机文件中的指定记录读取到变量中,如果忽略记录号,则读取当前一记录后的那一条记录。

实例 10.3 访问随机文件。

编写一个学生档案管理小程序,当学生信息输入后将被保存到文件中,以后可以读出输入的学生信息。

在窗体上放置 6 个标签控件、5 个文本框控件和 2 个按钮控件,如图 10.8 所示。其中各主要对象的属性设置如表 10.5 所示。

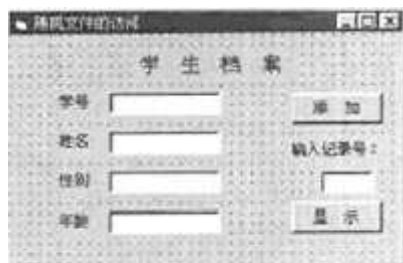


图 10.8 实例 10.3 的窗体设计

表 10.5 实例 10.3 中各对象的属性设置

对象	属性	值
窗体	Caption	随机文件的访问
学号文本框	名称	TexNo
	Text	置空
姓名文本框	名称	TexName
	Text	置空
性别文本框	名称	TexSex
	Text	置空
年龄文本框	名称	TexAge
	Text	置空
输入记录号文本框	名称	TexShow
	Text	置空
按钮 1	名称	ComAdd
	Caption	添加
按钮 2	名称	ComShow
	Caption	显示

```

Private Type student

    Sno As Integer

    Sname As String * 10

    Ssex As String *2

    Sage As Integer

End Type

Dim Stu As student

Private Sub ComAdd_Click()

    Stu.Sno =Val(TexNo.Text)

    Stu.Sname=TexName.Text

    Stu.Ssex=TexSex.Text

    Stu.Sage- Val (TexAge.Text)

    Open "E: \Student.dat" For Random As #1

    Len=Len(Stu)

    RecordNo=LoF (1) /Len(Stu)+1

    Put #1,RecordNo,Stu

```

```
Close #1

End Sub

Private Sub Comshow_Click()

RecordNo=val(TexShow.Text)

Open "E: \student.dat" For Random As #1 Len
=Len(stu)

Get #1,RecordNo,Stu

Close #1

TexNo.Text=Stu.Sno

TexName.Text=Stu.Sname

TexSex.Text=Stu.Ssex

TexAge.Text=Stu.Sage

End Sub
```

运行该程序，在表单中输入学生的信息，单击【添加】按钮即可将其追加到 E 盘的随机文件 Student.dat 中。在【输入记录号】文本框中输入记录号，单击【显示】按钮即可在表单中显示出该记录的内容。

VB6.0 基础教程 10.3.4 进制文件的访问.

二进制文件的访问与随机文件的访问很类似，不同的是随机文件是以记录为单位进行读写操作的，二进制文件则是以字节为单位进行读写操作的。

1.打开与关闭文件

打开二进制文件的 Open 语句如下：

`open<文件名>For Binary As <#文件号>.`

二进制文件的关闭操作与其他文件的关闭操作相同。

2.写操作。

在二进制文件打开后，可以使用 **Put** 语句对其进行写操作，**Put** 语句的形式如下：

`Put<#文件号> [<位置>],<变量>.`

Put 语句将变量的内容写入到所打开文件中指定的位置处，它一次写入的长度等于变量的长度。例如，若变量为整型，则写入 2 个字节的数据。如果忽略位置参数，则表示从文件指针所指的位置开始写入数据，数据写入后，文件指针会自动向后移动。在文件刚刚被打开时，文件指针指向第一个字节。

3.读操作。

可以使用 **Get** 语句或 **InPnt** 函数来读取二进制文件的数据，形式如下：

`Get<#文件号>,[位置],<变量>.`

`Input (<字节数>,<#文件号>).`

Input 函数返回从文件中读取的指定个字节的字符串口。

第十一章 数据库编程技术

VB6.0 基础教程 11.1 数据库的基本概念

数据库的基本概念是进行数据库编程的基础。了解数据库，就是了解数据库的数据结构、文件组织方式以及数据库应用程序的基本框架。

数据库是一组特定数据的集合，是提供数据的基地。它能保存数据并允许用户访问所需的数据。数据库中保存的数据都是相关数据，为了便于保管和处理这些数据，将这些数据存入数据库时必须具有一定的数据结构和文件组织方式：

数据库中数据的组织形式有多种，最近几年来，关系模型已经成为数据库设计的事实上的标准，在关系数据库中，实际保存数据的数据结构是一个或多个表（Table），每个表定义了某种特定的结构。

