可视化编程应用——Visual Basic 6.0

杜秋华
康慧芳 编著

人 民 邮 电 出 版 社

图书在版编目(CIP)**数据**

可视化编程应用——Visual Basic 6.0/**杜秋华，康慧芳编著**. **—北京：人民邮电出版社，**2004.6
（中等职业学校计算机系列教材）
ISBN 7-115-12157-5

Ⅰ. **可**… **Ⅱ**.**①杜**…**②康**… **Ⅲ**. BASIC **语言—程序设计** **Ⅳ**. TP312
中国版本图书馆 CIP **数据核字（**2004**）第** 046369 **号**

内容提要

本书以 Visual Basic 6.0 **为蓝本，详细介绍了使用** Visual Basic 6.0 **进行可视化编程的基础知识和操作方法，帮助学生建立起可视化编程的思想，培养使用可视化编程语言进行程序设计的能力。**

全书共分为 3 **个部分，共** 12 **章，内容主要包括** Visual Basic 6.0 **的安装、启动、集成开发环境、编程基础、标准控件的使用、菜单的设计、对话框的使用、图形处理、高级界面的设计、文件管理、数据库编程、程序维护与调试等。在每章的最后均设有习题，使学生能够巩固本章所学知识。**

本书适合作中等职业学校“可视化编程应用”课程的教材，也可作为 Visual Basic 6.0 **初学者的自学参考书。**

可视化编程应用——Visual Basic 6.0

◆ **编　　著　杜秋华　康慧芳**
责任编辑　王文娟

◆ **人民邮电出版社出版发行　北京市崇文区夕照寺街** 14 **号**
邮编　100061　**电子函件**　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
读者热线　010-671940942
北京汉魂图文设计有限公司制作
北京顺义振华印刷厂印刷
新华书店总店北京发行所经销

◆ **开本：**787×1092　1/16
印张：18.5
字数：441 **千字**　　2004 **年** 6 **月第** 1 **版**
印数：1—　000 **册**　　2004 **年** 6 **月北京第** 1 **次印刷**

ISBN 7-115-12157-　/TP・

定价：24.00 **元**

前　言

本书是为中等职业学校计算机及应用专业所编写的配套教材，根据教育部 2001 年颁布的《中等职业学校计算机及应用专业可视化编程应用课程教学基本要求》编写，同时参考了《全国计算机信息高新技术考试技能培训和鉴定标准》中的程序员级考试大纲。

本书面向中等职业学校的学生，在内容的安排上尽量贴近应用，在叙述上尽量做到通俗易懂，全书循序渐进地向学生讲授如何使用 Visual Basic 6.0 进行可视化编程。既方便教师讲授，又便于学生理解掌握。

本书以 Visual Basic 6.0 为蓝本，详细介绍了使用 Visual Basic 6.0 进行可视化编程的基础知识、操作方法，帮助学生建立起可视化编程的思想，熟练掌握可视化编程的方法。

本书分 3 部分，共 12 章：

- 第 1～4 章为基础概念部分，主要介绍 Visual Basic 6.0 的安装、启动、集成开发环境、编程基础等基础知识。
- 第 5～8 章为简单应用部分，详细地论述了 Visual Basic 6.0 在可视化编程中的简单应用，包括标准控件的使用、菜单的设计、对话框的使用、图形处理等内容。
- 第 9～12 章为高级应用部分，主要讲解如何使用 Visual Basic 6.0 来开发一些复杂的程序，主要包括高级界面的设计、文件管理、数据库编程、程序维护与调试等内容。

书中各章都配有相应的习题，以利于学生对本章内容的巩固。为了便于教学，每章还给出了大量的实例，让学生在学习基本理论知识的同时，能够提高动手实践能力。本书另外配有一本《可视化编程应用 Visual Basic 6.0 上机实验指导与练习》，使学生能够通过上机实验真正掌握相关知识点，具备使用可视化编程语言编写简单应用程序的能力。

教师一般可用 30 个学时来讲解本教材内容，然后配合《可视化编程应用——Visual Basic 6.0 上机实验指导与练习》一书，辅以 42 个学时的上机时间，即可较好地完成教学任务，总的讲课时间约为 72 个课时。两本教材中的内容可以满足扩充至 120 课时所需的内容要求，教师在实际授课过程中可以根据需要对学时进行适当的调整。

本书适合中等职业学校计算机及应用专业以及其他相关专业使用，也可作为各类计算机培训学校的教学用书以及全国计算机信息高新技术程序员级考试的辅导用书，还可供计算机爱好者参考使用。

由于作者水平有限，疏漏之处敬请各位老师和同学指正。

作者

2004 年 4 月

目　　录

第1章　Visual Basic 6.0 概述

本章主要介绍可视化编程的基本概念和 Visual Basic 6.0 的基本知识，包括 Visual Basic 6.0 的特点，Visual Basic 6.0 的安装和启动，组件的添加和删除等，从而对 Visual Basic 6.0 有一个初步的认识。

本章学习目标

- 可视化编程的概念和特点。
- Visual Basic 6.0 的特点。
- Visual Basic 6.0 的安装和启动。
- 添加、删除 Visual Basic 6.0 的组件。

1.1 可视化编程概述

在可视化编程出现之前的程序设计中，基本上采用传统的编制程序代码的方式来设计用户图形界面，不仅需要大量的程序代码，而且在程序设计过程中看不到界面显示的效果，只有在程序执行时才能观察到。当界面效果不好时还须回到程序中去修改。可视化编程则通过调用控件，并为控制对象设置属性，根据开发者的需要，直接在窗口中进行用户界面的布局设计，该项技术具有编程简单、自动生成程序代码、效率高的优点，因而在当今的编程语言中被广泛采用。

可视化技术是当前发展迅速并引人注目的技术之一，它的特点是把原来抽象的数字、表格、功能逻辑等用直观的图形、图像的形式表现出来。可视化编程是它的重要应用之一。

在了解可视化编程前，必须先了解可视化编程的一些基本概念。

(1) 对象（Object）

事物都可称作对象，比如计算机、鼠标就是对象，在 Visual Basic 里对象主要分为两类：窗体（Form）和控件（Control）。

- 窗体（Form）：窗体或称表单，其实指的就是 Window。
- 控件（Control）：控件，指的是各种按钮、选项卡、对话框等。

(2) 属性（Property）

指的是对象所具有的特征，比如把一个人做作为一个对象，那么姓名、身高、体重都是这个对象的属性。

在 Visual Basic 6.0 中，一个按钮有 Caption、Name、Font 等属性。可通过设置对象的属性来改变对象的外观。

有两种方法可以修改对象的属性：

- 在对象的属性窗口中找到相应的属性进行设置。
- 在程序代码中通过编程设置。设置方法为：

  对象名 . 属性名＝属性值

(3) 事件（Event）

事件是发生在对象上的动作。比如搬桌子是一个事件，它是发生在桌子这个对象上的一个动作。比如 Load 是发生在 From 控件上的事件。不同的对象能识别不同的事件。事件的发生不是随意的，某些事件仅发生在某些对象上而已，比如“考试作弊被抓住”可以发生在学生这个

对象上，但它不会发生在老师这个对象上。

每个对象能识别的只是一组预先定义好的事件。但并非每个事件都会产生结果。比如说：把一个学生作为一个对象，如果产生的是“手被烫的事件”，那么会自动产生缩手这一结果。但是如果“考试作弊被抓住”，如果学校没有任何处罚的规定，那么这个学生就会“逍遥法外”，不会发生任何事情。但是如果为这一事件编写一个制度：“考试作弊被抓住罚抄 100 遍笔记”。所以需要为大部份事件编写特定的程序（事件响应函数）。

(4) 方法（Method）

方法是对“Method”的直译，是一个较难理解的概念，它是对象本身内含的函数或过程，它也是一个动作（不受外界的影响也可以产生的动作），但不称作事件，在 Visual Basic 里，方法和事件是这样的：

事件：

```
Private Sub  对象名_事件名
      （事件响应代码）
End Sub
```

方法：

```
对象名．方法名
```

例如对于人这个对象，可以有很多“方法”，跑步、游泳等，方法是一个简单的、不必知道细节的、无法改变的事件，同样，方法也不是随意的，某些对象有一些特定的方法。

所谓可视化编程，就是指在软件开发过程中，用直观的具有一定含义的图标按钮、图形化的对象取代原来手工的抽象的编辑、运行、浏览操作，软件开发过程表现为鼠标单击按钮和拖放图形化的对象以及指定对象的属性、行为的过程。这种可视化的编程方法易学易用，而且大大提高了编程效率。

1.2 Visual Basic 6.0 的特点

本节将简单介绍 Visual Basic 6.0 的发展历史和目前的版本情况，以及 Visual Basic 6.0 的特点。

1.2.1 Visual Basic 6.0 版本简介

Microsoft 公司于 1991 年推出 Visual Basic 1.0 版，接着于 1992 年推出 2.0 版本，1993 年推出 3.0 版本，1995 年推出 4.0 版本，1997 年推出 5.0 版本，1998 年推出 6.0 版本。随着版本的改进，Visual Basic 已逐渐成为简单易学、功能强大的编程工具。从 1.0 到 4.0 版本，Visual Basic 只有英文版，而 5.0 之后的 Visual Basic 在推出英文版的同时，又推出了中文版，这大大方便了中国用户。

Visual Basic 6.0 共有 3 种版本，各自满足不同的开发需要，分别如下。

- Visual Basic 6.0 学习版(Learning)：Visual Basic 的基本版本，是一个入门的版本，主要针对初学编程的人员。该版本包含所有的内部控件（标准控件）、网格(Grid)控件、Tab 对象以及数据绑定控件。
- Visual Basic 6.0 专业版(Professional)：该版本为专业的编程人员提供了一套用于软件开发、功能完备的工具。它包括学习版本的全部功能，同时包括 ActiveX 控件、Internet 控件、Crystal Report Writer 和报表控件。

- Visual Basic 6.0 企业版(Enterprise)：可供专业编程人员开发功能强大的组内分式应用程序。该版本包括专业版本的全部功能，同时具有自动化管理器、部件管理器、数据库管理工具、Microsoft Visual SourceSafe 面向工程版的控制系统等。

根据安装版本的不同，Visual Basic 6.0 的程序界面也有一些变化，本书以 Visual Basic 6.0 企业版为例，但其内容同时适用于专业版和学习版。

1.2.2 Visual Basic 6.0 的特点

Visual Basic 6.0 是 Microsoft 公司推出的可视化的、面向对象的、基于事件驱动的应用开发环境。由于它继承了 BASIC 语言简单、易学的优点，又增强了可视化、分布式数据库及 Internet 的编程功能，是一种易学、实用的 Windows 面向对象的应用开发工具。

何谓 Visual Basic？从字面上看，“Visual”指的是开发图形用户界面（GUI）的方法，它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拖放到屏幕上合适的位置即可。“Basic”是“Beginner's A11-Purpose Symbolic Instruction Code”的简称，是一种在计算技术发展历史上应用最为广泛的语言。

Visual Basic 6.0 在原有 BASIC 语言的基础上有了很大的发展，至今包含了数百条语句、函数及关键词。专业人员可以用 Visual Basic 6.0 实现任何其他 Windows 编程语言所能实现的功能，而初学者也可以很容易建立起自己的应用程序。

Visual Basic 6.0 是强大的 Windows 平台上的开发工具，从开发个人或小组使用的小工具，到大型企业应用系统，甚至通过 Internet 遍及全球的分布式应用程序，Visual Basic 6.0 都可以实现。Visual Basic 6.0 之所以有如此广泛的应用是因为它具有以下特点：

- 真正面向对象的编程。开发人员在维护系统运行时只需修改少量的代码，同时也加快了系统开发的速度。
- 可视化的编程方法以及向导的功能。开发人员几乎不用加入太多代码就可以开发标准的 Windows 程序。
- 数据访问特性。允许对基于包括 Microsoft SQL Server 在内的数据库进行数据库应用程序开发。
- 可通过 ActiveX 技术使用其他应用程序提供的功能。例如 Word、Excel 以及其他 Windows 应用程序，也可直接使用 Visual Basic 6.0 创建的应用程序和对象。
- 强大的 Internet 应用程序开发能力。使用 Visual Basic 6.0，在应用程序中能够方便地通过 Internet 访问其他计算机的文档和应用程序。
- 开发的应用程序是真正的“.exe”文件，运行时可自由发布动态链接库(DLL)。

1.3 Visual Basic 6.0 的安装和启动

要想感受 Visual Basic 6.0 开发环境，首先要学会如何将 Visual Basic 6.0 安装在我们的计算机上，并学会如何启动 Visual Basic 6.0，下面将学习这部分内容。

1.3.1 安装 Visual Basic 6.0 的环境要求

下面列出了安装 Visual Basic 6.0 所需的软硬件环境，以下条目都是最小配置要求。

- 客户机安装 Windows 95 或 Windows NT(R)3.51 以上的操作系统。
- 一台带有 486/50MHz 处理器以上的兼容机。

- 鼠标。
- CD-ROM 驱动器。
- 16MB 以上内存。
- 如果是完全安装，则至少需要 50MB 的硬盘空间。
- 推荐使用 VGA 或更高分辨率的监视器。
- 对于网络用户，需要与 Windows 兼容的网络和服务器。

1.3.2　Visual Basic 6.0 的安装

安装 Visual Basic 6.0 一般在 CD-ROM 上进行，其安装步骤如下。

1. 插入 Visual Basic 6.0 光盘时，系统自动启动安装程序并显示【安装向导】窗口，如图 1-1 所示。
2. 单击[下一步(N) >]按钮，显示【最终用户许可协议】窗口，如图 1-2 所示。

图1-1　【安装向导提示】窗口

图1-2　【最终用户许可协议】窗口

3. 选择“接受协议”后单击[下一步(N) >]按钮，将显示【产品号和用户 ID】窗口。输入正确的产品号 ID、用户姓名、公司名称等注册信息后，单击[下一步(N) >]按钮，显示【Visual Basic 6.0 中文企业版】安装向导，如图 1-3 所示。
4. 单击[下一步(N) >]会出现【选择公用安装文件夹】对话框，如图 1-4 所示。

图1-3　【Visual Basic 6.0 中文企业版】安装向导

图1-4　【选择公用安装文件夹】对话框

5. 选择合适的文件夹，单击[下一步(N) >]按钮，显示【Visual Basic 6.0 中文企业版】安装程序窗口，如图 1-5 所示。

6. 单击 继续 按钮，输入合法的产品序列号后单击 确定 按钮。如果输入正确，则显示【产品序列号确认】窗口。之后，再单击 确定 按钮，显示选择安装类型窗口，如图 1-6 所示。

图1-5 【Visual Basic 6.0 中文企业版】安装程序

图1-6 【选择安装类型】窗口

7. 其安装的缺省路径为“C:\Program Files\Microsoft Visual Studio\VB98”，如果要自定义文件夹，则单击 更改文件夹(F)... 按钮，显示【改变目录】窗口，如图 1-7 所示。

图1-7 【改变目录】窗口

8. 选定文件夹后，单击 确定 按钮，【改变目录】窗口自动关闭，则安装文件夹为自定义文件夹。如图 1-6 所示，其中的典型安装操作简单，适用于初学者；自定义安装允许选择安装组件，要求用户熟悉各个组件的功能和用途，适用于高级用户。
9. 单击 按钮，选择典型安装，显示【使用新的 Visual SourceSafe 数据库格式】窗口。根据提示选择所需格式，单击 是(Y) 或 否(N) 按钮，系统便开始安装 Visual Basic 6.0 应用组件，如图 1-8 所示。

图1-8 【安装过程】窗口

10. 安装程序在安装完毕时，显示如图 1-9 所示的【重新启动 Windows】窗口，要求重新启动计算机。单击[重新启动 Windows(R)]按钮，重新启动计算机，以更新系统的配置。

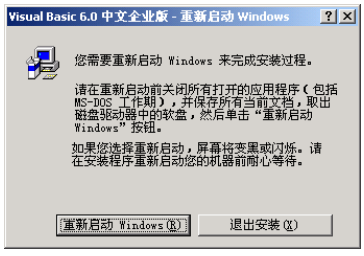


图1-9　【重新启动】窗口

如图 1-6 所示，在【选择安装类型】窗口中，单击按钮，选择自定义安装，显示【自定义安装】窗口，选择安装组件，如图 1-10 所示。

图1-10　【自定义安装】窗口

1.3.3　启动 Visual Basic 6.0

完成 Visual Basic 6.0 的安装过程后，就可启动 Visual Basic 6.0 了。

开机并进入 Windows 后，可用多种方法启动 Visual Basic 6.0。

- 第一种方法如图 1-11 所示，使用[开始]菜单中的【程序】命令。操作如下：

1. 单击 Windows 环境下的[开始]按钮，弹出一个菜单，把光标移到【程序】命令上，将弹出下一级联菜单。
2. 把光标移到【Microsoft Visual Basic 6.0 中文版】，弹出下一级联菜单，即进入 Visual Basic 6.0 程序组。
3. 单击【Microsoft Visual Basic 6.0 中文版】，即可进入 Visual Basic 6.0 编程环境。

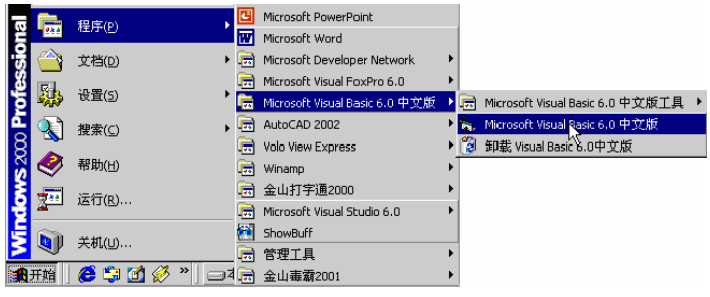


图1-11　启动 Visual Basic 6.0

- 第二种方法，通过使用“我的电脑”。操作如下：

1. 双击“我的电脑”，弹出一个窗口，然后单击 Visual Basic 6.0 所在的硬盘驱动器盘符，将打开相应的驱动器窗口。
2. 单击驱动器窗口中的“VB98”文件夹，打开【VB6】窗口。
3. 双击“VB6.exe”图标，即可进入 Visual Basic 6.0 编程窗口。

- 第三种方法，使用开始菜单中的【运行】命令。操作如下：

1. 单击开始按钮，弹出一个菜单，然后单击【运行】命令，将弹出一个对话框。如图 1-12 所示。

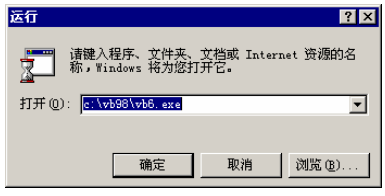


图1-12 【运行】对话框

2. 在【打开】栏内输入 Visual Basic 6.0 启动文件的名字（包括路径）。例如“c:\vb98\vb6.exe”。
3. 单击确定按钮就可启动 Visual Basic 6.0。

- 第四种方法，建立启动 Visual Basic 6.0 的快捷方式（具体操作查有关资料）。

1.3.4 添加、删除 Visual Basic 6.0 组件

在第一次安装 Visual Basic 6.0 时，由于经验不足，用户可能会多安装一些不必要的组件，或者少安装了需要的组件。这时，可以通过重新运行安装程序来定制需要的组件。步骤如下：

1. 在 CD-ROM 驱动器插入安装盘。
2. 安装程序将自动运行。
3. 选取“安装 Visual Basic 6.0”，安装程序会检测当前系统安装过的 Visual Basic 6.0 组件。检测完毕后，屏幕上会出现如图 1-13 所示的对话框。

图1-13 定制 Visual Basic 6.0 安装组件

4. 在【Visual Basic 6.0 中文专业版安装程序】对话框中选取添加/删除(A)...按钮，屏幕上会出现图 1-14 所示的对话框。

图1-14　安装维护

5. 在图 1-14 所示对话框的【选项】列表框中选定要安装的部件，也可以删除已安装的部件。
6. 按照屏幕上的安装指示进行下面的操作，即可完成添加或删除组件。

1.4 小结

通过本章的学习，我们熟悉了 Visual Basic 6.0 的基本特点，具体学习了以下内容：

- 可视化编程的概念和特点。
- Visual Basic 6.0 的版本及特点。
- Visual Basic 6.0 的安装和启动，以及如何添加、删除 Visual Basic 6.0 的组件。

1.5 习题

一、 选择题

1. Visual Basic 6.0 开发工具的特点是（　）。
   A. 面向对象　B. 可视化事件　C. 基于事件驱动　D. 全制动
2. 可视化开发的特点有（　）。
   A. 可利用图标创建对象　B. 在开发过程中就能见到开发的部分成果
   C. 开发工作对用户是透明的　D. 所见即所得
   E. 根据程序流程图开发
3. 为同一窗体内的某个对象设置属性，所用 Visual Basic 6.0 语句的一般格式是（　）。
   A. 属性名=属性值　B. 对象名.属性值=属性名
   C. Set 属性名=属性值　D. 对象名.属性名=属性值
4. Visual Basic 6.0 的启动有多种方法，下面不能启动 Visual Basic 6.0 的是（　）。
   A. 使用【开始】菜单中的【程序】命令
   B. 使用【开始】菜单中的【运行】命令，在弹出的对话框中输入 Visual Basic 6.0 启动文件的名字
   C. 使用“我的电脑”，在 Visual Basic 6.0 所在硬盘驱动器中找到相应的 Visual Basic 6.0 文件夹
   D. 先打开 Visual Basic 6.0 的【文件】菜单，再按 Zlt+Q 组合键

二、 填空题

1. VB 的全称是_____，其编程具有_____、_____、_____等特点。
2. Visual Basic 6.0 中的控件一般称为对象，对象具有_____、_____和事件，鼠标单击即是对象的_____。
3. Visual Basic 6.0 是一种面向_____可视化编程语句，采用了驱动的编程机制。
4. Visual Basic 6.0 分为 3 种版本，这 3 种版本是_____、_____和_____。
5. 事件的过程名由_____和事件名组成，其间用下划线连接。

三、 问答题

1. 安装 Visual Basic 6.0 时最基本的软硬件配置有哪些?
2. 如何利用命令行运行 Visual Basic 6.0?

第2章　Visual Basic 6.0 集成开发环境

本章主要介绍 Visual Basic 6.0 强大的集成开发环境，包括 Visual Basic 6.0 集成开发界面、菜单系统、工具栏以及各个工作窗口，通过本章的学习，将能够根据自己的需要设置开发环境。

本章学习目标

- Visual Basic 6.0 的菜单系统。
- Visual Basic 6.0 的工具栏。
- Visual Basic 6.0 的各种基本窗口。
- 集成开发环境的设置。

2.1 Visual Basic 6.0 集成开发环境

同大多数高级编程语言一样，Visual Basic 6.0 不仅仅是一种语言，而且是一个包括应用程序开发、测试、查错以及发布的综合集成开发环境。因此，要使用 Visual Basic 6.0 进行程序设计，必须首先熟悉 Visual Basic 6.0 的集成开发环境。

按照第 1 章介绍的方法启动 Visual Basic 6.0，可看到如图 2-1 所示的窗口。该窗口中列出了 Visual Basic 6.0 可建立的应用程序的类型，双击要建立的应用程序的图标，或者选择一个应用程序后单击 打开(O) 按钮，即可创建该类型的应用程序。

图2-1　Visual Basic 6.0 可创建的应用程序类型

如果需要调用原来已有的应用程序，单击【现存】选项卡，这时将出现一个【打开文件】的对话框，用户可以根据应用程序所在的目录找到该文件，然后打开它。

如果需要调用最近一段时间编制的应用程序，单击【最新】选项卡，这时该选项卡会列出用户最近编制的所有应用程序。

在图 2-1 中，可以看到下面 9 种类型的应用程序。

- 标准 EXE：标准 EXE 程序是 Visual Basic 6.0 典型的应用程序，一般用户要创建的应用程序都是这种类型的，它可以最终生成一个可执行的应用程序。
- Active EXE 和 ActiveX DLL：Active EXE 构件是支持 OLE 的自动化服务器程序，它可以嵌入或链接到用户的应用程序中进去。这两种类型的应用程序在编制程序时是一样的，只不过在编译时，Active EXE 编译为可执行文件，ActiveX

DLL 编译成动态连接库。

- ActiveX 控件：用于开发用户自己的 ActiveX 控件。
- ActiveX 文档 EXE 和 ActiveX 文档 DLL：ActiveX 文档实际上是能够在支持 Web 浏览器环境中运行的 Visual Basic 6.0 应用程序。同 Active EXE 和 ActiveX DLL 一样，这两种 ActiveX 文档在编译时，一个编译为可执行文件，一个编译成动态连接库。
- Visual Basic 6.0 应用程序向导：这个向导可以帮助用户建立应用程序的框架，通常是在需要开发项目时才会用到，它可以减轻用户在编程时的工作量。
- Visual Basic 6.0 企业版控件：这也是企业版中提供的类型，用于开发用户自己的 Visual Basic 6.0 控件。
- 外接程序：这一类型的应用程序可以扩展 Visual Basic 6.0 集成环境的功能。

在【新建工程】对话框中，选择【标准 EXE】后，单击 打开(O) 按钮，屏幕上会显示 Visual Basic 6.0 集成开发环境的主窗口，如图 2-2 所示。

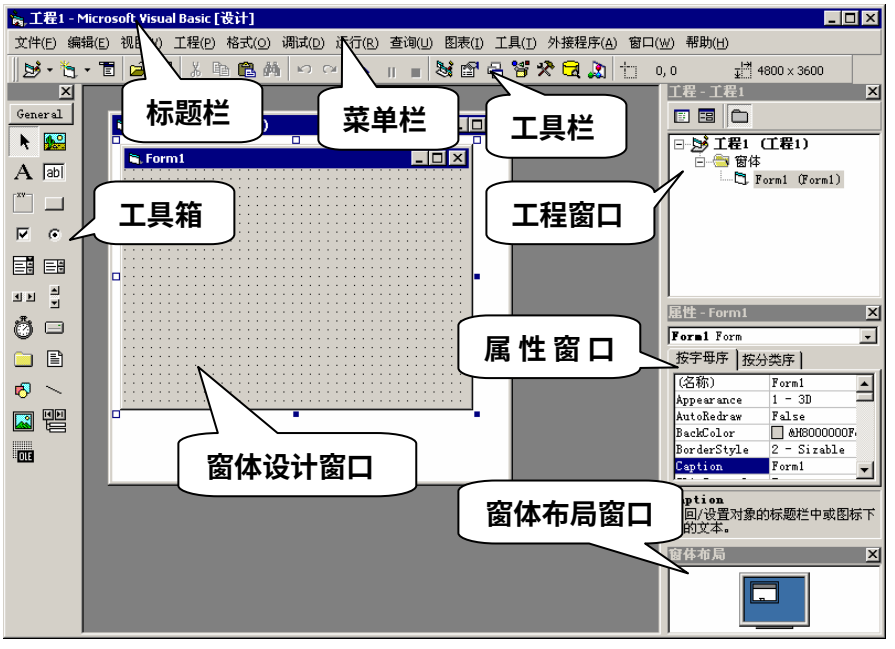


图2-2 Visual Basic 6.0 集成开发环境主窗口

从图 2-2 中可以看出 Visual Basic 6.0 的集成开发环境中包括标题栏、菜单栏、工具栏、工具箱、【窗体设计】窗口、【工程】窗口、【属性】窗口、【窗体布局】窗口等组件，其中，【工程】窗口也可称为【窗体资源管理器】窗口。下面将分别进行介绍。

2.2 Visual Basic 6.0 的菜单

在介绍 Visual Basic 6.0 的菜单之前，首先应当了解一些 Visual Basic 6.0 菜单系统的约定：

- 菜单项后面有组合键，例如【文件】菜单中的【新建文件】的选项后面有 Ctrl+N。这就说明该功能项有快捷键。使用方法是按住 Ctrl 键的同时，再按下 N 键，便可激活该选项。

- 菜单项的右边有一个小黑箭头，表示该菜单项有子菜单。
- 菜单项的右边是省略号(…)，表示单击该菜单项后，会弹出一个对话框。
- 菜单项的颜色为灰，表示该菜单项现在不可用。
- 菜单项的左边有√，表示开关的作用。出现√表示该功能项在使用中。例如显示菜单中属性选项，当选中后就会弹出属性窗口。

Visual Basic 6.0 的菜单栏如图 2-3 所示。上面显示了所有 Visual Basic 6.0 的命令。除了提供标准的【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】、【运行】和【调试】等。

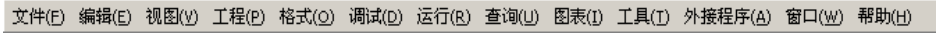


图2-3　菜单栏

2.2.1　【文件】菜单

【文件】菜单包含了与用户访问文件操作有关的选项，包括建立新文件、打开已有的工程、关闭工程、保存工程、打印等选项。表 2-1 列出了文件菜单中所有选项的说明。

表 2-1　　【文件】菜单功能简介

菜单项	功能简介
新建工程	显示【新建工程】对话框，选择要创建的工程类型，当新建工程时，如果存在另一个已打开的工程，那么将提示保存现有的工作
打开工程	关闭当前加载的工程或工程组，然后打开存在的工程或工程组 只要系统资源允许，可以打开尽可能多的工程
添加工程	显示【添加工程】对话框，以便能把新的或已有的工程添加到当前打开的工程组中。如果仅存在一个已打开的工程，则 Visual Basic 6.0 添加该工程并能创建工程组。仅在有多个工程时，工程组才能存在
移除工程	从当前打开的工程组中删除选定的工程。如果工程还须改变，则将提示保存它们，然后工程被关闭并从工程组中删除
保存工程	保存当前的工程及其所有部件，如果是第一次保存需要输入项目名和窗体名称
工程另存为	以其他名称保存当前的工程及其所有部件
打印	将窗体和代码的内容打印出来
打印设置	显示标准的【打印设置】对话框，指定打印机、页面方向、页面大小、纸张来源以及其他的打印选项
退出	退出 Visual Basic 6.0

2.2.2　【编辑】菜单

【编辑】菜单包括了与用户编辑程序、文本、对象等有关内容的菜单选项。表 2-2 列出了【编辑】菜单中所有选项的说明。

表 2-2　　【编辑】菜单的功能简介

菜单项	功能简介
撤销	撤销上一次操作
恢复	恢复上一次撤销的操作
剪切	将选中的内容删除掉，同时送到剪切板上

续表

菜单项	功能简介
复制	复制选中的内容到剪切板中
粘贴	将剪切板中的内容复制到光标处
粘贴链接	将 OLE 对象插入一般字段中
删除	清除所选的内容
全选	选择活动窗口中的所有内容
查找	打开查找对话框， 将光标定位在查找内容处
查找下一个	从插入点开始查找且选中下一个指定的字符串
替换	打开替换对话框，替换文本内容
缩进	将所有选定行移到下一个定位点。选定的所有行，会移动相同空格数到被选区间内，并保留同样的相对缩进位置
凸出	将所有选定行移到前一个定位点。选中的所有行，会移动相同空格数到被选区间里，并保留同样的相对缩进位置
插入文件	打开【Insert File】对话框，以便能在代码窗口中当前指针位置的现存文件插入文本
属性/方法列表	在添加对象之前，会在【代码】窗口打开一个列表框，其中包含对象可使用的属性及方法
常数列表	当设置一个属性，要定义其常数时，在键入“=”号之前，会在【代码】窗口中打开一个列表框，框内有正确的常数可供引用
快速信息	在【代码】窗口中提供选定变量、函数、语句、方法或过程的语法
参数信息	在【代码】窗口中显示一个弹出式窗口，其中包括了函数或语句参数的信息
自动完成关键字	自动补足在代码窗口中输入的指令
到行	在代码窗口中跳转到指定的行
书签	显示一个菜单，可让用户在代码窗口中创建或删除一个书签标记

2.2.3 【视图】菜单

【视图】菜单用于在【对象】、【代码】窗口之间的切换，显示与隐藏 IDE 构件等命令。表 2-3 列出了【视图】菜单中所有选项的说明。

表 2-3 【视图】菜单的功能简介

菜单项	功能简介
代码窗口	按当前所选择的对象，显示或启动过程代码窗口
对象窗口	显示活动的项目
定义	显示光标处变量或过程被定义在程序代码窗口中的位置。如果此定义是一个引用程序库，它会被显示在对象浏览器中
最后位置	让用户快速转到上次在过程代码中的位置
对象浏览器	显示对象浏览器，此对象浏览器会列出对象库、类型库、类、方法、属性、事件及可在过程中使用的常数，和在工程中定义的模块或过程
立即窗口	显示立即窗口及其中的过程代码中的调试语句生成或由直接键入窗口中的命令

续表

菜单项	功能简介
本地窗口	显示本地窗口且自动地显示在当前堆栈中所有的变量及其值
监视窗口	显示监视窗口并显示当前的监视表达式。如果该工程有定义监视表达式的话，监视窗口会自动出现
调用堆栈	显示一个对话框，其中会列出所有调用的过程
工程管理器	显示工程管理，其中可显示当前打开工程的层次列表及其内容
属性窗口	显示属性窗口，此属性窗口会依所选择的窗体、控件、类、工程或模块来列出设计时属性
窗体布局窗口	显示窗体布局窗口，这里可以预览及定位窗体
属性页	为用户控件显示属性页，在设计时改变该控件的属性
工具箱	显示标准的 Visual Basic6.0 控件连同已添加到工程中的 ActiveX 控件和可插入对象
调色板	显示或激活调色板，就能改变窗体或控件的颜色
工具栏	列出所有 Visual Basic 6.0 固有的工具栏及自定义命令

2.2.4　【工程】菜单

【工程】菜单中主要包括与 Visual Basic 6.0 工程管理相关的菜单项，例如向工程中添加窗体、模块、属性、过程等，它还提供了定制工具箱的功能。表 2-4 列出了【工程】菜单中各菜单项的功能。

表 2-4　　【工程】菜单的功能简介

菜单项	功能简介
添加窗体	显示【添加窗体】对话框，向当前工程中添加新的或现存的窗体
添加 MDI 窗体	显示【添加 MDI 窗体】对话框向当前工程中添加新的或现存的 MDI 窗体
添加模块	显示【添加模块】对话框，向当前工程中添加新的或现存的模块
添加类模块	显示【添加类模块】对话框，向当前工程中添加新的或现存的类模块
添加用户控件	显示【添加用户控件】对话框，向当前工程中添加用户控件
添加属性页	显示【添加属性页】对话框，向当前工程中添加新的或现存的属性页
添加用户文档	显示【添加用户文档】对话框，向当前工程中添加新的或现存的文档
添加 Data Report	向当前工程中添加一个新建窗体，该窗体包含有一个报表设计器
添加 Data Enviroment	向当前工程中添加一个新建窗体，该窗体包含有一个数据环境管理器
添加文件	向当前工程中添加已存在文件，它包含了前面所有添加菜单项
移除	从工程中删除当前选项
引用	显示【引用】对话框，添加对象库或类型库，即对工程的引用
部件	显示【部件】对话框，可向工具箱添加控件、设计器或可插入对象
工程 1 属性	显示【工程属性】对话框，能看到所选工程的工程属性

2.2.5　【格式】菜单

【格式】菜单提供了在创建窗体的过程中需要用到的调整控件布局的功能，主要包括控件

的对齐、控件之间的间距等，还提供了设置控件的 Tab 次序的功能。表 2-5 列出了【格式】菜单中菜单项的主要功能。

表 2-5　　【格式】菜单的功能简介

菜单项	功能简介
对齐	对齐所有选取的对象，参考对象为周围有白色控点的对象
统一尺寸	以周围有白色控点的对象为参考对象，将所有选取到的对象大小设成一样
按网格调整大小	调整选取到对象的长度与宽度，使每一个对象的四个角都对齐到最近的网格上
水平间隔	改变选取到的对象之间的水平间隔
垂直间隔	改变选取到的对象之间的垂直间隔
在窗体中居中对齐	将所有选取到的对象的中心点排在窗体中心线
顺序	改变选取到的对象在窗体上的显示层
锁定控件	在窗体上锁定当前位置的所有控件，这样它们就不再移动

2.2.6　【调试】菜单

【调试】菜单主要的功能是帮助用户对编制的应用程序进行调试。在该菜单中有 4 项控制应用程序的运行方式。其中，【逐语句】表示一次执行一个语句；【逐过程】与【逐语句】相似，只有在当前的语句含有一个对过程的调用时，两者才会有差异；【逐过程】的含义是执行当前执行点所在函数中剩余未执行的行，下一个被显示的语句是紧随在该过程调用后的语句。这 3 个命令都是在中断模式下执行的，如果用户的应用程序处于设计方式，可以使用【运行到光标处】来选定希望执行到哪一行语句停止。

单击【添加监视】命令，屏幕上会显示【添加监视】对话框，用户可以输入一个监视表达式，在中断模式下运行时，用户可以观察监视表达式的值的变化过程。选择【编辑监视】命令，屏幕上会显示【编辑监视】对话框，用户可以编辑或删除所有的监视表达式。选择【快速监视】命令，可以在【快速监视】对话框中直接监视选择表达式的当前值。

利用【切换断点】命令，可以设置或删除当前行上的一个断点。利用【清除所有断点】命令，可以清除所有工程中设置过的断点。

使用【设置下一条语句】命令，将光标定位到希望在执行点开始运行的那行代码上面，就可以设置程序的执行入口。利用【显示下一条语句】命令可以将光标移到下一行将被执行的程序，并且突出下一个将被执行的语句。

2.2.7　【运行】菜单

【运行】菜单的主要功能是运行用户应用程序，它包括 5 个菜单项。表 2-6 列出了它们的功能简介。

表 2-6　　【运行】菜单的功能简介

菜单项	功能简介
启动	关闭所有正在设计的窗体、从启动窗体运行应用程序
全编译执行	对工程执行全编译后运用应用程序

续表

菜单项	功能简介
中断	停止一个正在运行的程序，并切换到中断模式
结束	停止程序的运行，并返回设计状态
重新启动	在任何一种中断后重新启动应用程序

2.2.8　【工具】菜单

【工具】菜单的主要功能是向模块和窗体中加入过程和过程属性，还包括向窗体中添加菜单等功能。

通过单击【添加过程】按钮，可以向当前窗体或模块中添加过程，选择【过程】属性可以在当前窗体或模块中添加【过程】属性。

单击【菜单编辑器】按钮，屏幕上会出现【菜单编辑器】对话框，可以向当前窗体中添加菜单，这些内容将在后面章节详细介绍。

利用【选项】命令，可以设置 Visual Basic 6.0 编程环境的属性。

其他几个菜单或者使用比较少，或者在其他应用程序中经常出现，这里就不再赘述了。

2.3　Visual Basic 6.0 的工具栏

虽然仅仅通过菜单命令已经可以完成 Visual Basic 6.0 集成开发环境中的所有功能，但是利用菜单操作比较麻烦，尤其对于常用的命令。在 Visual Basic 6.0 中提供了各种工具栏，可供用户直接执行各种命令。用户还可以根据需要在屏幕上放置多个工具栏，并且可以按照自己喜欢的样式定制工具栏的样式，把工具栏放在屏幕的上部、底部或两边。Visual Basic 6.0 中提供了 4 种工具栏，分别是【标准】、【编辑】、【窗体编辑器】和【调试】工具栏。用户还可以根据自己的习惯，将常用的命令组合成新的工具栏。

2.3.1　【标准】工具栏

【标准】工具栏中包括了在集成环境中的 4 种最常用的命令。常用的编辑命令，如添加新的窗体、模块，以及常用的各种窗口的显示控件等。图 2-4 显示了【标准】工具栏的组成。