下面介绍关系数据库中的一些基本概念。

1.表。

关系数据库中的数据集合用表来表示，表是它的基本组成单元。一个数据库由一个或多个表组成。

表实际上就是一个二维表格，例如，表 11.1 所示的是一个通讯录表，其中包含姓名、电话、手机、传呼和地址等通讯信息。

表 11.1 通讯录表				
姓名	电话	手机	传呼	住址
郝春强	62777076	13700217717	191-1227255	清华 9# 116
陈 伟	62779501	13801012453	191-1227263	清华 13# 310
孙仁莉	8630156	13908527229	191-5284366	贵州遵义
齐小莉	7612120	13709118637	127-5535360	陕西吴旗

表中每一个人的信息称为一个记录（Record），即表的每一行就是一个记录，而且，表中的记录必须是唯一的。

表中的每一列称作一个字段（Field），描述了它所含有的数据。创建一个数据库时，要为每个字段设置字段名、数据类型、最大长度等属性。字段中存放的数据可以是各种字符、数字或者图形。同样，表中的字段也应该是唯一的。

2.主关键字。

每个表都应该有一个主关键字，它是记录的唯一标识符。例如，在学生管理数据库中，可以将学号作为主关键字。对于每个记录来说，主关键字必须具有一个唯一的值，即主关键字不能为空值：

3.索引。

数据库建成之后，为了便于查找，可以在数据库中建立索引来加快查找速度。数据库的索引与书的目录索引很类似，通过索引就能很快找到所需的内容。

VB6.0 基础教程 11.2 Visual Basic 数据库系统

Visual Basic 数据库系统由 3 部分组成：用户界面、数据库引擎和数据仓库。其中数据库引擎存在于用户界面和数据仓库之间，起着中介作用，用户通过它与要访问的特定数据库相连。对于 VB 所支持的任何数据库格式，所用的数据库编程技术都是相同的。

下面简单介绍数据库的这 3 个组成部分：

（1）用户界面。

用户界面是进行人机交互的界面，用于查看、显示数据或更新数据。驱动用户界面窗体的是用 Visual Basic 编写的代码，这些代码使得用户的操作能作用到数据库上，如添加或删除记录、执行查询等。

（2）数据库引擎。

Visual Basic 缺省的数据库引擎是 Microsoft Jet 数据库，它包含在一组动态链接库（DLL）中，运行时，这些动态链接库被链接到 Visual Basic 程序。数据库引擎的作用是把应用程序的请求翻译成对数据库的物理操作。

(3) 数据仓库。

数据仓库是包含数据库表的一个或多个文件。Visual Basic 支持多种数据库，默认的数据库是 Microsoft Access 数据库，即.mdb 文件。

VB6.0 基础教程 11.3 用数据管理器建立数据库

建立数据库的方式很多，用户既可以使用专门的数据库应用程序，如 Microsoft Access，来创建数据库，也可以使用 VB 自带的可视化数据管理器来创建和管理数据库。可视化数据管理器是 VB 提供了一种极为方便的数据库设计工具，具有创建数据库、设计与编辑表格等功能。



图 11.1 可视化数据管理器

执行【外接程序】菜单下的【可视化数据管理器】命令即可打开可视化数据管理器，如图 11.1 所示。

使用可视化数据管理器可以创建 Microsoft Access， Dbase， Foxpro， Paradox， ODBC 等多种数据库。本节将创建一个名称为 TelBook 的通讯录数据库，其中只包含一个表，名称为 TelTable，其结构如表 11.1 所示。

1.创建数据库。

打开可视化数据管理器的【文件】菜单，指向【新建】子菜单。在【新建】级联菜单中依次选择 Microsoft Access， Version 7.0 MDB，则打开如图 11.2 所示的对话框。



图 11.2 输入数据库文件名对话框

在该对话框中确定要创建的数据库的文件名和其存储路径，这里选择路径为 E:\，文件名为 TelBook.单击【保存】按钮，则在可视化数据管理器中出现了新建的数据库窗口，其中列出了新建的数据库的一些属性，如图 11.3 所示。

2.创建表。

此时的数据库仅仅是个空壳，除了路径有效外，没有实际内容。还需要进一步为数据库建立数据表。

在数据库窗口中单击鼠标右键，在弹出的快捷菜单中执行【新建表】命令，即可打开用于创建表的【表结构】对话框，如图 11.4 所示。



图 11.3 数据库窗口



图 11.4 【表结构】对话框

【表名称】文本框中输入数据表的名称为 TelTable，接下来就是为表添加字段。单击【添加字段】按钮，则弹出如图 11.5 所示的对话框。



图 11.5 【添加字段】对话框

在【名称】文本框中输入字段名，同时设置字段的类型及大小等选项。单击【确定】按钮即可创建出一个字段，它将出现在【表结构】对话框的【字段列表】框中。接着可以创建下一个字段。表 11.2 中列出了【添加字段】对话框中主要选项的含义。