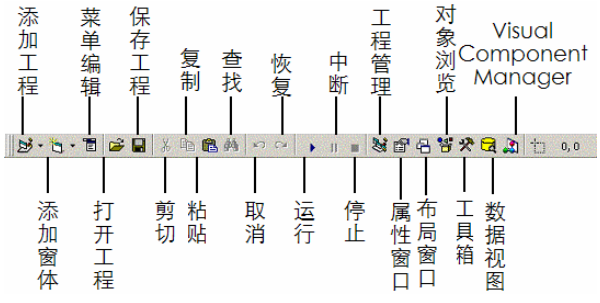


图2-4　【标准】工具栏

在【标准】工具栏中，【添加工程】按钮对应着【文件】菜单的【添加菜单】命令，前面已经对其功能作了简单的描述。在该按钮的右边还有一个小的黑箭头，用鼠标单击之后会出现一个下拉菜单，用户可以选择需要添加工程的类型。

【菜单编辑】按钮对应着【Tools】菜单中的【菜单编辑器】命令，用户可以利用它来创建菜单。【打开工程】和【保存工程】按钮分别对应着【文件】菜单中相应的命令。

【剪切】、【复制】、【粘贴】、【查找】、【取消】、【恢复】分别对应【编辑】菜单中的相应命令；【运行】、【中断】、【停止】分别对应【运行】菜单中的各项命令。【工程管理】、【属性】窗口、【布局】窗口、【对象浏览】、【工具箱】分别对应【视图】菜单中的相应命令。

由此可见，工具栏中的按钮实际上就是菜单使用的一种快捷方式，是否使用工具栏完全在于个人习惯。

2.3.2 【编辑】工具栏

在【编辑】工具栏中主要包括在代码编辑过程中需要用到的一些编辑命令和帮助命令。图 2-5 显示了【编辑】工具栏的组成。

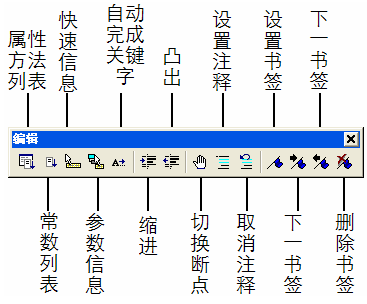


图2-5 【编辑】工具栏

在【编辑】工具栏中，几乎所有的按钮都对应【编辑】菜单中相应命令，其中前 5 个按钮提供了一些帮助信息，可以帮助用户在编写代码过程中更加方便准确地使用 Visual Basic 6.0 中的函数和命令。

【缩进】、【凸出】命令用来控制文本的对齐位置；【设置断点】按钮包含在在【调试】菜单中；【设置注释】和【取消注释】按钮可以非常方便地将选中的文字设置为注释文字，或者恢复为以前的样式。

【编辑】工具栏中最后 4 个按钮是用来设置和使用书签的。其中【设置书签】按钮用来在指定的位置设置书签标记，【上一书签】和【下一书签】用来切换不同书签的位置，【删除书签】用来在指定位置删除书签。

2.3.3 【窗体布局】工具栏

【窗体布局】工具栏主要对应着【格式】菜单中的命令，主要用来控制窗体中控件的相对位置和对齐方式。图 2-6 显示了【窗体布局】工具栏的组成。

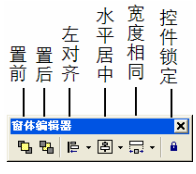


图2-6 【窗体布局】工具栏

其中，【置前】和【置后】按钮用来控制控件的叠放次序和显示方式。【左对齐】按钮用来控制一组按钮之间的左对齐，通过其右边的黑箭头按钮，用户还可以设置其他的对齐方式。

【水平居中】按钮用来控制一组控件之间居中对齐的方式，通过其右边的黑箭头按钮用户还可以设置其他居中对齐的方式。利用【宽度相同】按钮和其旁边的黑箭头按钮，用户可以将一组控件设置成尺寸相同的控件。利用【锁定控件】按钮，用户可以将当前窗体中所含的控件锁定，不能再移动。

2.3.4　【调试】工具栏

【调试】工具栏主要在调试应用程序时使用，它对应着【调试】菜单中的所有命令。图 2-7 显示了【调试】工具栏的组成，可以对照前面的对【调试】菜单的介绍了解各按钮的功能，这里就不再赘述了。

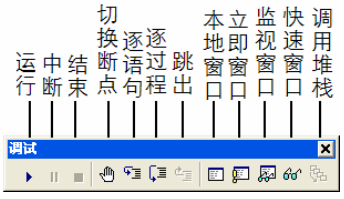


图2-7　【调试】工具栏

2.3.5　工具栏的设置

Visual Basic 6.0 的集成开发环境不仅提供了丰富的工具栏，使用户的操作更加方便快捷，而且还提供了定制工具栏的功能。通过定制工具栏，用户可以设置工具栏的位置、工具栏中的按钮、工具栏中按钮的图标等。本节将详细介绍定制工具栏的方法。

一、 移动工具栏

在 Visual Basic 6.0 中所有的工具栏都是可以移动的，它们可以移动到屏幕上的任何位置。如果移动到屏幕的顶部，工具栏将显示为条状，紧贴在菜单下面；如果是在屏幕的其他位置，工具栏将显示为一个窗口。

移动工具栏的方法很简单，将鼠标光标移动到工具栏左边双线样式的地方，按住鼠标左键不放，拖动鼠标。如果向下移动鼠标，用户可以看到工具栏显示为一个窗口，同时工具箱和工程管理器窗口也变高，如图 2-8 所示，拖动到合适的位置后松开鼠标左键，原来的工具栏就移动到了新定义的位置。

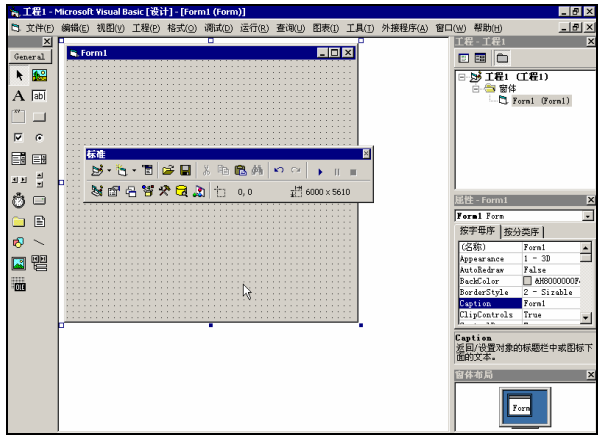


图2-8　移动工具栏

二、 控制工具栏的显示

Visual Basic 6.0 的集成开发环境中提供了许多工具栏，我们可以根据不同的要求将需要的工具栏打开，不需要的工具栏关闭。

控制工具栏显示的方法有两种。一种方法是利用菜单，用鼠标单击【视图】菜单，将鼠标移动到【工具栏】菜单上方，这时屏幕上会出现一个子菜单，如图 2-9 所示。单击要打开的工具栏，这时该工具栏前面会出现一个√，同时屏幕上也会出现该工具栏；如果要关闭该工具栏，可用鼠标再次单击菜单中相应的工具栏命令。

控制工具栏显示的另一种方法是利用工具栏设置对话框，用鼠标单击【视图】菜单，将鼠标移动到【工具栏】菜单上方，然后单击【自定义】命令，屏幕上会出现如图 2-10 所示的【自定义】对话框。

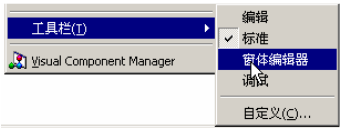


图2-9 【工具栏】菜单

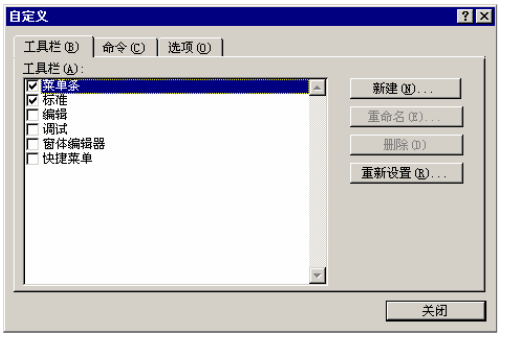


图2-10 【自定义】对话框

在该对话框的第一个选项卡中，显示了当前显示的工具栏和菜单条的情况。和前面用菜单操作的方法类似，用鼠标单击对应工具栏前面的复选框即可控制该工具栏的显示。

对话框中的快捷菜单是把 Visual Basic 6.0 在编程中常用的一些命令组合起来所形成的一个新的工具条，如果喜欢这种方式也可以将它打开，显示在主界面中。

三、 添加或删除工具栏中的按钮

如果在已有的工具栏按钮中没有所需要的功能，用户可以通过【自定义】对话框向工具栏中添加需要的功能按钮。同样，如果有一些功能按钮不需要的话，也可以很方便地删除掉。

向工具栏中添加或删除工具按钮的操作如下。

1. 打开需要调整的工具栏。
2. 单击【自定义】对话框中的【命令】选项卡，将当前对话框设置为【命令】选项卡，在该选项卡中列出所有该选项卡中提供的命令，如图 2-11 所示。

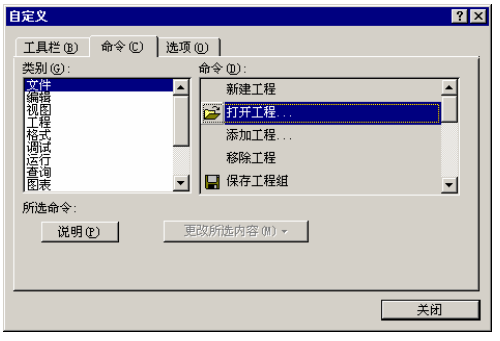


图2-11 【命令】选项卡

3. 在【类别】列表框中打开相应的菜单，这时在【命令】列表框中将列出该菜单下的所有命令。
4. 单击需要添加的命令按钮，该命令会变成图中选中的样式。按住鼠标左键不放，然后拖动到要添加工具按钮的工具栏，如图 2-12 所示。

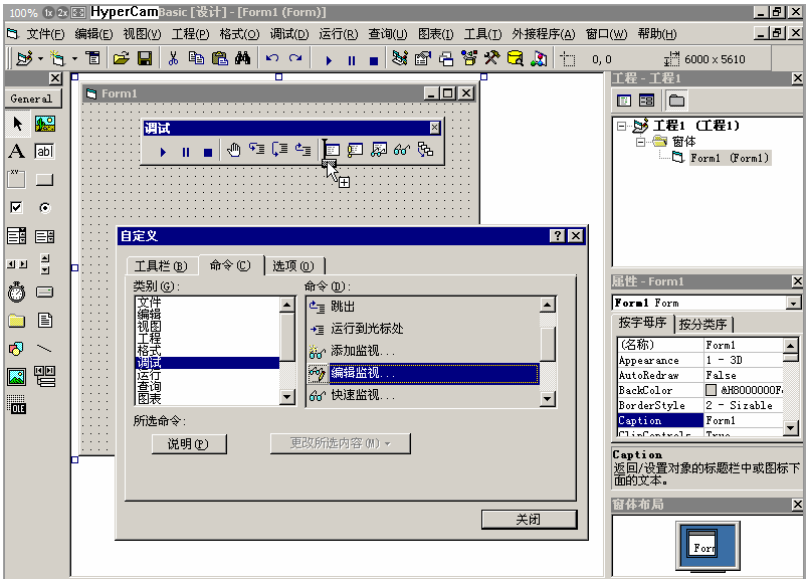


图2-12　向工具栏中添加工具按钮

5. 如果要向工具栏中添加【下拉式工具】按钮，在【类别】列表柜中选择【Built-in Menus】，这时在【命令】列表框中会列出所有的【下拉菜单】按钮，然后按照上一步的方法，向工具栏中添加【下拉式工具】按钮，用户会发现该工具按钮实际上是一个文字按钮，在后面会介绍如何将文字按钮改变成图形按钮。
6. 单击工具栏中新添加的【下拉式工具】按钮左边的小黑箭头，这时会出现一个下拉菜单，然后按照步骤 4 的方法，可以向下拉菜单中添加【命令】按钮。
7. 如果要删除工具栏中的【工具】按钮，单击工具栏中的相应【工具】按钮，然后与步骤 4 相反，按住鼠标左键不放，拖动到工具栏外的任何位置，该【工具】按钮即可被删除。

在添加和删除工具栏中的【工具】按钮时应当注意鼠标的形状，如果鼠标中有一个加号，表示当前可以将该按钮添加到相应的工具栏中；如果鼠标中有一个叉号，表示剔除该按钮或者该按钮不能添加到指定的位置。

四、 工具栏和菜单的显示样式

在使用工具栏的过程中，如果觉得系统设置的【工具】按钮的图标或说明不能准确反映该按钮的功能，可以按照自己习惯的方式重新定义【工具】按钮的显示样式。

要改变【工具】按钮的显示样式，可以按照下面的步骤进行：

1. 打开【自定义】对话框，并切换到【命令】选项卡中。
2. 用鼠标单击需要设置的工具栏中的【工具】按钮，该按钮周围将会出现一个方框，这时【命令】选项卡中的【更改所选内容】按钮中的文字将会变黑，表示可以使用。
3. 单击【命令】选项卡中的【更改所选内容】按钮，会出现一个下拉式菜单，如

图 2-13 所示。

4. 用鼠标单击【更改按钮图标】命令，打开一个【图标】菜单，选择一个图标即可改变原工具按钮图标。
5. 在命名后面的文本框中，用户可以输入对应于该命令的菜单文字。

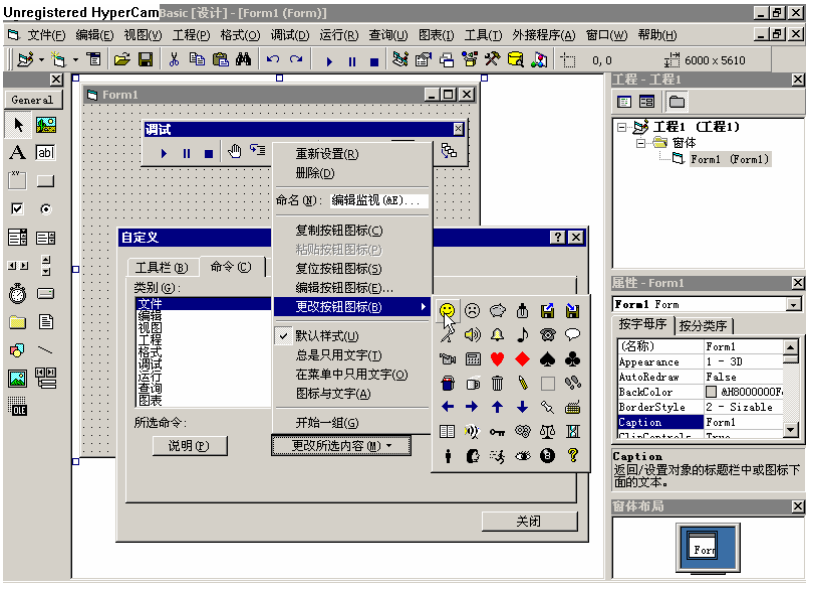


图2-13 【更改所选内容】下拉菜单

6. 设置【工具】按钮样式。若选择【默认样式】，则在菜单和工具栏中，选中的【工具】按钮将显示默认按钮样式，也即在工具栏中显示为【图标】按钮，在菜单中显示为图标与文字或者只显示文字；若选择【总是使用文字】，则在菜单和工具栏中，选中的【工具】按钮将显示为一个按钮，也即只显示文字；若选择【在菜单中只使用文字】，则在菜单中，选中的【工具】按钮将显示为一个【文字】按钮；若选择【图标与文字】，则在菜单和工具栏中，选中的【工具】按钮的图标与说明文字都会显示出来。
7. 选择【开始一组】，将会在该按钮前面加上一个竖线表示与其他按钮分组。

五、 创建【自定义】工具栏

在 Visual Basic 6.0 集成环境中，不仅允许改变工具栏的设置，还提供了创建【自定义】工具栏的功能，这样就可以把编程中经常使用的命令组合在一起形成新的工具栏，以方便使用。

创建【自定义】工具栏可以按照下面的步骤进行：

1. 打开【视图】菜单，选中【工具栏】菜单项，单击【自定义】命令，屏幕上出现【自定义】对话框。
2. 在对话框的第一个选项卡中，单击 新建(N)... 按钮，屏幕上会提示用户输入【自定义】工具栏的名称。如图 2-14 所示，默认状态下工具栏的名称为【自定义 1】。
3. 单击 确定 按钮，屏幕上会出现一个新建的【工具栏】窗口，如图 2-15 所示，但是该窗口中没有任何【工具】按钮，同时在【自定义】对话框中会出现该工具栏的名称。

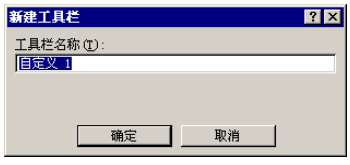


图2-14 新建工具栏

图2-15 【自定义 1】工具栏

4. 按照前面介绍的方法，向新建的工具栏中添加【工具】按钮，修改工具栏中【工具】按钮的显示方式。
5. 如果需要重新命名新建工具栏，在【自定义】对话框中选中该工具栏，然后单击 重命名(R)... 按钮，屏幕上会出现与图 2-14 所示类似的对话框，要求用户输入新的工具栏名称，输入修改的名称后，单击 确定 按钮即可修改该工具栏的名称。
6. 如果需要删除新建工具栏，在【自定义】对话框中选中该工具栏，然后单击 删除(D) 按钮，即可从系统中删除新建工具栏的设置。
7. 【自定义】工具栏创建完成后，用户可以按照前面介绍过的方法，通过菜单或者【自定义】对话框将自定义对话框显示出来。

六、 其他杂项的设置

前面介绍了工具栏设置的主要内容，Visual Basic 6.0 还提供了一些附加的功能用来设置工具栏的显示特性。

打开【自定义】对话框，单击【选项】选项卡，屏幕上的【自定义】对话框会变成如图 2-16 所示的界面。在该选项卡中总共有 4 个选项，下面列出了各个选项的含义和使用方法。

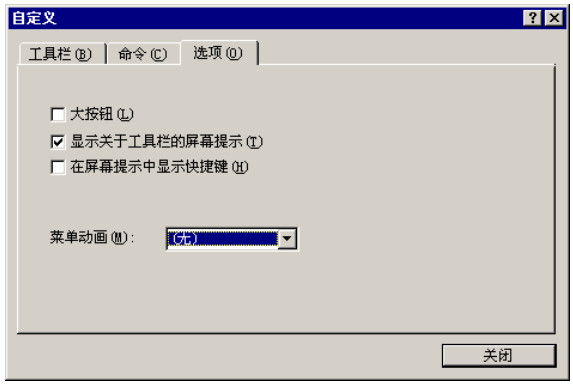


图2-16　设置工具栏的特殊显示

- 【大按钮】：可以使工具栏中的各命令按钮放大显示。
- 【显示关于工具栏的屏幕提示】：选中这个选项后，当鼠标停留在工具栏按钮上时，屏幕上会出现提示条，显示出该按钮的主要功能。
- 【在屏幕提示中显示快捷键】：选中这个选项后，屏幕的提示条除了显示该按钮的主要功能，还会显示该【工具】按钮的快捷键。
- 【菜单动画】：在这个下拉列表一共有 4 个选项，分别代表在打开菜单时不同的显示效果．用户可以自己体会一下每种显示效果的特点和不同。

2.4 Visual Basic 6.0 的基本窗口

Visual Basic 6.0 集成环境中有许多不同用途的窗口。包括【工程管理器】窗口、【代码】窗口、【属性】窗口等。在本节中将简要介绍这些窗口的组成、用途和主要使用方法。

2.4.1 【工程管理器】窗口

工程管理器，又称工程资源管理器，是在 Visual Basic 6.0 集成开发环境中用来管理工程的窗体。在这个窗体中显示工程的层次列表，以及所有的工程。同时还提供了一定的管理功能。

在 Visual Basic 6.0 集成开发环境中，默认状态下【工程管理器】的窗体是打开的，如果当前没有打开，可以通过【视图】菜单中的【工程管理器】命令打开；或者单击【标准】工具栏中的按钮激活该窗口。

打开后的【工程管理器】一般包括两个部分，分别是【工具】按钮和【浏览】窗口，如图 2-17 所示是一个典型的【工程管理器】。

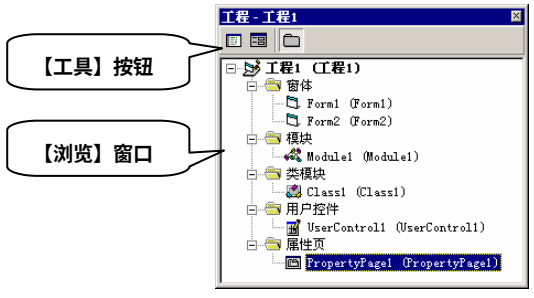


图2-17 工程管理器

在【工程管理器】中，一共有 3 个工具按钮，其功能分别如下。

- 【查看代码】按钮：显示代码窗口，以编写或编辑所选项目的目标代码。
- 【查看对象】按钮：显示选取的工程，可以是 ActiveX 对象或用户控件、模块的对象窗口。
- 【切换文件夹】按钮：单击该按钮可以将当前所有的文件夹都去掉，只显示包含在对象文件夹中的所有文件，再单击一次后会恢复为原来的样子。

在【工程管理器】的列表窗口中，列出了所有已装入的工程以及工程中的所有文件。在图 2-17 中已经可以看到一些项目文件类型。下面列出了在 Visual Basic 6.0 中可能出现的所有项目文件的类型和对应的解释。

- 窗体(Form)：所有与此工程有关的“.frm”文件。
- 模块(Module)：工程中所有的“.bas”模块。
- 类模块(Class Module)：工程中所有的“.cls”文件。
- 用户控件(User Control)：工程中所有的用户控件。
- 用户文档(User Document)：工程中所有的文档，即“.dob”文件。
- 属性页(Property Page)：工程中所有的属性页，即“.pag”文件。
- 相关文档(Related Document)：列出所有需要的文档，在此存放文档的路径而不是文档本身。当单击【查看对象】按钮时，Visual Basic 6.0 就会搜寻有关此文档类型的注册表，然后执行适当的过程来打开，可以将任何适当的文档类型放到工程中。
- 资源(Resource)：列出工程中所具有的资源。

2.4.2 【属性】窗口

在 Visual Basic 6.0 编程中，每一个控件和窗体都有许多属性，这些属性直接控制所对应的窗体和控件的外观和特性。在【属性】窗口中列出了所选取对象的属性以及属性值，用户可

以在设计时改变这些属性。当选取了多个控件时，【属性】窗口会列出所有控件都具有的公共属性。

若要打开【属性】窗口，应当在【窗体设计器】中选中一个窗体或窗体控件，然后在【工程设计器】中单击【查看属性】按钮，或者在【视图】菜单中选择【属性窗体】命令。【属性】窗口的界面如图 2-18 所示。

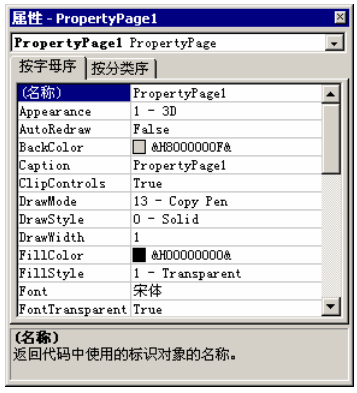


图2-18　【属性】窗口

【属性】窗口主要由下面几部分组成。

【对象】框：列出当前所选的对象，但只能列出现用窗体中的对象。如果选取了好几个对象，则会以第一个对象为准，列出各对象具有的公共属性。

【属性】列表：列出所选对象的属性及属性值。其中【按字母序】选项卡将按字母顺序列出所选的对象的所有属性，【按分类序】选项卡将根据性质列出所选对象的所有属性。同类型的属性被归为一类，用类似于文件管理的方式管理。如图 2-17 所示，当扩充或折叠列表时，可在分类名称的左边看到一个加号（+）或减号（−）图标，若要改变属性的设定，可以选择属性名然后输入，或直接选取新的设定。

显示属性类型和对属性的简短帮助。利用菜单中的快捷方式，可以打开或关闭属性的简短帮助，可用箭头键来移动属性的帮助列表。

2.4.3　【代码】窗口

在 Visual Basic 6.0 的编程中，所有的代码都是在【代码】窗口中编写的。【代码】窗口实际上就是个文本编辑器，它可以用来编写、显示和编辑表单、事件和方法程序的代码，可以打开任意多个【代码】窗口，从而方便地查看、复制和粘贴来自不同表单的代码。

(1)　启动【代码】窗口

若要打开【代码】窗口，在【窗体设计器】窗口中双击一个窗体或窗体控件，或者在【工程管理器】中单击按钮。从【视图】菜单中选择【代码窗口】命令，也可以打开【代码】窗口。【代码】编辑窗口的界面如图 2-19 所示。

(2)　【代码】窗口的组成

在【代码】窗口中主要由下面几个部分组成：

- 【对象】框：显示所选对象的名称。可以按下拉列表框中的右边箭头来显示此窗体中的对象。

- 【过程/事件】框：列出所有对应于【对象】框中对象的事件，这些事件是按名称的字母来排列的。当选择了一个事件时，与事件名称相关的事件过程就会显示在代码窗口中。如果在对象框中显示的是“通用”，则【过程】框会列出所有声明，以及为此窗体所创建的常规过程。如果正在编辑模块中的代码，则【过程】框会列出所有模块中的常规过程。在上述两实例中，在【过程】框中所选的过程都会显示在【代码】窗体中。

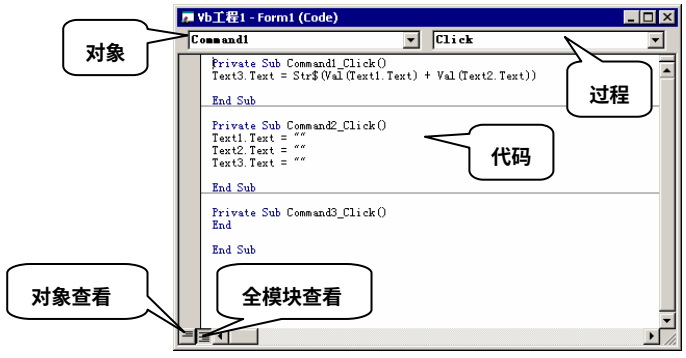


图2-19 【代码】窗口

- 拆分栏：将拆分栏向下拖放，可以将【代码】窗口分隔成两个水平窗格，两者都具有滚动条，可以在同一时间查看代码中的不同部分。显示在【对象】框以及【过程/事件】框中的信息是以当前拥有焦点的窗格之内的代码为准的。将拆分栏拖放到窗口的顶部或下端，或者双击折分栏，可以关闭一个窗格。
- 【过程查看】按钮：显示所选的过程，同一时间只能在【代码】窗口中显示一个过程。
- 【全模块查看】按钮：显示模块中全部的代码。

(3) 使用快捷方式编程

【代码】窗体是用来编辑代码的，在 Visual Basic 6.0 集成环境中提供了非常方便的编程信息帮助完成编码工作。下面通过一个例子来说明如何利用这些帮助信息。

假设要编写一条函数，例如编写下面的语句：

```
a=InputBox("请输入一个数","输入",1)
```

这是一个用来提示输入的函数，即使用户对它的使用方法还不熟悉也没有关系，Visual Basic 6.0 的集成环境会帮助完成函数的编写。在【代码】窗口中首先写“a= InputBox(”这时屏幕上就会出现该函数的提示信息，如图 2-20 所示。

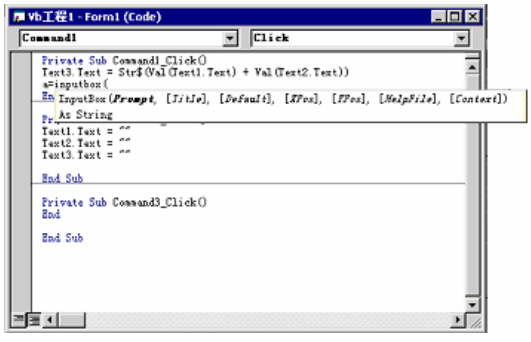


图2-20 快速信息提示

在该提示信息中，显示出了用户所写函数的结构、参数的类型等，可以根据这些提示信息进行代码的编制。当前参数是用黑体显示的，写完一个参数后，下一个参数会自动变成黑体。

在 Visual Basic 6.0 编程中，每一个控件和窗体都有一系列的属性、方法和事件，这些内容会在后面详细介绍。利用 Visual Basic 6.0 提供的【属性/方法列表】可以清楚地列出这些属性、方法和事件。

在编写程序的时候，如果要用到某个控件的属性或方法，只需要先写出该控件的名称和点操作符，这时屏幕上就会出现一个【属性/方法列表】，如图 2-21 所示。用户可以在该列表中选择一个属性或方法，然后双击它或按空格键即可。

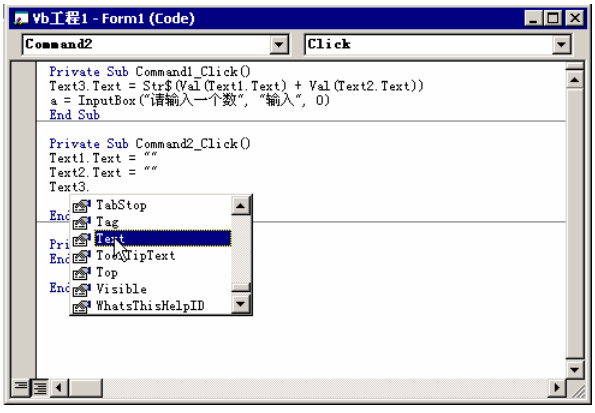


图2-21　利用【属性/方法列表】

2.5 集成开发环境的设置

熟悉了 Visual Basic 6.0 集成环境，用户还可以根据自己的习惯定义集成环境的样式。单击【工具】菜单中的【选项】命令，可以打开【选项】对话框，在这个对话框中用户可以设置 Visual Basic 6.0 集成环境的样式，如图 2-22 所示。

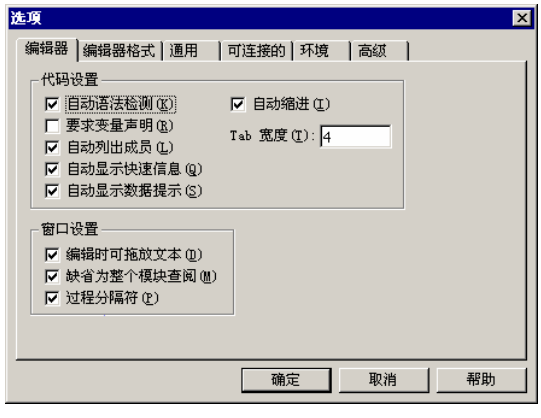


图2-22　【选项】对话框

2.5.1 设置【编辑器】

在【选项】对话框的【编辑器】选项卡中可以设置【代码】窗口和【项目】窗口的编辑特性，主要包括【代码设置】和【窗口设置】，如图 2-23 所示。

在【代码设置】选项框中可以设置有关快速信息的显示方式，下面列出了备选项的主

要功能。

- 【自动语法检测】：选中该选项后，Visual Basic 6.0 自动对输入的代码校验语法。
- 【要求变量声明】：决定模块中是否需要明确的变量说明。
- 【自动列出成员】：当输入一个控件名加上点操作符后，自动显示控件成员的列表清单。
- 【自动显示快速信息】：自动显示关于函数及其参数的信息。
- 【自动显示数据提示】：自动显示放置光标位置的变量值。
- 【自动缩进】：当设置了某一行缩进后，所有后续行都将以该缩进位置为起点。
- 【Tab 宽度】：设置制表符宽度，其范围可以是 1～32 个空格，缺省值是 4 个空格。

【窗口设置】选项框中主要用来设置【代码编辑器】中的几个基本特性，包括下面的内容。

- 【编辑时可拖放文本】：在当前代码窗口中，可以向【立即窗口】或者【监视窗口】内拖放代码文本。
- 【缺省为整个模块查阅】：选中这个选项后，用户可以用滚动条在代码窗口内查看多个过程，而不用改变过程名。
- 【过程分隔符】：选中这个选项后，在【代码】窗口中每个过程结尾处将显示分割线，注意只有当【缺省为整个模块查阅】被选中时它才起作用。

2.5.2 设置【编辑器格式】

选中【选项】对话框中的【编辑器格式】选项卡，进入【编辑器格式】设置对话框，可以设置【代码】窗口中文本的显示样式，如图 2-23 所示。在这个选项卡中可以设置文本的颜色、字体、大小等常用的属性。

(1) 设置文字颜色

在【颜色】设置框中首先选中需要设置文本颜色的文本类型，然后分别在【前景色】、【背景色】、【标识色】下拉列表中选中需要的颜色，即可设置该文本类型的文本颜色，然后利用上面的方法依次可以设置所有文字的颜色。

(2) 设置【字体】和【大小】

在【字体】和【大小】列表中选择对应文字的字体和大小即可设置编辑器中的文本的字体样式。注意，在编辑器中所有的文本的字体和大小是相同的，只有颜色可以不同。

(3) 【边界标识条】

选中该选项可以设置文本边界器中的有无边界标识条，可以在【选项】对话框右下方的【示例】区看到设置后的结果。

2.5.3 设置【通用】特性

选中【通用】选项卡后，可以设置当前项目中的【网格】、【错误捕捉】和【编译】选项设置，如图 2-24 所示。

(1) 【窗体网格设置】

这部分可以设置窗体设计器中网格设置的特性，这些特性包括是否显示网格等。下面是各功能项的使用简介。

- 【显示网格】：指定窗体设计器中是否显示网格的格线。
- 【网格单位】：设置网格的相邻格线之间的距离，其中【高度】和【宽度】文本框中分别可以设置网格单元的高度和宽度。
- 【对齐控件到网格】：控制窗体设计器中的控件是否按照网格线对齐。
- 【显示工具提示】：控制当鼠标移动到工具条中的按钮上时是否显示工具提示条。
- 【折叠工程，隐藏窗口】：选中这个选项后，当某个窗体在工程管理器中重叠时，会自动隐藏窗口。

图2-23　设置【编辑器格式】

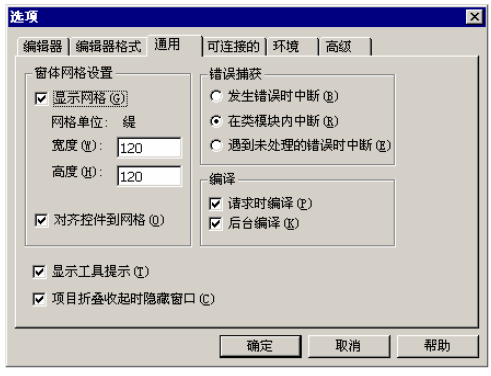


图2-24　设置【通用】属性

(2)　【捕获错误】

这部分可以指定应用程序运行时出错判断的条件，主要有 3 种方式。

- 【发生错误时中断】：指定程序在任何出现错误的地方进入中断状态。
- 【在类模块内中断】：指定程序在任何出现在类模块中的错误，都将导致工程在类模块中产生该错误的地方进入中断模式。
- 【遇到未处理的错误时中断】：当错误处理器没有被激活时，任何错误都会导致工程进入中断模式。而类模块中未经处理的错误，将导致工程在调用该类未结束过程的代码行上进入中断模式。

(3)　【编译】

【编译】选项可以指定 Visual Basic 6.0 何时以及如何编译应用程序。

- 【请求时编译】：指定 Visual Basic 6.0 是在启动一个工程之前完全编译它，还是根据需要来编译代码。选中了这个选项之后，则 Visual Basic 6.0 将编译了整个项目之后再开始运行它。
- 【后台编译】：决定在运行时是否使用空闲时间在后台完成对工程的编译。【后台编译】可以提高运行时的执行速度。注意，若要选择该选项，必须同时选中【请求时编译】命令，否则该特性是不起作用的。

2.5.4　设置【可连接的】窗口

选中【可连接的】选项卡，如图 2-25 所示，用户可以指定想要连接的窗口。

在 MDI 模式时，如果一个窗口被设置为和另一个【可连接的】窗口或者主窗口连接，当把两个窗口移动到一起时，它们就会连接在一起。在这里，所谓的连接就是指两个窗口合并成了一个窗口，在前面启动 Visual Basic 6.0 时，用户可以看到【工程管理器】窗口、【属性】窗口和【布局】窗口就是连接在一起的。

在该选项卡中可以将 9 种 Visual Basic 6.0 集成环境中的窗体连接起来。这包括【立即窗口】、【本地窗口】、【监视窗口】、【工程管理器】、【属性窗口】、【对象浏览器】、【窗体布局窗口】、【工具箱】和【调色板】。

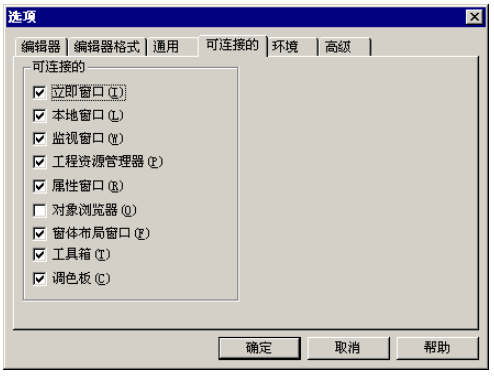


图2-25　设置【可连接的】窗口

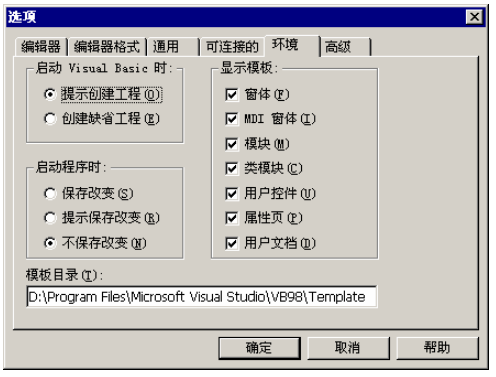


图2-26　设置【环境】属性

2.5.5　设置【环境】

在【环境】选项卡中，用户可以设置 Visual Basic 6.0 集成开发环境的各种属性，如图 2-26 所示。在这里对集成环境所做的设置，将会被保存到 Windows 的注册表中，在每次启动 Visual Basic 6.0 时都会调用这些设置。

在【环境】选项卡中主要包括 4 个方面的设置内容，分别介绍如下。

(1)　【启动 Visual Basic 6.0 时】

- 【提示创建工程】：选中这个选项后，每次启动 Visual Basic 6.0 时会出现新建窗口，提示用户打开需要的工程。
- 【创建缺省工程】：选中这个选项后，每次启动 Visual Basic 6.0 时，会自动打开一个缺省的可执行（EXE）工程。

(2)　【程序启动时】

- 【保存修改】：选中这个选项后，每次运行项目时，无需提示就可自动保存修改后的内容。这样就可以经常保存对 Visual Basic 6.0 集成环境的修改。
- 【提示保存修改】：选中这个选项后，每次运行项目时，总会显示一个对话框，询问是否保存修改的内容。
- 【不保存修改】：在运行工程时，Visual Basic 6.0 会运行内存中的工程版本，而不保存用户对项目做的任何修改。

(3)　【显示模板】

当向工程中添加项目时，在这里确定显示哪些模板。如果删除该模板选项，则在添加项目时，屏幕不会显示该项目对应的模板。

从图 2-26 所示中可以看出在 Visual Basic 6.0 中提供了 7 种可用的模板，分别是普通【窗体】、【MDI 窗体】、普通【模块】、【类模块】、【用户控件】、【属性页】、【用户文档】。

(4)　【模板目录】

在这里用户可以输入模板文件的完整路径，Visual Basic 6.0 集成环境会在这个路径下寻找模板文件并进行加载。

2.5.6　设置【高级】选项

在【高级】选项设置中，用户可以设置下面的内容，如图 2-27 所示。

- 【在后台加载工程】：确定是否在后台加载代码。当代码在后台加载时，可以迅速地将控制返回给用户。
- 【当改变共享工程项时提示】：当修改一个诸如窗体或模块这样的共享工程项而且还要保存它时，该选项卡决定系统是否给出提示。
- 【SDI 开发环境】：选择该项时，开发环境会从多文档界面（MDI）转换到单文档（SDI）界面。这时，每一个窗口都是独立的。一般默认情况是多文档界面。