表 11.2 【添加字段】对话框中主要选项的含义

选项	含义
名称	用来输入字段名
类型	用来设置字段的类型，以决定字段中所存储的数据类型
大小	设置字段的最大长度（字节数）
固定字段	如果选中该单选按钮，则所输入的字段必须为定长
可变字段	如果选中该单选按钮，则所输入的字段可以不定长
允许零长度	验证输入字段的数值时所使用的简单规则
必需的	选中该复选框表示该字段必须输入，不能省略
顺序位置	用来决定字段在表中的顺序位置
验证文本	当输入无效数据时显示的提示信息
缺省值	生成记录时字段的初始值

注意：不能使所有的字段都允许为零长度，在一个表中至少要有
一个字段不能为空，这个字段可以作为主关键字，

这里创建 5 个字段，分别是姓名、电话、手机、传呼和地址，类
型均设置为 Text，并且长度是可变的，除姓名字段为“必要的”外，其
余的字段都设置为“允许零长度”。

添加了字段后的【表结构】对话框如图 11.6 所示。



图 11.6 为表添加了字段

3. 添加索引。

数据库中表的索引不是必须的，但是它能大大加快查询的速度。
索引字段一般要选择字段值唯一的字段，而且该字段不能为空值。这
里将“姓名”作为表 TelTable 的索引。

单击【表结构】对话框上的【添加索引】按钮，则弹出如图 11.7
所示的对话框，在【名称】文本框中输入索引名，同时从【可用字段】
列表选定用作索引的字段，这里选择“姓名”字段，并且选中【主要
的】和【唯一的】复选框，这样就将姓名指定为主关键字，在输入记
录时，如果输入的姓名相同，将会出错。单击【确定】按钮返回到【表

结构】对话框中，单击【生成表】按钮即可完成数据表的创建，在数据库窗口会出现新创建的表。

用户还可以随时编辑所创建的表，方法是在数据库窗口的表名称 TelTable 上单击右键，执行弹出的快捷菜单中的【设计】命令即可打开[表结构]对话框，从中可进行添加或删除字段的操作。需要指出的是，在【表结构】对话框中不能更改字段的类型、大小等设置，不过，用户可以先将其删除，然后重新添加一个新的字段。

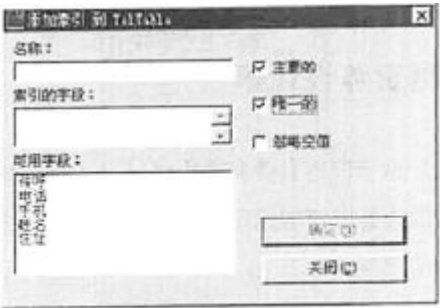


图 11.7 【添加索引】对话框

4.输入记录。

在完成了数据表 TelTable 的创建后，就可以向表中输入记录了。输入的方法可以是以 Data 控件模式、无 Data 控件模式或 DBGrid 控件模式。这里采用 Data 控件模式为数据表 TelTable 输入记录，具体步骤如下：

- （1）单击数据管理器工具栏中的【表类型记录集】按钮和 Data 控件按钮，这表明以 Data 控件模式向表中添加记录。
- （2）在数据库表上单击右键，执行弹出的快捷菜单中的【打开】命令，则打开添加记录对话框，如图 11.8 所示。



图 11.8 添加记录对话框

(3) 输入记录后单击【更新】按钮，则弹出消息框，提示用户是否保存新记录，单击【是】按钮即可将记录添加到表中。

(4) 单击【添加】按钮可以输入下一个记录。

VB6.0 基础教程 11.3.1 创建 Access 格式数据库

使用可视化数据管理器可以创建 Microsoft Access， Dbase， Foxpro， Paradox， ODBC 等多种数据库。本节将创建一个名称为 TelBook 的通讯录数据库，其中只包含一个表，名称为 TelTable，其结构如表 11.1 所示。

1.创建数据库。

打开可视化数据管理器的【文件】菜单，指向【新建】子菜单。在【新建】级联菜单中依次选择 Microsoft Access， Version 7.0 MDB， 则打开如图 11.2 所示的对话框。



图 11.2 输入数据库文件名对话框

在该对话框中确定要创建的数据库的文件名和其存储路径，这里选择路径为 E:\， 文件名为 TelBook.单击【保存】按钮，则在可视化

数据管理器中出现了新建的数据库窗口，其中列出了新建的数据库的一些属性，如图 11.3 所示。

2.创建表。

此时的数据库仅仅是个空壳，除了路径有效外，没有实际内容。还需要进一步为数据库建立数据表。

在数据库窗口中单击鼠标右键，在弹出的快捷菜单中执行【新建表】命令，即可打开用于创建表的【表结构】对话框，如图 11.4 所示。



图 11.3 数据库窗口



图 11.4 【表结构】对话框

【表名称】文本框中输入数据表的名称为 TelTable，接下来就是为表添加字段。单击【添加字段】按钮，则弹出如图 11.5 所示的对话框。



图 11.5 【添加字段】对话框

在【名称】文本框中输入字段名，同时设置字段的类型及大小等选项。单击【确定】按钮即可创建出一个字段，它将出现在【表结构】对话框的【字段列表】框中。接着可以创建下一个字段。表 11.2 中列出了【添加字段】对话框中主要选项的含义。