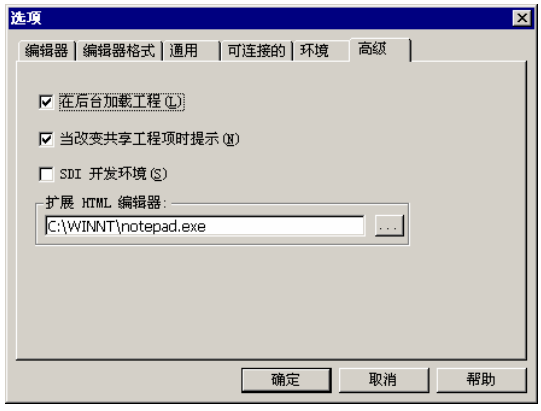


图2-27　设置【高级】选项

2.6　小结

本章主要介绍了 Visual Basic 6.0 集成开发环境。通过本章的学习，应当掌握 Visual Basic 6.0 集成开发环境的菜单栏、工具栏、【工程管理器】窗口、【属性】窗口以及【代码编辑器】的组成和使用方法。

我们主要学习了以下内容：

- Visual Basic 6.0 的菜单。
- Visual Basic 6.0 的工具栏。
- Visual Basic 6.0 的基本窗口。
- 集成开发环境中的选项设置及其自定义方法。

2.7　习题

一、 选择题

1. 以下叙述中错误的是（　　）。
   A. 在工程资源管理器窗口中只能包含一个工程文件及属于该工程的其他文件
   B. 以.BAS 为扩展名的文件是标准模块文件
   C. 窗体文件包含该窗体及其控件的属性
   D. 一个工程中可以含有多个标准模块文件
2. 下列不能打开属性窗口的操作是（　　）。
   A. 执行【视图】菜单中的【属性窗口】命令

B. 按 F4 键
C. 按 Ctrl+T
D. 单击工具栏上的【属性窗口】按钮 。

3. 下列可以打开立即窗口的操作是（ ）。
A. Ctrl+D
B. Ctrl+E
C. Ctrl+F
D. Ctrl+G

4. 在设计阶段，当双击窗体上的某个控件时，所打开的窗口是（ ）。
A. 【工程资源管理器】窗口　　B. 【工具箱】窗口
C. 【代码】窗口　　D. 【属性】窗口

5. VB 的【工程资源管理器】可管理多种类型的文件，下面叙述不正确的是（ ）。
A. 窗体文件的扩展名为“.frm”，每个窗体对应一个窗体文件
B. 标准模块是一个纯代码性质的文件，它不属于任何一个窗体
C. 用户通过类模块来定义自己的类，每个类都用一个文件来保存，其扩展名为“.bas”
D. 资源文件是一种纯文本文件，可以用简单的文字编辑器来编辑

6. 如果要向工具箱中加入控件的部件，可以利用【工程】菜单中的（ ）命令。
A. 引用　　B. 部件　　C. 工程属性　　D. 加窗体

7. 工程文件的扩展名是（ ）。
A. .vbg　　B. .vbp　　C. .vbw　　D. .vbl

8. 通过（ ）窗口可以在设计时直观的调整窗体在屏幕上的位置。
A. 代码窗口　　B. 窗体布局窗口　　C. 窗体设计窗口　　D. 属性窗口

二、 填空题

1. 在打开 Visual Basic 6.0 集成开发环境时，可以看到 9 种应用程序类型：_____
_____。

2. 菜单项的右边有一个小黑箭头，表示_____；菜单项的颜色变暗，表示该菜单项_____。

3. 显示【添加工程】对话框，以便能把新的或已有的工程添加到当前打开的工程组中。需要执行_____菜单的_____菜单命令。

4. 如果用户要向工具栏中添加下拉式工具按钮，在【类别】列表柜中选择_____。

5. _____又称工程资源管理器，是在 Visual Basic 6.0 集成开发环境中用来管理工程的一个窗体。

6. 代码窗口主要由_____、_____、_____、_____、_____组成。

三、 问答题

1. Visual Basic 6.0 集成开发环境主要包括哪些组成都分？

2. 利用 Visual Basic 6.0 可以创建哪几种类型的文件？

3. 如何创建用户自定义的工具栏？

4. 【工程管理器】可以管理那些类型的文件？

第3章　创建第一个简单的应用程序

在了解 Visual Basic 6.0 集成开发环境之后，本章将通过一个简单的例子介绍创建 Visual Basic 6.0 应用程序的过程，为以后的可视化编程打下基础。

本章学习目标

- Visual Basic 6.0 中语句的使用。
- Visual Basic 6.0 简单应用程序的创建。
- 用户界面的设置。
- 属性的设置。
- 应用程序代码的编写。

3.1 Visual Basic 6.0 中的语句

Visual Basic 6.0 中的语句是执行具体操作的指令，每个语句以回车符结束。如果设置了“自动语法检测”(在【工具】菜单【选项】命令对话框中的【编辑器】选项卡中设置)，则在输入语句的过程中，Visual Basic 6.0 将自动对输入的内容进行语法检查，如果发现了语法错误，则弹出一个信息框，提示出错的原因。

Visual Basic 6.0 会按规则对于语句进行简单的格式化处理，例如方法的第一个字母大写，运算符前后加空格等。在输入语句时，方法、函数等可以不必区分大小写，例如，在输入 MsgBox 时不管输入 MsgBox、msgbox 还是 MSGBOX，输入完按回车键后都变为 MsgBox。为了提高程序的可读性，在代码中应加上适当的空格。

在编写 Visual Basic 6.0 程序时，最好一行里只写一条语句，如果一条语句太长，需用续行符“_”把一个长语句分成若干行来存放，续行符与它前面的字符之间至少要有一个空格。在 Visual Basic 6.0 中，允许使用复合语句，即如果要把几个语句放在一行中，各语句之间用“:”隔开。

在这一节中，将介绍 Visual Basic 6.0 中的几个常用语句。

3.1.1 注释语句

为了提高程序的可读性，通常应在程序的适当位置加上必要的注释。在 Visual Basic 6.0 中，注释语句有两种，一种是用“Rem”关键字，还有一种是利用单引号“'”。

格式：

```
Rem 注释内容
' 注释内容
```

例如：

```
'Dim a As String
```

与

```
Rem Dim a As String
```

的作用是一致的。

注释不仅可以对程序进行解释，有时它还可以用在程序的调试中，譬如说可以利用注释屏蔽一条语句以观察程序的变化，发现问题和错误。以后注释语句将是我们在编程里最经常用到

的语句之一。

3.1.2 赋值语句

赋值语句的作用是将指定的值赋给某个变量或某个带有属性的对象，其一般格式为：

目标操作符 = 原操作符

这里的“目标操作符”包括变量、表达式、常量及带有属性的对象；而“原操作符”指的是变量和带有属性的对象；“=”称为“赋值号”，它与数学上的等号意义不一样。赋值语句兼有计算和赋值双重功能，它首先计算赋值号右边“原操作符”的值，然后把结果赋给赋值号左边的“目标操作符”。例如：

```
first=1          把数值常量 1 赋给变量 first
second=15+3      把表达式的值赋给变量 second
Text1.Text=a     把变量 a 的值赋给带有 Text 属性的对象 Text1
A= Text1.Text    把带有 Text 属性的对象 Text1 值赋给变量 A
Text2.Text= Text1.Text   把带有 Text 属性的对象 Text1 赋给带有 Text 属性的对象
                         Text
```

3.1.3 结束语句

用来终止程序的运行，常用的语句为 end。可以把它放在事件过程中，例如：

```
Sub Command1_Click( )
    End
End Sub
```

该过程用来结束程序，即单击命令按钮时，结束程序的运行。

End 语句除了用来结束程序外，在不同的环境下还有其他用途，如：

End Sub	结束一个 Sub 过程
End Function	结束一个 Function 过程
End If	结束一个 If 语句快
End Select	结束情况语句

3.2 Visual Basic 6.0 简单应用程序的介绍和创建

在做每一件事情之前，一般都要明白为什么要做这件事情，做这件事情想要产生一个怎样的效果，达到一个什么样的目的。同样，程序员在创建应用程序之前，必须要了解这个应用程序要做什么用，实现什么样的功能，有什么要求等基本情况，然后才能按要求创建应用程序。

3.2.1 加法计算器简介

加法计算器是一个简单的 Visual Basic 6.0 应用程序，如图 3-1 所示，这个应用程序提供了简单的加法计算功能。通过学习这个程序的编制过程，读者可以了解到 Visual Basic 6.0 编程的最基本的方法和步骤。

在加法计算器中，当在“文本框 1”和“文本框 2”中分别输入两个数后，单击 加法 按钮实现这两个数的加法运算，并将运算结果在“文本框 3”中显示，如图 3-1 所示；当单击 清除 按钮时，将 3 个文本框清空；当单击 关闭 按钮时，关闭对话框，退出加法计算器，回

到 Visual Basic 6.0 集成化开发环境。

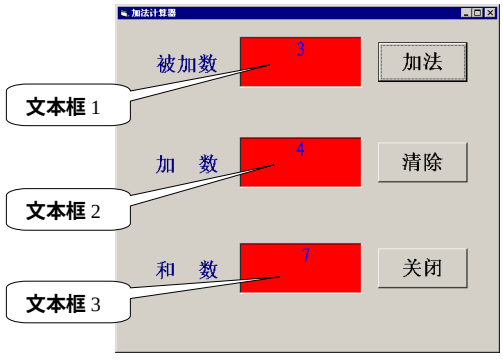


图3-1　加法计算器界面

3.2.2　加法计算器应用程序的创建

Visual Basic 6.0 是一种可视化的开发工具，用 Visual Basic 6.0 创建应用程序是比较容易的。但是人们对事物的了解总是有一个循序渐进的过程，学习 Visual Basic 6.0 也是一样的，应该从最基本的程序开始，逐步提高。所以弄明白第一个应用程序的创建过程，对后面的深入学习将会奠定良好的基础。

要创建加法计算器，首先应当创建一个新的应用程序，其步骤如下。

1. 在【文件】菜单中，单击【新建工程】命令，屏幕上会出现一个对话框，如图 3-2 所示。

图3-2　【新建工程】对话框

2. 在该对话框中选择“标准 EXE”，单击 打开(O) 按钮。这时 Visual Basic 6.0 集成环境将创建一个名为【工程 1】的工程，并且在窗体设计器自动创建一个名为“Form1”的窗体文件。
3. 打开【文件】菜单，单击【保存工程】命令，屏幕会出现一个【文件另存为】对话框，要求用户保存当前的窗体文件，在【文件名】文本框中输入“加法计算器”，然后单击 保存(S) 按钮，如图 3-3 所示。

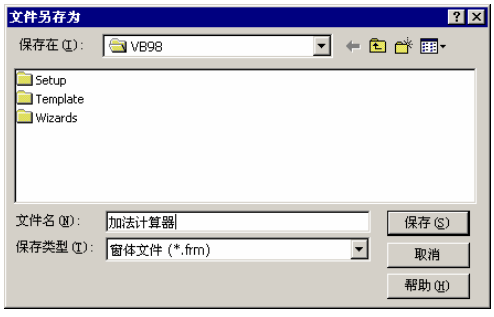


图3-3 【文件另存为】对话框

4. 在存储了窗体文件后，集成环境会要求用户存储工程文件，按照上一步的操作，将新建的工程保存为“加法计算器”工程文件。

这样，一个新的工程就创建完毕了，下面的任务就是向窗体中添加控件和代码。注意在创建新工程的时候，并不一定要先存储所建的工程，这里只是提醒用户在编制 Visual Basic 6.0 程序时，要注意存盘，以免出现意外情况。

3.3 设计 Visual Basic 6.0 应用程序用户界面

在完成编制 Visual Basic 6.0 应用程序的第一步（创建新的工程）后，就要进入第二步，即设计应用程序界面。在 Visual Basic 6.0 应用程序设计中，设计应用程序界面是一个关键部分，这也是 Visual Basic 6.0 编程可视化的具体表现。

3.3.1 初步创建加法计算器界面

设计加法计算器的界面可以按照下面的步骤进行。

1. 单击工具箱中的标签控件A（Label），然后把鼠标移到窗体上，这时鼠标指针变成了一个“+”字形状，“+”字光标用于绘出命令按钮的矩形外框。在适当的位置按下鼠标左键，并拖曳鼠标，如图 3-4 所示，此时标签对象就是虚线框的大小，当标签对象的大小合适时，停止拖曳鼠标指针，然后放开鼠标左键，这时窗体上就会在虚线框的位置出现一个标签，标签中会自动显示“Label1”的字符串。

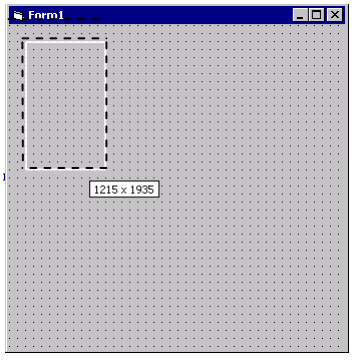


图3-4 添加标签控件

2. 同上步一样，在窗体上再添加两个标签控件。它们的位置可以根据图 3-1 中各标签的位置确定。

每一个新添加的标签的名称，及其在标签中的显示都是“ Label ”再加上当前存在的标签数。例如，加入第 2 个标签时，其名称就是“ Label2 ”。这个规则在 Visual Basic 6.0 集成环境中是通用的。无论是新添工程、新添窗体还是新添控件，只要用户没有为它们重新命名，则用其默认的名称，这些新添的文件名称都是该文件或控件的名称再加上当前文件或控件的数目。

3. 在【工具箱】窗口中，选择文本框控件（TextBox），在 3 个标签旁边分别添加 3 个文本框。其添加方法与标签添加方法类似。
4. 在【工具箱】窗口中，选择命令按钮控件（CommandButton），按照上步的方法，添加 3 个命令按钮，其位置可以按照图 3-1 所示中命令按钮的位置确定。这样加法计算器器的界面就初步完成了，如图 3-5 所示。这时界面还不美观，下面就要调整控件布局，美化窗体界面。

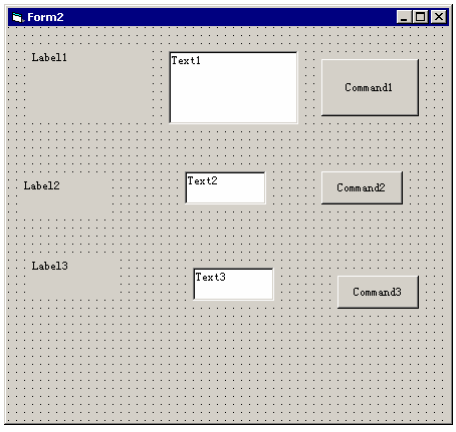


图3-5　加法计算器的初步界面

5. 用鼠标单击第一个标签，这时“ Label1 ”标签控件周围就会出现 8 个蓝色的小方块，代表控件被选中。这 8 个小方块代表该控件可调整的 8 个方向，将鼠标移动到方块上，此时鼠标形状变为双箭头形状，表示可以改变控件的大小，按住鼠标左键然后拖动，这时会出现一个虚线框，这个虚线框与创建控件时的虚线框相同。将鼠标拖动到合适的大小后，松开鼠标左键就可以改变控件的大小。

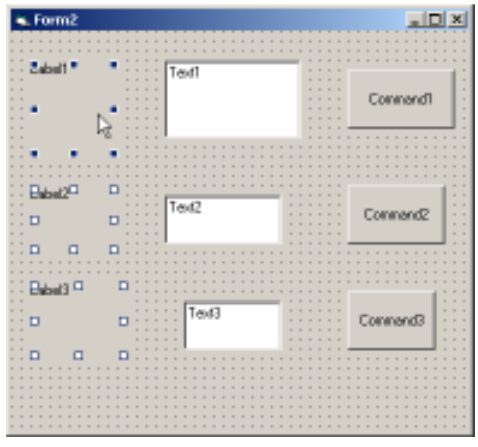


图3-6　选择一组控件

6. 选中“ Label1 ”标签控件，按住 Shift 键，然后单击“ Label2 ”和“ Label3 ”标签控件，就可选中这 3 个标签控件组，如图 3-6 所示。在这一组被选中的控件中，不是所有的控件周围的小方块都是蓝色，只有最后一个被选中的“ Lable3 ”控件是蓝色的，其他控件周围都是白色的小方块。这个最后被选中的“ Label3 ”控件被称为基准控件，要调整 3 个标签的大小及位置，就要以这个基准标签控件为基准。如果要以第一个标签为基准，则可按住 Shift 键双击第一个标签即可。

选择控件和控件组的方法有两种：“单击法”和“区域法”，其中第（5）步和第（6）步采用的是“单击法”。“区域法”选择控件方法很简单，将鼠标移动到窗体中合适的位置，按住鼠标左键，然后拖动鼠标，这时在窗体中会出现一个虚线框，这个虚线框所圈住的控件都会被选中，如图 3-7 所示。一般选择一组控件时都是使用区域法选择的，也可以两种方法配合使用，先用区域法选择大多数控件，然后再用单击法选择其余控件。

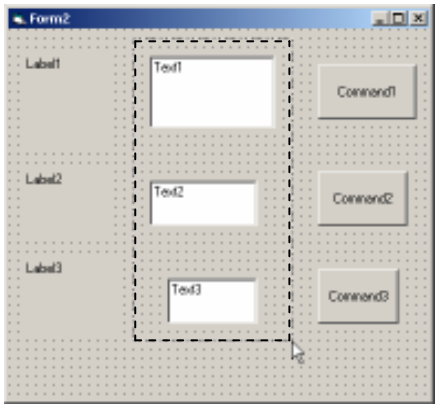


图3-7 “区域法”选择一组控件

7. 选择【格式】/【统一尺寸】/【两者都相同】命令，这时 3 个标签会自动调整为和“Label1”大小相同。
8. 选择【格式】/【对齐】/【居中对齐】命令，这时 3 个标签就会自动居中对齐。
9. 选择【格式】/【垂直间距】/【相同间距】命令，这时 3 个标签的垂直方向的间距会自动调整至相同。
10. 按照第 5 步的方法，调整“Text1”到合适的大小。
11. 按照第 6 步的方法，选择“Text1”、“Text2”和“Text3”3 个文本框，并保证以“Text1”为基准。然后按照第 7 步和第 8 步的方法，调整 3 个文本框的布局。
12. 同样方法调整“Command1”、“Command2”和“Command3”3 个命令按钮的布局。
13. 选择“Label1”、“Text1”和“Command1”3 个控件，并保证以“Label1”为基准。
14. 按照第 13、14 步的方法，调整其他控件的水平方向的位置。
15. 选择【格式】/【锁定】命令，将设置后的结果固定，这时用户如果选择控件后，就不能再进行调整了。这样所有的布局设置就完成了，完成后的结果如图 3-8 所示。

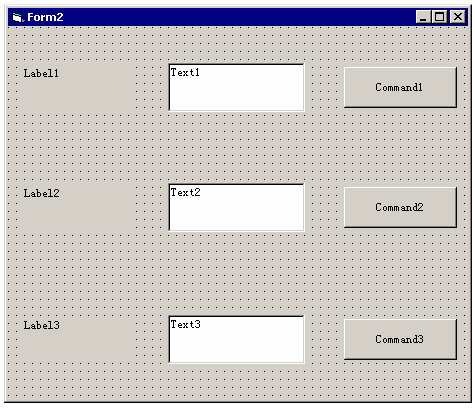


图3-8 调整布局后的界面

3.3.2 设置属性

调整完布局之后，可以看到加法计算器的界面已经初步形成了，但是许多地方还是不完

善，例如标签控件上的文字的字体、位置以及颜色，文本框的显示字体，这些都要通过属性设置来实现。

在这里首先简单介绍一下属性的概念，属性是每个控件对象所固有的特性。就像每个人，它有姓名、年龄、身高、体重等属性，这些属性就构成了人这个对象，但是每个人的姓名、年龄、身高、体重的属性具体值是不相同的，这就是属性值。用户在窗体中创建的所有控件，都有其缺省的属性值，如标题、文字、颜色等，但这些属性值并不能完全满足要求，这时就必须对控件的属性进行设置。