表 11.2 【添加字段】对话框中主要选项的含义

选项	含义
名称	用来输入字段名
类型	用来设置字段的类型，以决定字段中所存储的数据类型
大小	设置字段的最大长度（字节数）
固定字段	如果选中该单选按钮，则所输入的字段必须为定长
可变字段	如果选中该单选按钮，则所输入的字段可以不定长
允许零长度	验证输入字段的数值时所使用的简单规则
必需的	选中该复选框表示该字段必须输入，不能省略
顺序位置	用来决定字段在表中的顺序位置
验证文本	当输入无效数据时显示的提示信息
缺省值	生成记录时字段的初始值

注意：不能使所有的字段都允许为零长度，在一个表中至少要有一个字段不能为空，这个字段可以作为主关键字，

这里创建 5 个字段，分别是姓名、电话、手机、传呼和地址，类型均设置为 Text，并且长度是可变的，除姓名字段为“必需的”外，其余的字段都设置为“允许零长度”。

添加了字段后的【表结构】对话框如图 11.6 所示。



图 11.6 为表添加了字段

3.添加索引。

数据库中表的索引不是必须的，但是它能大大加快查询的速度。索引字段一般要选择字段值唯一的字段，而且该字段不能为空值。这里将“姓名”作为表 TelTable 的索引。

单击【表结构】对话框上的【添加索引】按钮，则弹出如图 11.7 所示的对话框，在【名称】文本框中输入索引名，同时从【可用字段】列表选定用作索引的字段，这里选择“姓名”字段，并且选中【主要的】和【唯一的】复选框，这样就将姓名指定为主关键字，在输入记录时，如果输入的姓名相同，将会出错。单击【确定】按钮返回到【表结构】对话框中，单击【生成表】按钮即可完成数据表的创建，在数据库窗口会出现新创建的表。

用户还可以随时编辑所创建的表，方法是在数据库窗口的表名称 TelTable 上单击右键，执行弹出的快捷菜单中的【设计】命令即可打开[表结构]对话框，从中可进行添加或删除字段的操作。需要指出的是，在【表结构】对话框中不能更改字段的类型、大小等设置，不过，用户可以先将其删除，然后重新添加一个新的字段。

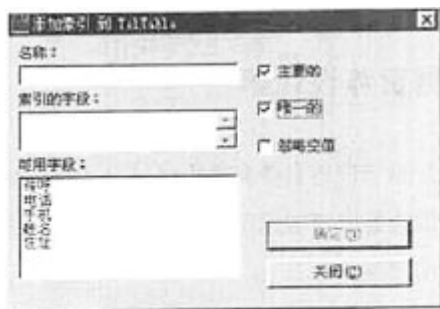


图 11.7 【添加索引】对话框

4.输入记录。

在完成了数据表 TelTable 的创建后，就可以向表中输入记录了。输入的方法可以是以 Data 控件模式、无 Data 控件模式或 DBGrid 控件模式。这里采用 Data 控件模式为数据表 TelTable 输入记录，具体步骤如下：

(1) 单击数据管理器工具栏中的【表类型记录集】按钮和 Data 控件按钮，这表明以 Data 控件模式向表中添加记录。

(2) 在数据库表上单击右键，执行弹出的快捷菜单中的【打开】命令，则打开添加记录对话框，如图 11.8 所示。



图 11.8 添加记录对话框

(3) 输入记录后单击【更新】按钮，则弹出消息框，提示用户是否保存新记录，单击【是】按钮即可将记录添加到表中。

(4) 单击【添加】按钮可以输入下一个记录。

VB6.0 基础教程 11.3.2 使用数据窗体设计器

可视化数据管理器自带有数据窗体设计器，使用它可以在最短的时间内设计出符合要求的数据库管理应用程序。本节将利用数据窗体设计器为新建的 TelBook 数据库设计用户窗体。

设计窗体之前首先要打开要设计的数据库。如果 TelBoo 数据库已经被关闭，可通过可视化数据管理器的【文件】菜单中的【打开数据库】选项打开它。

执行【实用程序】菜单中的【数据窗体设计器】命令，可打开【数据窗体设计器】对话框，在【窗体名称】文本框中输入要建立的窗体名称，在【记录源】下拉列表框中选择数据表 TelTable，则在【可用的字段】列表中列出表 TelTable 的所有字段，如图 11.9 所示。



图 11.9 【数据窗体设计器】对话框

单击按钮，将所有的字段传到【包括的字段】列表中，该列表中的字段将出现在窗体上。单击【生成窗体】按钮，就可以在当前工程中添加一个新的数据库窗体，如图 11.10 所示。该窗体采用 Data 控件作为数据源，文本框用来显示和添加记录。

可见，使用数据窗体设计器可以很容易地建立一个数据库应用程序。将该数据库窗体设置为启动窗体，运行程序后可以通过文本框浏览到数据库中的记录，也可以进行添加、删除、刷新以及更新记录等操作。

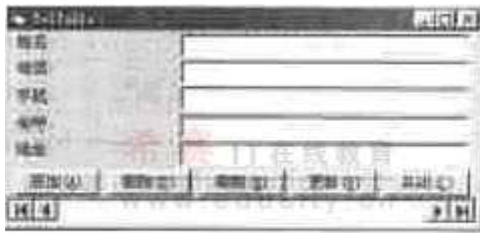


图 11.10 数据窗体设计器创建的窗体

VB6.0 基础教程 11.4 使用 Data 控件访问数据库

Data 控件是 VB 提供的内部控件，使用它可以不编写任何代码就可以对数据库进行访问，从而大大简化了数据库的编程。此外也可以把 Data 控件和 VisualBasic 代码及 SQL 语言结合起来创建完整的应用程序，为数据处理提供高级的编程控制。

在同一个窗体中可以同时使用多个 Data 控件，但是每个 Data 控件只能访问一个数据库。在设计阶段要为 Data 控件指定它所要访问的数据库，而且在运行其间不可以更改。

将 Data 控件放置在窗体上，其外观如图 11.11 所示，其中显示的文本由它的 Caption 属性决定，各按钮的功能如下：



图 11.11 Data 控件

1. Connect 属性。

Connect 属性用来指定数据库的类型，VB 支持的数据库类型众多，如 Microsoft Access、Excel、Foxpro 等。其中默认的数据库为 Access。单击 Connect 属性右边的向下箭头，可下拉出一个 Data 控件所支持的数据库类型列表，如图 11.12 所示。用户可从中选择要操作的数据库类型。



图 11.12 Connect 属性的下拉列表

2. DatabaseName 属性。

该属性用于设置或返回 Data 控件所使用的数据库的名称。

3. RecordSource 属性。

一个数据库中可能有多个表，RecordSource 属性用于指定 Data 控件所要操作的表。在设置了 DatabaseName 属性后，在 RecordSource 属性的下拉列表中会出现所选数据库中的所有表，用户可以选择一个表。有时 RecordSource 属性的值也可以不是一个完整的表，而是 SQL 查询语言的一个查询语句，这样，Data 控件可访问的数据将只是查询后的结果。

由 RecordSource 属性确定的具体可访问的数据构成一个记录集（Recordset），Recordset 是一个对象，它具有属性和方法，操作数据库其实就是使用该对象的方法，Recordset 对象的两个较重要的属

性是 B0F 和 EOF，当 BOF 属性的值为 True 时，表明当前位置位于第一个记录之前，当 EOF 属性的值为 True 时，表明当前位置位于最后一个记录之后，在操作数据库时，经常要使用这两个属性来判断是否已到达数据库的首录或末记录。

4. RecordsetType 属性。

该属性用来设置记录集的类型。记录集共有 3 种类型，分别是 Table（表）、Dynaset（动态集）和 Snapshot（快照）。

Table 类型是以表格直接显示数据，需要系统资源最多，但是其处理速度最快。

Dynaset 类型的记录集可以在表中增加、修改和删除记录，是最灵活的，种记录集类型。Snapshot 类型的记录集只能静态显示数据（只读），其灵活性最低，但是所需的系统资源最少。

5. Exclusive 属性。

该属性的功能是决定 Data 控件所链接的数据库文件在运行时是否允许其他进程将它打开。若该属性的值为 True，则表明不允许其他进程打开该数据库。

6. ReadOnly 属性。

该属性用来决定是否能够通过数据绑定控件来编辑数据库中的记录内容。该属性的默认值为 False，表明用户可以通过数据绑定控件编辑数据库中记录的内容。

7. BOFAction 与 EOFAction 属性。

BOFAction 与 EOFAction 属性用来决定当记录移动超出起点或结束时，程序要执行的操作。它们的取值及含义如表 11.3 所示。

表 11.3 BOFAction 与 EOFAction 属性的取值及含义

属性	取值	操作
BOFAction	0	重定位到第一个记录
	1	移过记录集的开始位置，定位到一个无效位置，且触发 Date 控件的 Validate 事件
EOFAction	0	重定位到最后一个记录
	1	移过记录集的结束位置，定位到一个无效位置，且触发 Date 控件的 Validate 事件
	2	自动执行 Date 控件的 AddNew 方法向记录集加入新的空记录