设置控件属性是在【属性】窗口中完成的，当用户在窗体选中某个控件之后。在【属性】窗口中就会列出该控件的所有属性和属性值。设置属性时，单击属性列表中的某个属性，在其右边对应的属性值框中输入该属性的属性值或者选择属性值列表中的一项，就完成了该属性的设置，如图 3-9 所示。

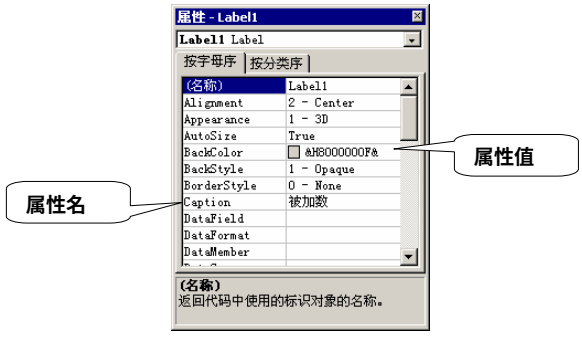


图3-9　设置属性值

了解了设置属性的过程后，就可以开始为加法计算器设置属性了。其设计步骤如下。

1. 按照表 3-1 所示的各控件对象的相关属性值，设置这个对象的属性，其设置结果如图 3-10 所示。

表 3-1　　对象属性设置

对象	属性名	属性值
窗体（Form）	Caption	加法计算
标签 1（Label1）	Alignment	2 – Center
	AutoSize	True
	Caption	被加数
标签 2（Label2）	Alignment	2 – Center
	AutoSize	True
	Caption	加　数
标签 3（Label13）	Alignment	2 – Center
	AutoSize	True
	Caption	和　数
文本框 1（TextBox1）	Alignment	2 – Center
	Text	（空）
文本框 2（TextBox2）	Alignment	2 – Center
	Text	（空）

续表

对象	属性名	属性值
文本框 3（TextBox3）	Alignment	2 – Center
	Text	（空）
命令按钮 1（CommandButton1）	Alignment	2 – Center
	Caption	加法
命令按钮 2（CommandButton2）	Alignment	2 – Center
	Caption	清除
命令按钮 3（CommandButton3）	Alignment	2 – Center
	Caption	关闭

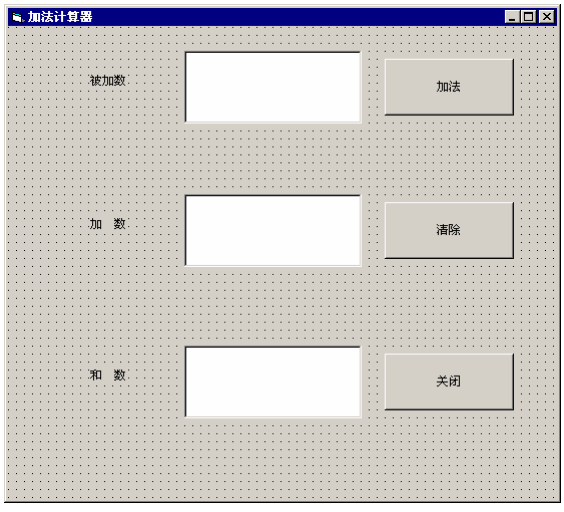


图3-10　设置一部分属性后的界面

2. 选中“ Label1 ”标签控件，在【属性】窗口选择 Font 属性，在【Font】属性值后有一个...样的小按钮，单击该按钮后便会出现【字体】对话框如图 3-11 所示，默认的【Font】属性为“ 宋体 ”、“ 常规 ”、“ 小五 ”号。

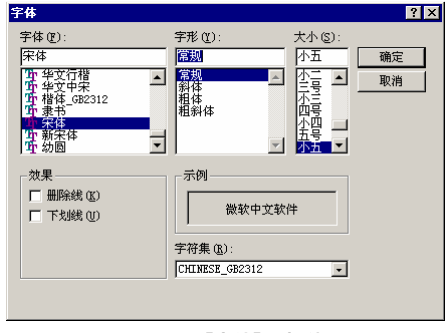


图3-11　【字体】对话框

3. 在【字体】对话框中的【字形】列表中选择“ 粗体 ”，在【大小】列表中选择“ 二号 ”字，然后单击 确定 按钮设置文字属性。
4. 按照第 2、3 步的方法，将其余 2 个标签、3 个文本框和 3 个命令按钮的“ Font ”属性设置相同的值。
5. 选中“ Text1 ”文本框控件，在【属性】窗口选择【BackColor】属性，在属性值框中单击右边的下拉按钮，会出现一个【颜色】列表，单击【调色板】对话框按钮，【颜色】列表变为调色板，如图 3-12 所示。单击红色板块，将文本框中的文字背景颜色属性设置为“ 红色 ”。
6. 按照第 5 步的方法，将其余两个文本框的背景颜色设置为“ 红色 ”。

7. 选中“Text1”文本框控件，在【属性】窗口选择【FontColor】属性，同第 5 步设置背景颜色方法相同，设置其字体颜色为“蓝色”。
8. 同样，设置其余两个文本框以及 3 个标签的【FontColor】属性。这样主界面中的控件属性就设置完毕了。其设置结果如图 3-13 所示。

图3-12　设置字体颜色

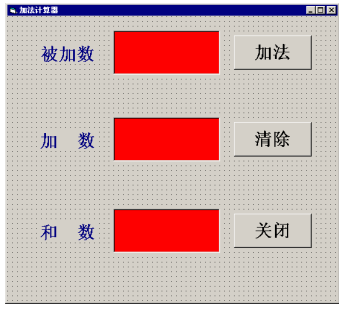


图3-13　窗体设置结果

3.4 编写代码并运行程序

应用程序的界面就好比人的外表，而代码则好比人的大脑及灵魂。仅仅有优美界面的应用程序，就好比只有躯壳而没有思维的一个植物人，不能算一个完整的应用程序。要使一个应用程序能够执行一定的操作，完成一定的功能，就必须给它编写代码。

Visual Basic 6.0 编程是面向对象的、可视化的、基于事件驱动的编程机制。大部分的代码都是和对象的事件相连的。在这里首先介绍一下事件的概念，事件可以简单地理解为外界对对象的刺激，例如用户用鼠标单击了一个按钮，这时就出现了单击事件（C1ick）；如果是双击了按钮，就出现了一个双击事件（DblClick）。当激发了一个事件后，就调用执行相应事件的代码，所针对的对象就会产生相应的操作。在 Visual Basic 6.0 程序中，基本上就是采用这种事件触发的机制，而不是 DOS 那种流程机制，但是在每个程序段内还是流程机制。

现在开始向加法计算器添加代码，可以按照下面的步骤进行。

1. 在工程管理器中双击“Form1”，在窗体设计器中出现加法计算器的主界面。
2. 在加法计算器主窗口中用鼠标双击命令按钮“Command1”，屏幕上会出现【代码】窗口，并且光标在命令按钮“Command1”的单击事件“Click”内跳动。在光标跳动的地方，即命令按钮 1 的单击事件“Command1_Click”内，编写如下代码：

```
Private Sub Command1_Click()
    Text3.Text = Str$(Val(Text1.Text) + Val(Text2.Text))
End Sub
```

3. 同样，在“Command2_Click”事件内编入以下代码：

```
Private Sub Command2_Click()
    Text1.Text = ""
    Text2.Text = ""
    Text3.Text = ""
End Sub
```

4. 在“Command3_Click”事件内编入以下代码：

```
Private Sub Command3_Click()
```

```
        End
    End Sub
```

事件过程的首尾两行（粗体）：

Private Sub Command1_Click()

End Sub

是系统自动给出的代码，不必重复输入。

单击工具栏上的【启动】按钮，运行工程如图 3-1 **所示。如果工程中有错误，就会出现如图** 3-14 **所示的【错误提示】对话框。**

图3-14 **【错误提示】对话框**

单击 确定 **按钮，回到代码编辑界面，出错的地方会用蓝色高亮度显示，可根据提示修改代码，系统会继续在运行中检查直到能正常运行为止。**

设计好的应用程序在调试正确后需要保存工程，即以文件的方式保存到磁盘上。常用下面的方法保存工程：

- **单击【文件】菜单中的【保存工程】或【工程另存为】。**
- **单击工具栏上的【保存工程】按钮。**

系统则打开如图 3-15 **所示的【工程另存为】对话框。**

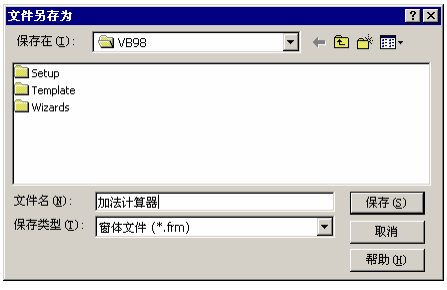


图3-15 **【工程另存为】对话框**

由于一个工程可能含有多种文件，如工程文件和窗体文件，这些文件集合在一起才能构成应用。

在【工程另存为】对话框中，注意到保存类型，保存窗体文件（*.frm**）到指定文件中。窗体文件存盘后系统会弹出【工程另存为】对话框，保存工程文件（***.vbp**）到指定文件夹。**

3.5 小结

本章通过一个应用程序的编制，介绍了如何创建一个应用程序，如何设计程序的界面，如何设置控件的属性，以及如何编制代码，使读者对 Visual Basic 6.0 **编程的过程有了一个初步的了解，对于** Visual Basic 6.0 **事件驱动的编程机制也有了一个初步的概念。在后面的章节将详细地介绍** Visual Basic 6.0 **编程的特点和方法。**

3.6 习题

一、 选择题

1. Visual Basic 6.0 的赋值语句的作用为（　）。
   A. 兼有计算与赋值双重功能
   B. 计算
   C. 赋值
   D. 比较与赋值
2. 为了把窗体上的某个控件变为活动的，应执行的操作为（　）。
   A. 单击窗体的边框
   B. 单击该控件的内部
   C. 双击该控件
   D. 双击窗体
3. 假定已在窗体上画了多个控件，并有一个是活动的，为了在属性窗口中设置窗体的属性，预先应执行的操作为（　）。
   A. 单击窗体上没有空件的地方
   B. 单击任一个控件
   C. 不执行任何操作
   D. 双击窗体的标题栏
4. 为了同时改变一个活动空间的高度和宽度，正确的操作是（　）。
   A. 控件 4 个角上的某个小方块
   B. 只能拖拉位于控件右下角的小方块
   C. 只能拖拉位于控件左下角的小方块
   D. 不能同时改变控件的高度和宽度

二、 填空题

1. Visual Basic 6.0 程序中最好一行里只写一条语句，如果一条语句太长，需用_____把一个长语句分成若干行来存放。
2. 单击工具箱中的标签控件 A（Label），然后把鼠标移到窗体上，这时鼠标指针变成了一个 _____字形状。
3. 为了选择多个控件，可以按住_____键，然后单击每个控件。
4. _____是每个控件对象所固有的特性。

三、 编程题

在窗体上添加 3 个文本框、3 个标签和 3 个命令按钮，并把 3 个命令按钮的标题设置为“减法”、“清空”和“退出”，文本框和标签的字体设为三号隶书，颜色为紫色。运行应用程序时，单击“减法”命令按钮，实现文本框 1 种输入的数值减去文本框 2 中输入的数值，并把结果显示在文本框 3 中。

第4章　Visual Basic 6.0 编程基础

本章介绍运用 Visual Basic 6.0 编程的基础知识，其中包括编程语言的基础知识以及 Visual Basic 6.0 的面向对象的概念。在学习本章之后，读者可以具有一定的 Visual Basic 6.0 编程的基础知识，为以后的编程打下基础。

本章学习目标

- Visual Basic 6.0 程序框架导论。
- Visual Basic 6.0 中的 3 个基本概念：对象、属性、方法。
- 编程语言中的数据类型。
- 变量的概念和使用。
- 编程中的表达式及其运算符。
- Visual Basic 6.0 语言中数据的另外存储表达方式—数组。
- 程序设计基本结构的基本表达和使用。
- 过程和函数的定义及其使用。
- 参数的传递。

4.1 Visual Basic 6.0 程序框架导论

Visual Basic 6.0 是一种模块化的语言，也是一种面向对象的开发工具，在 Visual Basic 6.0 工程中主要有 4 种项目类型，分别是窗体、多文档窗体、模块、类模块。其中，窗体文件和多文档窗体文件都是程序的界面接口，也就是说通过这两种文件类型来建立应用程序的用户界面。每个窗体文件和多文档窗体文件都包含许多事件过程，在这里可以编写响应特定事件而执行的代码。除了事件过程，窗体模块还可以包含通用过程，它是一种模块级过程，即在通用过程中声明的变量，只能被本窗体模块中的事件过程调用。模块文件相当于用户的程序库，用户可以将常用的函数和过程在模块文件中定义为公用代码。而类模块文件相当于用户的自定义对象库，在类模块文件中用户可以编写自定义对象，类模块在一定程度上和普通控件类似。

从上面的阐述中可以看出，Visual Basic 6.0 的程序结构是一种完全模块化的程序结构。在 Visual Basic 6.0 程序中，最小的程序模块是过程或者函数，这些过程或者函数从属于不同的窗体文件、多文档窗体文件、模块文件和类模块文件。这些文件之间是相对独立的，它们都可以独立运行。图 4-1 所示说明了 Visual Basic 6.0 的程序结构图。

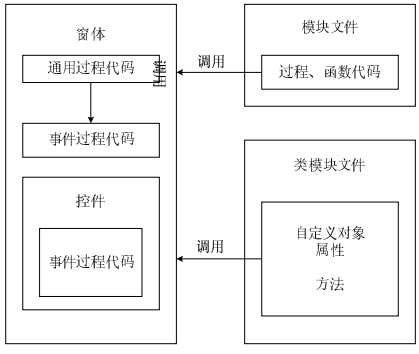


图4-1 程序结构图

4.2 对象和类的概念

Visual Basic 6.0 是一种面向对象（Oriented-Object）的可视化开发工具。不仅提供了大量的控件对象，而且还提供了用户自定义对象的方法和工具。在介绍 Visual Basic 6.0 面向对象的程序设计之前，先介绍一些面向对象的基本概念。

4.2.1 对象的定义

对象是现实生活中很常见的。比如，一个人是一个对象，一台计算机是一个对象，一辆汽车也是一个对象，对象是有某些特性的具体的事物的抽象。每一个对象包含了许多属性，例如，一台计算机包括显示器、键盘、鼠标等属性。对象也可以分为很多个更小的对象，例如，显示器也是一个对象，它包含了显示屏、数据线等属性。

编程中的对象是将数据和方法包装在一起而形成的一些实体或者说是一种数据结构，它使这些实体变得独立，当外界必须和对象发生关系时，便可以通过预先设定好的渠道进行交流，这些渠道就是所谓的方法，通过对象的方法可以和对象发生对话。例如，一个矩阵对象包括了矩阵行和列的数据，另外还包括矩阵复制、矩阵乘法、加法等方法，用户使用的时候可以通过矩阵提供的方法和它打交道，从而了解具体的实现方法。

4.2.1.1 对象的属性

对象的属性用来标识一个对象的所有特征，它实际上就是对象所属类的成员变量。对象属性属于对象的数据部分，例如控件的颜色、大小、字体都是对象的属性。大多数对象属性是在对象生成时自动设置的，用户可以在设计时通过【属性】窗口或运行时通过代码来改变属性。

Visual Basic 6.0 对窗体中新近创建的控件对象都会赋予缺省属性，这些在高级语言编译环境中都是有相同之处的。例如，缺省的“Name”属性就是控件的名称再加上一个序号。这种缺省属性实际上是这种控件对象所对应的类中定义的成员变量的缺省值。

控件对象的属性中，有些属性只能在设计中设置，有些属性只能在运行时设置。例如，列表框中的"Text"属性，这个属性返回当前列表框中所选项目的内容，在设计时设置是没有用的，因为在运行时这个值会经常改变。但是对应地看来，文本框中的"Multiline"属性必须在设计时设置，这个属性是设置文本框中的文本能否多行显示，在运行时设置是没有用的。

属性又分为只读属性和可读写属性。只读属性在运行时使用，如 ListBox（列表框）中的“ListCount”属性，它就是只读属性，在运行时只能选取列表中的项目数，而不能被设置。

在 Visual Basic 6.0 的对象控件中，有许多属性是所有的控件都具有的，熟练掌握 Visual Basic 6.0 编程的一个前提就是熟悉每一个控件的属性。但是每个控件都会有很多属性，应该先掌握一些控件的公共属性，然后再熟悉每个控件的特有属性，这样才能熟练掌握每个控件的使用方法。

4.2.1.2 对象的方法

对象的方法指的就是对对象可以进行的操作，举例来说，对于一个窗体对象，可以利用 Clear 方法来消除窗体中的文字和图形，利用 Hide 和 Show 方法控制窗体的隐藏和显示。对象的方法实际上是在控件对象的类中定义的一些成员函数，在生成一个对象实例的时候就可以利用对象类的方法。

在 Visual Basic 6.0 中，所有控件的方法都是有一定含义的动词。通过这个动词用户就可以了解控件方法的含义（注意，这里和下面变量的取名可以联系起来，取名合理将易于理解和

维护)。在调用方法时，如果有参数，在方法后加上参数值，参数中间用空格隔开。同控件的属性一样，控件也有许多常用的方法。方法的使用也可类比于函数。

下面列出控件所有的常用方法：

- C1ear：清除控件中的内容。如果对象是列表框，Clear 用于删除控件中所有的项目，如果是剪贴板则清除其中的内容。
- Drag：用于控件开始、结束或取消拖动操作。只有当对象的“ DragMode ”属性置为手工时，才需要使用 Drag 方法控制拖动操作。但是，也可以在“ DragMode ”属性设置为自动的时候使用 Drag。当其参数为 0 时表示取消移动操作，其参数为 1 时表示开始移动操作，参数为 3 时表示结束移动操作。
- Move：用于移动控件并且改变控件的尺寸，该方法的语法格式为：

```
object.Move  left，top，width，height
```

其中“ left ”参数是必须有的。

要指定任何其他的参数，必须先指定出现在语法中该参数前面的全部参数。例如，如果不先指定“ 1eft ”和“ top ”参数，则无法指定“ width ”参数。任何没有指定的参数都保持原来的数值不变。

- OLEDrag：用于开始一个部件的 OLE 拖动操作。
- Refresh：用于一个窗体或控件重新刷新。
- Setfocus：用于将焦点移动到指定的窗体或控件，但是该控件的“ Enable ”和“ TabStop ”属性必须设置为“ True ”。
- ShowWhatThis：用于显示 Windows Help 所提供的“ 这是什么 ”弹出式帮助菜单，帮助的内容是在“ HeIpContextlD ”属性所设置的帮助文件。
- Zorder：用于设置窗体和控件的图层位置，在设计时可以通过【格式】菜单中【置前】和【置后】命令设置控件的图层位量。

4.2.2 类的定义

类是一个最基本的程序语言概念，可以说是建立一个对象的模子，同一个模子能建立具有相同特征的对象，只要先定义了一个类，此类便是建立一个对象的依据。类和对象的关系很密切。类包含了有关对象的特征和行为信息，它是对象的蓝图和框架。每个对象都有其具体属性，但类不会对具体属性赋值。例如，电话是一个类，但具体到每一部电话就是对象了，它有一定的颜色和大小，即每一个属性，可以定义为类中的成员变量。

类中还具有一些特殊而重要的特征，这些特征对提高代码的可重用性和易维护性很有用处。这些特征分别是封装、子类、继承性。详细介绍如下：

- 类的封装特性，就是隐藏不必要的复杂性，它将对象的方法、程序和属性代码包装在一起，这样一来，用户只用关心类的输入输出，而不必知道其具体实现。
- 定义子类是减少代码的一条途径。父类高于子类，父类的属性在子类中都有体现，这易于实现合理地代码维护。继承性的概念是使在一个类上所做的改动反映到它的所有子类当中。这种自动更新节省了用户的时间和精力。
- 继承性只体现在软件中，而它不可能在硬件中实现。若发现类中存在一个小小的错误，用户不必逐一修改子类中的代码，而只需要在类中改动一处，然后这个变动将会涉及全部的子类。