VB6.0 基础教程 11.4.2 数据绑定控件

Data 控件能够操作数据库，但它本身不能显示数据库中的数据，在编写数据库应用程序时，还需要借助其他控件，如文本框等控件来显示数据。这就需要将文本框等控件与 Data 控件关联起来使之成为 Data 控件的数据绑定控件。

VB 中能够显示数据的控件基本都提供了数据绑定，例如，文本框、标签、图片框等都可以作为数据绑定控件。VB 还提供了专门的数据绑定控件，如 DBGrid（数据网格）、DBList（数据列表框）、DBCombo（数据组合框）等。

可以说，Data 控件是 VB 和数据库之间的联系桥梁，而数据绑定控件则把 Data 控件和用户界面联系起来，两者构成了 VB 开发数据库的主体。

可以通过设置控件的以下两个属性来使它成为 Data 控件的数据绑定控件：

1. DataSource 属性。

该属性用来指定要与控件绑定的 data 控件。在【属性】窗口中选中该属性，然后单击其右边的向下箭头按钮，可在下拉列表中选择当前可用的 Data 控件。

2. DataField 属性。

该属性用来设置控件对应的数据库字段。在设置了 DataSource 属性后，DataField 属性的下拉列表中，将列出可用的字段。

实例 11.1 显示 TelBook 数据库中的数据。

在该实例中，将创建一个简单的数据库应用程序，可以用它来浏览 TelBook 数据库中的记录。

在窗体上放置 5 个标签、5 个文本框和 1 个 Data 控件，如图 11.14 所示。其中各对象的属性设置如表 11.4 所示。



图 11.14 实例 11.1 的窗体设计

表 11.4 实例 11.1 中各对象的属性设置

对象	属性	值
窗体	Caption	显示记录
Data 控件	名称	Data1
	DataBaseName	E:\TelBook.mdb
	RecordSource	TelTable
标签 3	Caption	手机
标签 4	Caption	传呼
标签 5	Caption	住址
文本框 1	DataSource	Data1
	DataField	姓名
文本框 2	DataSource	Data1
	DataField	电话
文本框 3	DataSource	Data1
	DataField	手机
文本框 4	DataSource	Data1
	DataField	传呼
文本框 5	DataSource	Data1
	DataField	住址

用户不需要编写任何代码，运行程序，一单击 Data 控件的 4 个箭头按钮，即可浏览整个数据库中的记录，如图 11.15 所示。

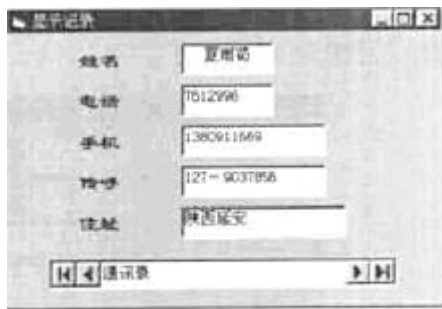


图 11.15 实例 11.1 的运行效果

VB6.0 基础教程 11.4.3 Data 控件的常用方法

使用 Data 控件不仅可以浏览数据库中的记录，还能编辑数据库中的记录，这些可以通过 Data 控件的方法来实现。

1. 与浏览有关的方法。

可以使用 Data 控件的箭头按钮来浏览记录，也可以使用 Data 控件的 Move 方法来操作。表 11.5 中列出了 Data 控件的 5 个 Move 方法。

表 11.5 Data 控件的 5 个 Move 方法

方法	功能
MoveFirst	移动至第一个记录
MoveLast	移动至最后一个记录
MoveNext	移动至下一个记录
MovePrevious	移动至上一个记录
Move (n)	向前或向后移动 n 个记录

2. 与查询有关的方法。

使用 Find 方法可在数据记录集中查找到与指定条件相符的一个记录,并使之成为当前记录。表 11.5 中列出了 Data 控件的 4 个 Find 方法。

表 11.6 Data 控件的 4 个 Find 方法

方法	功能
FindFirst	找到满足条件的第一个记录
FindLast	找到满足条件的最后一个记录
FindNext	找到满足条件的下一个记录
FindPrevious	找到满足条件的上一个记录

这 4 种 Find 方法的语法格式相同,例如:

查找记录集中姓名为夏雨荷的第一条记录

Data1.Recordset.FindFirst “姓名=夏雨荷”

查找记录集中姓名为夏雨荷的下一条记录

Data1.Recordset.FindNext “姓名=夏雨荷”

如果条件部分的常数来自于变量,例如由用户在文本框中输入,则条件表达式应该按以下格式书写:

Data1.Recordset.FindNext “姓名=” & “'” & text1.text &

除了普通的关系运算符外,还可以使用 Like 运算符来查找匹配某个模式的记录。例如,要查找住址中带“陕西”的记录,可以用以下语句:

Data1.Recordset.FindFirst“住址 Like 陕西”

在调用 Find 方法时，如果查找到符合条件的记录，则将 Data 控件定位到这个记录，并将 NoMatch 属性的值设置为 False 如果没有找到符合条件的记录，则将 NoMatch 属性的值设置为 True.