4.2.3　模块和工程介绍

一个 Visual Basic 6.0 应用程序由若干个程序模块组成，每个程序模块都以文件形式存储。

4.2.3.1　模块（Module）

Visual Basic 6.0 的代码存储在模块中。模块有 3 种类型：窗体模块、标准模块（通用模块）和类模块。简单应用时，可以只有一个窗体。应用程序的所有代码都驻留在窗体模块中，文件扩展名为“.frm”。窗体文件既有代码，又有对象和属性设置。而当应用程序庞大复杂时，就要另加窗体，最终可能会发现在几个窗体中都有要执行一个或几个的公共代码。因为不希望在两个窗体中存在重复代码，所以要创建一个独立模块，它包含实现公共代码的过程。

- 窗体模块（文件扩展名为“.frm”）

窗体模块是 Visual Basic 6.0 应用程序的基础。一般每一个窗体都有自己的窗体模块。每个窗体模块中可以包含若干个过程（对象—事件过程、通用过程）以及窗体级的声明。

窗体是一种容器，其中可以包含许多控件。在窗体文件中，每个控件也都有一个对应的事件过程集，这些事件所对应的过程集是从属于窗体文件的。每个窗体文件都包含许多事件过程，在这里可以编写响应特定事件而执行的代码。除了事件过程，窗体模块还可以包含通用过程，它是一种局部公用过程，也就是说，只有在窗体中的所有事件过程才可调用这些通用过程代码。

一个窗体模块中的代码可以引用其他窗体中对象的属性和方法，但引用时，应使用：“窗体名.对象名.属性名”或“窗体名.对象名.方法名”。应用本窗体的对象时，无需指出窗体名。

- 标准模块（或通用模块，文件扩展名为“.bas”）

有时，在多个窗体模块中需要执行相同的公共代码。如果在每个窗体模块中重复编写这些代码，显然这不是好的方法。在 Visual Basic 6.0 中，可以将这种公共代码放到一个独立的模块中，在各个窗体模块中都可以调用这种公共代码。这种独立的模块称为标准模块（或通用模块）。通用模块并不属于任何一个窗体。开发者可以利用【工程】/【添加模块】菜单来编写通用模块。一个应用程序可以有若干个通用模块。甚至通用模块不一定绑定在特定的应用程序中，还可以在某个共享的模块库中。它是应用程序内其他模块访问的过程和声明的容器，可以包含变量、常数、类型、外部过程和全局过程的全局声明和模块级声明。

- 类模块（文件扩展名为“.cls”）

类模块是面向对象编程的基础。Visual Basic 6.0 中含有很多预定义的类模块，例如：命令按钮类、文本框类等，用户就是根据这种类模块来创建对象的。Visual Basic 6.0 允许用户自定义类模块，再根据这种类模块来创建自定义的对象，还可以为自定义的对象定义属性、定义事件以及添加方法。类模块在一定程度上和普通控件类似，例如它们都有自己的属性、可以响应的事件、可以执行的方法等。但是普通的控件或者窗体都是有其图形界面的，而类模块是没有的。可以把类模块看作是没有物理表示的对象。一个应用程序中可以有若干个自定义的类模块。

利用【视图】/【对象浏览器】菜单查看 Visual Basic 6.0 中预定义的、自定义的以及当前工程创建的各个对象类的属性和方法，选择某个属性或方法后，还能查看其有关的说明。

4.2.3.2　工程

一个工程是一个应用程序中所有相关模块的集合。多个相关的工程在一起构成工程组。工程

管理器窗口是用来显示与 Visual Basic 6.0 工程或工程组相关的文件列表或工程组中的引用。

当创建一个应用程序时，通常要创建一些新的窗体。利用工程管理器，可以调用以前所创建的窗体进行修改或直接引用，这样可以缩短应用程序的开发时间。

当工程所有的部件被汇集在一起并完成代码编写后，便可以编译工程，创建一个可执行文件。

4.3 数据类型

数据是程序的必要组成部分，也是程序处理的对象。在高级语言中，广泛使用“数据类型”这一个基础的概念，数据类型体现了数据结构的特点。Visual Basic 6.0 提供了系统定义的基本数据类型，并且允许用户根据自己的需要定义自己的数据类型。

4.3.1 系统数据类型

不同类型的数据，所占的存储空间是不一样的，选择使用合适的数据类型，可以优化代码，而且可以防止在计算中溢出。数据类型不同，对其处理的方法也是不同的，这就需要进行数据类型的说明或定义。只有相同（或兼容）的数据类型之间才能进行操作，不然在程序中会出现错误。

4.3.1.1 数值型数据（Numeric）

Visual Basic 6.0 中常用的数值类型的数据有整型数和浮点数。其中整型数又分整数和长整数，浮点数分为单精度浮点数和双精度浮点数。

1. 整型数

整型数是不带小数点和指数符号的数，可以是正整数、负整数或者 0。

- 整数（Integer）：整数是由两个字节的二进制码表示并参加运算。

  整数的范围为-32768～＋32767，例如：

  254、5478、-23、0

  提醒：(32767-（-32768)）＋1＝2^16，这个和计算机什么有关？

- 长整型数（Long）：长整型数也是一个整型数，它表示的范围更大，在计算机中存储时占用 4 个字节。在 Visual Basic 6.0 中，长整型数中的正号可以省略，并且在数值中不能出现逗号（分节符）。

  长整型数的范围是-2147483648～＋2147483647，例如：

  65448、44656、-54457、0

2. 浮点数

浮点数也称实型数或实数，是带有小数点部分的数值。单精度的数可以用定点形式和浮点形式来表示。

单精度数的定点形式是在该范围内含有小数的数，例如：

-2.6、+25.45、0.000012、-6454.45　　单精度浮点数

-12.123456478456、0.987546653、100000.245　　双精度浮点数

浮点数的浮点采用的是科学计数法，它由符号、尾数、指数 3 部分组成。单精度浮点数和双精度浮点数的指数分别用“E”(或“e”）和“D”(或“d”）来表示。例如：

568.721E+4 或 568.721e4　　单精度浮点数，相当于 568.721 乘以 10 的 4 次幂。

568.72189D4 或 568.72189d+4　　双精度浮点数，相当于 568.72189 乘以 10 的 4 次幂。

在上面的例子中，568.721 和 568.72189 是尾数部分，E+4、e4、D4 及 d+4 是指数部分。

- **单精度浮点数（Single）：以 4 个字节存储，其中符号位占 1 位，尾数位占 23 位，指数位占 8 位。可以表示最多 7 位有效数字的数，小数点可以位于这些数字的任何位置，正号可以省略。其负数的取值范围为**-3.402823E+38~ -1.401298E-45，**其正数的取值范围为** 1.401298E-45~3.402823E+38。
- **双精度浮点数（Double）：以 8 个字节存储，其中符号位占 1 位，尾数位占 52 位，指数位占 11 位。可以表示最多 15 位有效数字的数，小数点可以位于这些数字的任何位置，正号可以省略。**

其负数的取值范围为-1.797693134862316D+308~ -4.94065D-324。

其正数的取值范围为 4.94065D-324~1.797693134862316D+308。

4.3.1.2　字符型数据（String）

字符型数据是一个字符排列，由 ASCII **字符组成，包括标准** ASCII **字符和扩展** ASCII **字符。**

Visual Basic 6.0 **中，字符串是放在双引号里面的，其中一个西文字符占一个字节，一个汉字或者全角字符占两个字节。长度为** 0（**不含任何字符**）**的字符串称为空串。**

Visual Basic 6.0 **中包括两种类型的字符串：变长字符串和定长字符串。**

1. **变长字符串**

变长字符串是指字符串的长度是不固定的，随着对字符串变量赋予新的字符串，它的长度是可以改变的，可以变大也可以变小。缺省情况下，如果一个字符串没有定义成固定长的，那么它就是属于可变长字符串。例如：

“Hello，World”、“2+3＝”、“**型号**”、“800-142-546-987”

2. **定长字符串**

定长字符串是指在程序的执行过程当中，保持字符长度不变的字符串。例如，声明了长度的字符串，假设为 8 **位，这样的情况下，如果字符数没有** 8 **个，余下将被空格填满，如果超过** 8 **个，超过的部分将被舍弃。**

其长度用类型名加上一个星号和常数指明，一般格式为：

```
String*常数
```

这里的“常数”是字符个数，它指定定长字符串的长度。

4.3.1.3　布尔型数据（Boolean）

布尔类型数据是一个逻辑值，用两个字节（byte**）存储，它只有两个值：“**True**”or “**False**”，就是真和假。**

数值类型数据向布尔类型数据转换的时候，0 **为“**False**”，非** 0 **值为“**True**”。**

布尔类型转换到数值类型的时候，“True**”成**-1**，“**False**”为** 0。

4.3.1.4　变体型数据（Variant）

变体型数据是一种可变的数据类型，可以存放任何类型的数据，因此，变体型可以说是 Visual Basic 6.0 **中用途最广、最灵活的一种变量类型。**

程序中没有说明的时候，Visual Basic 6.0 **会自动将该变量默认为** Variant **型变量，例如：**

```
a=“6”
a＝6-2
a＝“D”&a
```

以上介绍了 Visual Basic 6.0 **中的基本数据类型。表** 4-1 **列出了这些数据类型的名称、存储**

空间、和取值范围。

表 4-1　　Visual Basic 6.0 基本数据类型

数据类型	存储空间	取值范围
Integer（整型）	2 字节	-32768 ~ 32767
Long（长整型）	4 字节	-2 147483648 ~ 2 147483647
Single（单精度）	4 字节	负数的取值范围-3.402823E+38~ -1.401298E-45 正数的取值范围 1.401298E-45~3.402823E+38
Double（双精度）	8 字节	负数的取值范围-1.79769313486232D+308~-4.9406564584127D-324 正数的取值范围 4.940654584127D-324~1.79769313486232D+308
Boolean（布尔型）	2 字节	True 或 False
Byte（字节型）	1 字节	CHR（0）~CHR（255）
String（变长字符串）	10 字节加字符串长度	0 到大约 21 亿
String（定长字符串）	字符串长度	0~65535
Variant（数字）	16 字节	任何数字值，最大可达到 Double 的范围
Variant（字符）	22 字节加字符串长度	与变长 String 有相同的范围

4.3.2 用户自定义数据

使用 Visual Basic 6.0 提供的数据类型基本上已经可以满足用户的要求，但有时会需要存放一组不同类型的数据。例如，一个管理学生的教务系统，一个学生通常要有许多特征，如学生的姓名、年龄、性别等。如果每一个特征都用一个变量表示，当有许多学生时很可能产生混乱。这时，就可以把学生的所有特征构造为一个数据类型。

在 Visual Basic 6.0 中构造数据类型可以用 Type 语句定义，其格式如下：

```
Type  数据类型名
    数据类型名元素名   As    类型名
    数据类型名元素名   As    类型名
End  Type
```

其中“数据类型名”是要定义的数据类型的名字，其命名规则相同（见下一节）；“数据类型名元素名”也遵守同样的规则，且不能是数组名；“类型名”可以是任何基本数据类型，也可以是用户定义的类型。例如：

```
Type Student
    Name  As  String
    No  As  Integer
    Age  As  Integer
    Sex  As  String*1
End Type
```

这里的“Student”是一个用户定义的类型，它由 4 个元素组成：“Name”、“No”、“Age”、“Sex”。其中“Name”是变长字符串；“No”和“Age”是整型；“Sex”是由 1 个字符组成的定长字符串。

4.4 Visual Basic 6.0 的变量和常量

变量和常量是 Visual Basic 6.0 乃至所有编程语言中非常重要的概念。变量（Variable）和常量（Constant）是用来存储数据的，它们记载了程序运行过程中需要用到的一些数据。但是，变量和常量是有区别的，它们的区别是：变量的值是根据程序运行的需要而可以随时改变的，而常量的值在程序运行过程中是不可改变的。

4.4.1 变量

变量是指在程序的运行过程中随时可以发生变化的量。可以把它想象成一种容器，比如一只篮子，它可以装一些苹果，也可以装满一篮鲜花或者只装一枝鲜花，甚至可以什么都不装。变量也是如此，它可以存放一个数字，也可以存放一个字符串，甚至可以什么值都没有。

当在窗体中设计用户界面时，Visual Basic 6.0 会自动为产生的对象（包括窗体本身）创建一组变量，即属性变量，并为每个变量设置其缺省值。这类变量可供直接使用，比如引用它或给它赋新值。用户也可以创建自己的变量，以便存放程序执行过程中的临时数据或结果数据等。在程序中，这样的变量是非常需要的。

4.4.1.1 变量命名规则和注意事项

变量是任何一门高级语言所必须具有的过程中传递的参数。变量的值可以改变，变量有一个名字和一定的数据类型，在内存中占有一定的存储单元。在该存储单元中存放变量值，注意变量名和变量的值是不同的两个概念。

在 Visual Basic 6.0 中变量的命名是有一定规则的，这些规则指出了用户变量和其他语言要素之间的区别。这些规则如下。

- 一个变量名的长度不能超过 255 个字符。
- 变量名的第一个字符必须是字母 A～Z，第一个字母可以是大写，也可以是小写
- 其余的字符可以由字母，数字和下划线组成。
- Visual Basic 6.0 中的保留字不能用作变量名，保留字包括 Visual Basic 6.0 的属性、事件、方法、过程、函数等系统内部的标识符。

根据上面的规则，class，my_var，sum 是合法的变量名，而 Elton．D．John，#9，8abc 等是不合法的变量命，如果用户定义并且使用了这些非法变量，那么在程序编译时就会出错。

在 Visual Basic 6.0 中，变量名是不区分大小写的，也就是说如果有两个变量：abc 和 ABC，那么这两个变量是相同的。比如，如果有下面几条语句，系统会认为它们是相同的：

abc＝1；

ABC＝1；

AbC＝1；

定义和使用变量时，通常要把变量名定义为容易使用和能够描述所含数据用处的名称，建议不使用一些很难理解的缩写：A、C2 等。譬如说编写学生管理程序的时候，定义 student_No 代表学号，student_Score 代表成绩，易于理解和改正错误。Visual Basic 6.0 是 32 位的开发工具，因此变量名长度可以支持到 255 个字符。这对于用户编程是非常重要的，因为在开发大型的系统时变量会非常多，如果变量名长度不够长就很可能出现重名的情况。

4.4.1.2 声明变量

在使用变量之前，很多语言首先需要声明变量。也就是说，必须事先告诉编译器在程序中

使用哪些变量、变量的数据类型以及变量的长度。这是因为在编译程序执行代码之前编译器需要知道如何给语句变量开辟存储区，这样可以优化程序的执行。

Visual Basic 6.0 的显著特点就是不需要在使用一个变量之前一定要定义该变量，这样虽然程序运行的速度和效率上降低了，但是给程序员带来了一定的方便。不过在严格的编程思想学习方面，建议不采用这样的方法。

声明语句有如下 4 种格式（其使用范围将在下几节讲到）。

- Dim 语句，其格式为：

  `Dim<变量名>[As<数据类型>]`

- Private 语句，其格式为：

  `Private<变量名>[As<数据类型>]`

- Public 语句，其格式为：

  `Public<变量名>[As<数据类型>]`

- static 语句，其格式为：

  `Static<变量名>[As<数据类型>]`

用 Dim 语句在窗体的过程中声明的变量称为局部变量。下列代码段中的 Dim 使用，读者可以先不要担心这些代码段的复杂或者不解，这里需要掌握的就是在代码中 Dim 的使用和注意事项，以及 Dim 在里面的作用。代码如下：

```
Private Sub Command_Click()
    Dim h As Integer,m As Integer,s As Integer
    Dim x As long
    h=Val(Text1.Text)
    m=Val(Text2.Text)
    s=Val(Text3.Text)
    x=h+m+x
    Text4.text=str$(x)
End Sub
```

这段代码完成了一个“加法计算”功能的命令按钮，其他都无须去考虑，Dim 语句在里面起到了声明变量的作用。Dim 语句声明的变量只能在这个 Sub 程序中使用，也就是说 Dim 语句声明的变量为局部变量，只能在声明的程序段中才能使用。

在 Visual Basic 6.0 中可以显式或者隐式地声明变量。显式地声明变量就是明确定义变量的类型、变量名和长度，让编译器事先编译。显式声明使用 Dim 语句，（其他格式的声明使用方式类似）格式如下：

```
Dim  Var[As type]
```

例如要声明一个名为“student_name”的字符串型变量就可以用下面的代码：

```
Dim  student_name  As  String
```

在这个例子中，“As”用来声明变量的数据类型或对象类型．这里“String”表示是字符串数据类型，其他数据类型还有“Integer”等，在后面将详细介绍各种数据类型。

用一个 Dim 可以定义多个变量，例如：

```
Dim Var1 As Sting, Var2 As Integer
```

将 Var1 和 Var2 分别定义为字符串和整型变量。

当在一个 Dim 语句中定义多个变量时，每个变量都要使用 As 字句声明其类型，否则改变量被看作是变体类型。例如，上面的例子改为：Dim Var1, Var2 As Integer
则 Var1 将被定义为变体型变量，Var2**、被定义为整型变量。**

缺省情况下，每种数据类型都有一定的缺省长度，如果需要定义固定长度的变量，可以根据下面的例子：

```
Dim  student_name As String*20
```

其中“20**”代表每个变量字符串的长度，定义了变量之后，当编译器发现** Dim **语句时，就会根据语句中定义生成新的变量，即在内存中保留一定空间并为其取名，当后面用到这个变量时就会利用这个内存区来读取或者设置变量的值。**

例如，使用下面的语句时：

```
student_name=“张三”
```

这时，系统就会把“张三”这个字符串存储到系统开辟的变量“student_name**”的内存空间中，当程序中引用这个变量时就会从内存中读取这个变量的值。如：**Print student_name**。**

如果只是 Dim　A**，这样的情况下，没有指定变量的类型，那么变量** A **为变体型。**

4.4.1.3　变量的作用范围

在 Visual Basic 6.0 **中声明变量时，说明部分的放置位置决定了变量只能在程序中的某一部分有效，变量对于程序的可识别程度称为变量的作用范围。**

在前面已经讲述过，Visual Basic 6.0 **应用程序由** 3 **种模块组成，即窗体模块、标准模块、和类模块。如图** 4-2 **所示。其中窗体模块包括事件过程、通用过程和声明过程；而标准模块由通用过程和声明部分组成。**

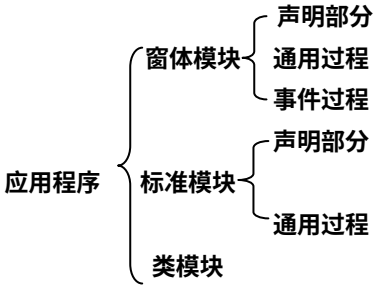


图4-2　Visual Basic 6.0 **应用程序的结构**

根据变量的定义位置和所使用的变量定义语句的不同，Visual Basic 6.0 **中的变量可以分为** 3 **类：局部变量、模块变量和全局变量，其中模块变量包括窗体模块变量和标准模块变量。**

- **全局变量：在标准模块的声明部份，用** Public **声明的变量就是全局变量，程序中的任何窗体和模块都能访问到它。**
- **局部变量：在过程和函数中用** Dim **或** Static **等声明的变量只在定义它的过程和函数中有效。**
- **模块或窗体变量：在模块和窗体中用** Dim **或** Private **等声明的变量只在本模块或窗体中起作用，这样的变量叫模块或窗体变量。**

4.4.1.4　静态变量

前面介绍过可以用 Dim **语句来声明过程级局部变量，这种局部变量在每次过程调用结束时消失，但是有时用户会希望过程中的某个变量的值一直存在，这就要用静态变量。静态变量用** Static **声明，例如：**

```
Static  I  As  Integer
```

声明了静态变量之后，每次过程调用结束时系统就会保存该变量的变量值。在下一次调用该过程时，该变量的值会一直存在。例如，在窗体设计器中加入一个命令按钮，在按钮的单击事件中加入下面的代码，使用 Static 来声明变量“n”，每次调用该过程时就会形成一个计数的功能。

```
Private  Sub  Cmd1_Click
    Static  n  As  Integer
    n=n+1
End Sub
```

运行该程序后，用户每单击一下命令按钮，n 的数值加 1。

4.4.1.5 同名变量

若在不同模块中的公用变量使用同一名字，则通过同时引用模块名和变量名就可以在代码中区分它们。例如，如果有一个在窗体模块（Form1）和通用模块（Module1）中都声明了的公用 Integer 类型变量“i”，那么引用时只需要用“Form1.i”和“Module1.i”，引用它们就可以得到正确的结果。如果，在标准模块中的变量没有同名的变量，就可以省略前面的模块名，而只引用“i”。

不同的作用范围内也可以有同名的变量。例如，有名为“T”的公用变量，然后在过程中声明名为“T”的局部变量。这时在引用变量时可以按照下面的规则：

在过程内通过引用名字“T”来访问局部变量，通过模块名加上点操作符和“T”变量名可以访问公用变量，如“Form1.T”。而在过程外则通过引用名字“T”来访问公用变量，而不能访问局部变量。

一般说来，当变量名称相同而范围不同时，局部变量优先被访问，模块变量可以通过引用关系进行访问。

虽然同名的变量处理并不十分复杂，但是这样很可能会带来麻烦，而且可能会导致难以查找的错误。因此，对不同的变量使用不同的名称才是一种好的编程习惯。

4.4.1.6 变量的数据类型

在程序中使用的任何变量都有其数据类型，其中包括基本数据类型和用户自定义数据类型。Visual Basic 6.0 中，可以用下面两种方法来规定变量的数据类型。

- 用类型说明符来标识

把类型说明符放在变量名的尾部，可以标识不同的变量类型，这些类型说明符分别为：

%	整型
&	长整型
!	单精度符点数
#	双精度符点数
$	字符串型

例如：I%、No&、Name$、Score!