例如，可以使用下面代码来告诉用户没有找到所要查找的记录：

```
Data1.Recordset.FindFirst “传呼” & “'” & text1.text &
```

```
If Data.RecordSet.NoMatch Then
```

```
MsgBox “记录不存在”， 64, “64”
```

```
End If
```

3.与编辑有关的方法。

(1) AddNew 方法。

该方法将当前记录指向缓冲区，从而可以添加新的记录。

(2) Update 方法。

该方法在修改或添加记录后将数据从缓冲区读入数据库。单击 Data 控件的箭头按钮、将自动调用 Update 方法。在调用 AddNew 方法添加新记录后，必须调用 Update 方法来保存新添加的记录，否则所添加的记录无效。

(3) Delete 方法。

该方法用于删除当前的记录。在删除一条记录后，它并不会自动从数据绑定控件中消失，因此，需要调用 Refresh 方法来刷新记录集，以反映最新的变化。

实例 11.2 创建通讯录。

本实例是对实例 11.1 的完善，将 Data 控件的 Visible 属性设置为 False，使之在运行时不可见，而以按钮来操作数据库，使得界面更友好。并且在该程序中能执行添加、删除与查找记录的操作。

在如图 11.14 所示的窗体上再放置 5 个按钮控件、1 个组合框控件和 1 个文本框控件，如图 11.16 所示，各控件的属性设置如表 11.7 所示。

表 11.7 实例 11.2 中控件的属性设置

对象	属性	设置
窗体	Caption	通讯录
按钮 1	名称	ComAdd
	Caption	添加
按钮 2	名称	ComDel
	Caption	删除
按钮 3	名称	ComFind
	Caption	查询
按钮 4	名称	ComPrev
	Caption	上一个
按钮 5	名称	ComNext
	Caption	下一个
组合框	名称	CobFind
	Style	2
文本框	名称	TexFind
	Text	置空

编写代码如下：

```

Private Sub Form_Load()

    CobFind.AddItem "姓名"

    CobFind.AddItem "电话"

    CobFind.AddItem "住址"

    CobFind.AddItem "手机"

    CobFind.AddItem "传呼"

    CobFind.AddItem "姓名"

End Sub

```

添加记录:

```
Private Sub ComAdd_Click()  
  
If ComAdd.Caption = "确定" Then  
  
On Error GoTo errorhandler  
  
Data1.UpdateRecond  
  
Data1.Recordset.MoveLast  
  
ComFrev.Enabled = True  
  
ComNext.Enabled = True  
  
ComDel.Enabled = True  
  
ComFind.Enabled = True  
  
ComAdd.Caption = "添加"  
  
Else  
  
Data1.Recordset.AddNew  
  
ComAdd.Caption = "确定"  
  
ComFrev.Enabled = False  
  
ComNext.Enabled = False  
  
ComDel.Enabled = False  
  
ComFind.Enabled = False  
  
End If  
  
Exit Sub
```

删除记录:

```
Private Sub ComDel_Click()
```

```

Dim i As Integer

i=MsgBox(“真的要删除当前记录吗?”, 52, “警告”)

If i=6 Then

Data1.Recordset.Delete

Data1.Refrst

End If

End Sub

下一个

Private Sub comnext_Click()

Data1.Recordset.MoveNext

ComPrev.Enabled=True

If Data1.Recordset.EOF Then

Data1.Recordset.MoveLast

ComNext.Enabled=False

End If

End Sub

上一个

Private Sub comprev_Click()

Data1.Recordset.MovePrevious

ComNext.Enabled=Ture

If Data1.Recordset.BOF Then

```

```
Datal.Recordset.MoveFirst
```

```
Comprev.Enabled=False
```

```
End If
```

```
End Sub
```

查询:

```
Private Sub ComFind_Click()
```

```
If TextFind.Text= “ ” Then
```

```
MsgBox “请输入查询内容!”, 48, “提示”
```

```
Exit Sub
```

```
End If
```

```
If CobFind.Text= “姓名” Then
```

```
Datal.Recordset.FindFirst “姓名=& ” ’ ”&  
TexFind.Text &” ”
```

```
Else IfCobFind.Text= “电话” Then
```

```
Datal.Recordset.FindFirst “电话=& ” ’ ”&  
TexFind.Text &” ”
```

```
Else IfCobFind.Text= “住址” Then
```

```
Datal.Recordset.FindFirst “住址=& ” ’ ”&  
TexFind.Text &” ”
```

```
Else IfCobFind.Text= “手机” Then
```

```
Datal.Recordset.FindFirst “手机=& ” ’ ”&  
TexFind.Text &” ”
```

```
Else IfCobFind.Text= “寻呼” Then  
    Datal.Recordset.FindFirst “传呼=& ” ’ ”&  
    TexFind.Text &” ”  
End If  
  
End Sub
```

运行该程序，窗体如图 11.17 所示，通过单击【上一个】按钮和【下一个】按钮可遍历所有记录，并且当移动到首记录或末记录后，相应的按钮会自动变为无效。

单击【添加】按钮后，界面如图 11.18 所示，用户可以在各个文本框中输入新的记录，输入完成后，单击【确定】按钮即可将该记录添加到数据表中。如果输入的姓名在数据库的记录中已存在，则弹出消息框提示用户记录已存在，不能添加该记录。

单击【删除】按钮可将当前的记录删除，为了防止误删掉某个记录，在删除记录前，会弹出一个消息框询问用户是否真要删除当前记录。单击【是】按钮将记录删除，单击【否】按钮则取消删除。

在该程序中，可以通过任何一种信息来查找记录。首先，在【查询】按钮下方的下拉列表框中选择一种查询方式，然后在该下拉列表框下面的文本框中输入相应的信息，单击【查询】按钮即可找到与查询条件相符的记录。例如，在组合框中选择姓名，在文本框中输入“郝春强”，单击【查询】按钮通讯录即显示姓名为“郝春强”的信息：如果不存在与查询条件相符的记录，则弹出消息框告知用户记录不存

在。在查询记录前如果在文本框中没有输入任何内容，也会弹出消息框提示用户。



图 11.17 通讯录

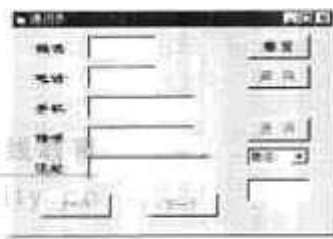


图 11.18 添加记录

VB6.0 基础教程 11.4.4 DBGrid(数据网格)控件

在上例中，每次只可以浏览一条记录，如果要同时在窗体上显示多条记录，则可采用 VB 提供的 DBGrid 控件。DBGrid 控件可增强程序的功能和灵活性，它不仅可以显示记录，还具有编辑功能，例如，进行添加、修改、删除一记录等操作。

DBGrid 控件是一个 ActiveX 控件，在【部件】对话框中选择添加 Microsoft Data Bound Grid Control 选项，即可将该控件添加到工具栏中。

DBGrid 控件的主要属性如下：

(1) DataSource 属性。

该属性用来设置所要绑定的 Data 控件。

(2) AllowAddNew, AllowDelete 和 AllowUpdate 属性这几个属性分别用来决定是否可以添加、删除与更新记录，值为 True 表示可以。AllowAddNew 和 AllowDelete 属性的默认值为 False 而，AllowUpdate 属性的默认值为 True。

下面将 Data 控件和 DBGrid 控件一起使用，来构造一个用于访问数据库 TelBook 的用户界面。

实例 11.3 使用 DBGrid 控件。

将 Data 控件和 DBGrid 控件放置在窗体上，如图 11.19 所示。各对象的属性设置如表 11.8 所示。

表 11.8 实例 11.3 中各对象的属性设置

对象	属性	值
窗体	Caption	使用 DBGrid 控件
Data 控件	名称	Data1
	DatabaseName	E:\TelBook.mdb
	RecordSource	TelTable
DBGrid 控件	Caption	通讯录
	DataSource	Data1
	AllowAddNew	True
	AllowDelete	True

运行该程序，窗体如图 11.20 所示，可见，数据库 TelBook 中的记录出现在 DBGrid 控件中，通过拖动滚动条即可浏览所有记录。将鼠标指针移动到 DBGrid 控件的列或行上，当鼠标指针变成双向箭头后，拖动鼠标可以改变列或行的宽度。

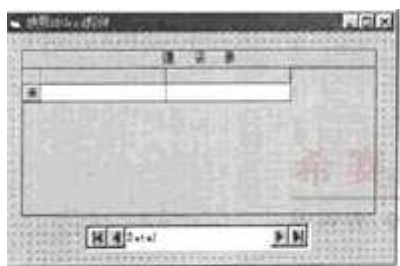


图 11.19 实例 11.3 的窗体设计



图 11.20 实例 11.3 的运行效果

记录的最后一行以星号 (*) 开头，表示在该行可以输入新的记录。在某行的左边单击可选中该行（选中的行以高亮度显示），按 Del 键即可将该条记录删除。