- 在定义变量时指定其类型

在定义变量时并指定其类型，可以用下面的格式：

```
Declare  变量名  As  类型名
```

其中“Declare”可以是 Dim、Static、Public 或 Private，“As”是关键字，“类型名”可以

是数据的基本类型或用户定义的类型。

4.4.2　常量

常量是在整个程序中事先设置的、其值不会改变的数据。一般对于程序中使用的常数，能够用常量表示的尽量用常量表示。这样可以用有意义的符号来标识数据，增强程序的可读性；而且如果要一次性改动程序中存在的全部某个常数时，不需要在程序中通过查找来进行修改，可以只改变与这个常数对应的常量的值即可，增强了程序的可维护性。常量可分为直接常量和符号常量。

直接常量以直接的方式给出数据。如：123、“abc”、ture 等。

对于符号常量的定义用 Const 定义，其定义格式如下：

```
[Public] Const 常量名[As 类型名]=表达式
```

其中，说明类型是可选的，当省略说明常量类型时，常量的类型由它的值决定。等号后面的表达式必须用常量表达式，不能包含变量。例如：

```
Const PI=3.1416
```

上面这行语句就定义了一个代表圆周率的常量，它的类型是浮点型的，在以后用到圆周率时，就可以用常量“PI”来代替。例如要计算圆的面积，可用如下的代码：

```
S=PI*r^2
```

其中“S”和“r”分别是存放面积和半径的变量。程序执行到此处时，自动将常量“PI”换成 3.1416，因此，对常量的处理要比变量快一些。使用常量还有一个好处，就是当以后要改变“PI”的精度时，如把它改成 3.1415926，只需要在定义 PI 处改变一下就行了；而如果直接使用数值 3.1415 的话，该数出现了多少次，就要改动多少次，这样不仅麻烦还极易出错。

同变量一样，常量的作用域也可以分为 3 种：局部常量、模块级窗体常量以及全局常量。局部常量必须在过程或函数内部定义，模块级常量是在某个模块的“声明”部分定义的，它们都是使用的“Const”关键字，只是定义地方不一样而已。全局常量则是在标准模块的“声明”部分定义的，而且需要在“Const”前面加上“Public”关键字。

定义常量时，在表达式中还可以包含已经定义过的常量，现举例如下：

```
Const PI=3.1416
Const R=2
Const S=PI*R^2
```

在此例中，定义常量“S”时，其中包含了已经定义过的常量“PI”和“R”。

需要注意的是在定义字符型常量时，后面的字符串必须加上双引号，否则的话 Visual Basic 6.0 会把这个字符串认作是已经定义过的常量，从而导致错误。如“定义过程级和模块级常量”中的常量文本，如果把“PI”的双引号去掉，Visual Basic 6.0 会认为 PI 是已经定义常量而不是字符串。另外，常量的值在定义之后，就再也不能在程序中进行改变，如果试图以常量赋值将会发生错误。

常量的其他一些性质类似于变量，例如：符号常量的命名规则、常量的数据类型等。

4.5　运算符和表达式

运算（即操作）是对数据的加工。最基本的运算形式常常可以用一些简洁的符号来描述，这些符号称为运算符或操作符，被运算的对象，即数据，称为运算量或操作数。由运算符和运

算量组成的表达式描述了对哪些数据以何种顺序进行什么样的操作。运算量可以是常量，也可以是变量，还可以是函数。例如：2+3、a+b、Sin(x)、a=2、PI*r*r 等都是表达式，单个变量和常量也可以看成是表达式。

Visual Basic 6.0 提供了丰富的运算符，它包括算术运算符、关系运算符、逻辑运算符以及字符串连接运算符。它们可以构成多种表达式。

4.5.1 算术运算符

算术运算符是最为常用的运算符，可以进行简单的算术运算。在 Visual Basic 6.0 中提供了 8 种算术运算符，表 4-2 按优先级列出了这些算术运算符。

表 4-2　　Visual Basic 6.0 算术运算符

运算符名称	运算符	表达式例子
幂	^	A^2
取负	-	-A
乘法	*	A*B
浮点除法	/	A/B
整数除法	\	A\B
取余	Mod	A Mod B
加法	+	A+B
减法	-	A-B

在这 8 种算术运算符中，除负（-）是单目运算符（只有一个运算变量）外，其他均为双目运算符（需要两个运算变量）。

加（+）、减（-）、乘（*）、除（／）以及取负（-）几个运算符的含义和用法与数学中基本相同，下面介绍其他几种运算符的含义和用法。

1. 幂运算

幂运算（^）与数学运算中的指数运算类似，用来进行乘方和方根运算。例如：2^8 表示 2 的 8 次方，即为数学运算中的 2^8。下面是幂运算的几个例子：

10^3　　10 的立方，即 10^3=1000

81^0.5　　81 的平方根，即 $81^{1/2}$=9

10^-1　　10 的倒数，即 1÷10=0.1

2. 整数除法

整数除法运算符（\）进行整除运算，结果为整型值，因此表达式“5\3”的结果为 1。整除的操作数一般为整型数，其取值必须在-217483648.5~2147483647.5 范围内。当其操作数为浮点型时，首先进行四舍五入为整型或长整型，然后进行整除运算。其运算结果被截断为整型数（Integer）或长整型数（Long），不进行舍入处理。例如：

```
a=5\3
b=21.81\3.4
```

运算结果为：

a=1，b=7。

3. 取余运算

取余运算符（Mod）用来求余数，其结果为第一个操作数整除第二个操作数所得的余数。例如：5 Mod 3 的结果为 2，即 5 整除 3，则其余数为 2。

同整数的除法运算一样，取余运算符的操作数一般也为整型数，它的取值范围必须在 -217483648.5~2147483647.5 之间。当其操作数为浮点型时，首先进行四舍五入为整型或长整型，然后进行取余运算，例如：21.81 Mod 3.4 的结果为 1。

4.5.2　字符连接符

字符串表达式是采用连接符将两个字符串常量、字符串变量、字符串函数连接起来的式子。连接符有两个：“ & ”和“ + ”。

其作用都是将两个字符串连接起来，运算结果是一个字符串。例如：

“ 计算机 ” & “ 网络 ”　　结果是“ 计算机网络 ”

“ 123 ” + “ 45 ”　　结果是“ 12345 ”

“ 123 ” & “ ABC ”　　结果是“ 123ABC ”

4.5.3　关系运算符

关系运算符是用来对几个表达式的值进行比较运算的，也称比较运算符。其比较的结果是一个逻辑值，即真（True）或假（False）。Visual Basic 6.0 中提供了 8 种关系运算符，表 4-3 按优先级列出了这些关系运算符。

表 4-3　　Visual Basic 6.0 关系运算符

运算符名称	运算符	表达式例子
=	相等	A=B
<>或><	不相等	A<>B 或 A><B
<	小于	A<B
>	大于	A>B
<=	小于或等于	A<=B
>=	大于或等于	A>=B
Like	比较样式	
Is	比较对象变量	

用关系运算符连接的两个操作数或算术运算表达式组成的式子叫关系表达式。关系表达式的结果是一个逻辑值，即真（True）或假（False）。例如：

3>2　　结果为“ True ”即 -1

（A+B）<T/2（其中 A=1，B=2，T=4）　　结果为“ False ”即 0

关系运算符既可以进行数值的比较，也可以进行字符串的比较。当进行字符串比较时，首先比较两个字符串的第一个字符，其中 ASCII 码值较大的字符所在的字符串大。如果第一个字符相同，则比较第二个，依此类推。例如：

“ abcdhijlsa ” > “ aeabdf ”　　其结果为“ False ”即 0

在数学中判断 x 是否在区间[a，b]时习惯上写成 a≤x≤b，但在 Visual Basic 6.0 中不能写成 a<=x<=b，应写成 a<=x And x<=b。“ And ”是下面将要介绍的逻辑运算符“ 与 ”。

4.5.4 逻辑运算符

逻辑运算符是用来连接两个或多个关系式，组成一个布尔表达式，也称布尔运算符。在 Visual Basic 6.0 中有 6 种逻辑运算符。表 4-4 按优先级列出了这些逻辑运算符。

表 4-4　　逻辑运算符

运算符名称	运算符	表达式例子
非	Not	Not(A>B)
与	And	(A<B)And(2>3)
或	Or	(A<B)Or(2>3)
异或	Xor	(A<B)Xor(2>3)
等价	Eqv	(A<B)Eqr(2>3)
蕴含	Imp	(A<B)Imp(2>3)

在 6 种逻辑运算符中，除了非（Not）是单目运算符外，其他均为双目运算符。

1. 非运算符（Not）

非运算符（Not）进行“取反”运算，即使真变假或是假变真。例如：

4>5　　结果为“False”即 0

Not(4>5)　　结果为“True”即-1

2. 与运算符（And）

与运算符（And）是对两个关系表达式的值进行比较运算，如果表达式的值均为“True”，结果才为“Ture”；否则为“False”。例如

(4>5)And(6>3)　　其结果为“False”即 0

(4>5)And(6>8)　　其结果为“False”即 0

(8>5)And(6>3)　　其结果为“True”即-1

3. 或运算符（Or）

或运算符（Or）进行两个表达式的比较运算，如果其中一个表达式的值为“True”，结果才为“True”；只有两个表达式的值都为“False”时，结果才为“False”。例如：

(4>5)Or(6>3)　　其结果为“True”即-1

(4>5)Or(6>8)　　其结果为“False”即 0

(8>5)Or(6>3)　　其结果为“True”即-1

4. 异或运算符（Xor）

用异或运算符（Xor）运算时，当两个表达式的值同时为“True”或同时为“False”时，结果才为“False”。否则为“True”。例如：

(4>5)Xor(6>3)　　其结果为“True”即-1

(4>5)Xor(6>8)　　其结果为“False”即 0

(8>5)Xor(6>3)　　其结果为“False”即 0

5. 等价运算符（Eqr）

当两个表达式的值同时为“True”或同时为“False”时，结果才为“True”。否则为“False”。例如：

(4>5)Eqr(6>3)　　其结果为“False”即 0

(4>5)Eqr(6>8)　　其结果为“ True ”即-1

(8>5)Eqr(6>3)　　其结果为“ True ”即-1

6. 蕴含表达式（Imp）

只有当第一个表达式的值为“ True ”，且第二个表达式的值为“ False ”时，其结果为“ False ”，否则为“ True ”。例如：

(4>5)Imp(6>3)　　其结果为“ True ”即-1

(4>5)Imp(6>8)　　其结果为“ True ”即-1

(8>5)Imp(6>3)　　其结果为“ True ”即-1

(6>3)Imp(4>5)　　其结果为“ False ”即 0

表 4-5 列出了 6 种逻辑运算符的运算值。

表 4-5　　逻辑运算的运算值

A	B	Not A	A And B	A Or B	A Xor B	A Eqr B	A Imp B
-1	-1	0	-1	-1	0	-1	-1
-1	0	0	0	-1	-1	0	0
0	-1	-1	0	-1	-1	0	-1
0	0	-1	0	0	0	-1	-1

4.5.5 常用内部函数

Visual Basic 6.0 提供了大量的内部函数，使用函数可以带来很大的方便。使用函数有如下两种方法。

- 如果需要使用返回值，其格式为：

  变量名=函数名(参数列表)

- 如果不需要使用返回值，其格式为：

  函数名 参数列表

所谓参数，就是在调用函数时交给函数处理的数据。所谓返回值，就是函数经过一系列运算后返回给调用者的值。表 4-6 列出了 Visual Basic 6.0 中部分常用的函数。

表 4-6　　Visual Basic 6.0 中部分常用函数

类别	函数名	作用
类型转换函数	Cint(x)	将 x 的值的小数部分四舍五入转换为整型的
	CLng(x)	将 x 的值的小数部分四舍五入转换为长整型的
	CSng(x)	将 x 的值转换为单精度浮点型的
	CDbl(x)	将 x 的值转换为双精度浮点型的
	CStr(x)	将 x 的值转换为字符型的
	CBool(x)	将 x 的值转换为布尔型的
	Cvar(x)	将 x 的值转换为变体型的
	Val	将代表数值的字符串转换成数值型数据
	Str$	将数值型数据转换成代表数值的字符串

续表

类别	函数名	作用
数学函数	Abs(x)	返回 x 的值绝对值
	Sqr(x)	返回 x 的值平方根
	Fix(x)	若 x 是正数，则返回该数的整数总数；若参数是负数，则返回一个不小于参数的最小整数
	Int(x)	若 x 的值是正数，则返回该数的整数部份；若参数是负数，则返回一个不大于参数的最大整数
	sgn(x)	返回 x 的符号，x 为正数时返回 1，为 0 时返回 0，若 x 为负数则返回-1
	Exp(x)	返回以 e 为底的 x 的指数的值
	Log(x)	返回以 e 为底的 x 的对数的值
	Sin(x)	返回 x 的正弦值
	Cos(x)	返回 x 的余弦值
	Tan(x)	返回 x 的正切值
	Atn(x)	返回 x 的余切值
	Rnd	返回一个 0～1 之间的一个单精度随机数

在 Visual Basic 6.0 中除了常用的一些字符转换函数和数学函数外，还提供了十分丰富的字符串处理函数。字符串函数是用来对字符串进行处理或操作的函数，主要有如下这些。

- Len：用来返回字符串的长度（即字符串中字符的个数）。例如：Len("Hello")、Len ("Good")的值分别为 5 和 4。
- Left：从某字符串的左边截取子字符串。其使用格式为：Left(原字符串，截取长度)。该函数有两个参数，第一个是被截取的原字符串，第二个是截取的字符个数。例如：Left("Hello",2)是从字符串“ Hello ”左边截取 2 个字符，返回值是“ He ”。
- Right：从字符串的右边截取子字符串，使用方法与 Left 一样。例如：Right("Hello",2)的值为“ lo ”。
- Mid：是从中间截取子字符串，是 Left 函数和 Right 函数的综合。该函数的使用格式为 Mid(字符串，启始位置，截取个数)。例如：Mid("Hello",3,2)，表示从该字符串的第三个字符处截取 2 个字符，其值为“ ll ”。
- StrReverse：返回与原字符串反向的字符串。例如：StrReverse("Hello")的值为“ olleH ”。
- LTrim：清除字符串左边的空格。例如：LTrim("Hello")的值为“ Hello ”。
- RTrim：清除字符串右边的空格。例如：LTrim("Hello")的值为“ Hello ”。
- Trim：清除字符串两边的空格。例如：Trim("Hello")的值为“ Hello ”。
- Space：返回一个由指定空格组成的字符串。注意该返回值与空字符串(“ ”)并不相同，前者是由空格组成的字符串，而后者中不包含任何内容。例如：Space("5")的值为“ ”。
- String：返回一个由指定字符组成的字符串。例如：String(5，"#")的值为“ ##### ”。
- LCase：将字符串的所有字母变成小写。例如：LCase("Hello")的值为“ hello ”。

- UCase：将字符串的所有字母变成大写。例如：UCase("Hello") 的值为“HELLO”。

另外 Visual Basic 6.0 还提供了输入输出，本书后面将详细介绍函数、日期函数、时间函数等大量的内部函数。

4.5.6 表达式的执行顺序

当一个表达式中出现多个运算符时，计算机会按照一个规定的顺序对表达式求值。一般其运算顺序为：

1. 首先进行括号内的运算。
2. 其次进行函数的运算。
3. 接着进行算术运算，其算术运算的内部运算顺序为幂（^）→取负（-）→乘（*）、浮点除法（/）→整数除法（\）→取余（Mod）→加（+）、减（-）。当乘法和浮点除法或加与减，同时出现在表达式中时，将按照它们从左到右出现的顺序进行计算。用括号可以改变它们的优先级。
4. 然后进行字符串连接运算（&或+）。
5. 再进行关系运算（=、>、<、<>或><、<=、>=）。
6. 最后进行逻辑运算，其内部顺序为非（Not）→与（And）→或（Or）→异或（Xor）→等价（Eqr）→蕴含（Imp）。各种运算符的运算执行顺序见表 4-7 所示。

与数学表达式相比，Visual Basic 6.0 中的表达式与其有类似的地方，但也有区别，因此在书写表达式时应注意以下几点。

- 在一般情况下，不允许两个运算符相连，应当用括号隔开。
- 括号可以改变运算顺序。在表达式中只能是用圆括号“()”，不能使用方括号“[ ]”或花括号“{ }”。
- 乘号（*）不能省略，也不能用“ ”代替。

表 4-7　　运算符执行顺序

算术	连接	比较	逻辑	优先级
幂（^）	字符串连接运算（&或+）	相等（=）	非（Not）	高
取负（-）		不等于（<>或><）	与（And）	
乘（*）、浮点除法（/）		小于（<）	或（Or）	
整数除法（\）		大于（>）	异或（Xor）	
取余（Mod）		小于等于（<=）	等价（Eqr）	
加（+）、减（-）		大于等于（>=）	蕴含（Imp）	
		Like		低
		Is		

4.6 数组

在实际应用中，通常需要处理成批的同一类型的数据。例如，为了处理 50 个学生的考试总成绩，可以使用 $S_1,S_2,S_3,S_4.....S_{50}$ 来代表每个学生的分数，其中 S_1 代表第一个学生的分

数，S_2 代表第二个学生的分数……这里的 S_1,S_2,S_3,S_4…….S_{50} 是带有下标的变量，通常称为下标变量。显然，用一批具有相同名字、不同下标的变量来表示同一属性的一组数据，能更清楚地表示它们之间的关系。在 Visual Basic6.0 中，把一组具有同一名字、不同下标的下标变量称为数组，其一般形式如下：

S(i)

其中 S 成为数组名，i 是下标。一个数组可以含有若干个下标变量，下标用来指出某个数组元素在数组中的位置，S(8)代表 S 数组中的第 8 个元素。在 Visual Basic 6.0 中，使用下标变量时，必须把下标变量放在一对紧跟在数组名之后的圆括号中，即必须把下标变量写成 S(8)，不能写成 S8 或 S_8，也不能写成 S[8]或 S{8}。

数组是任何高级语言都具有的一种数据结构，数组的基本功能是存储一系列变量，并且可以用相同名字引用这些变量，引用时用数字下标来识别它们。当使用多个类型和功能一致的数据时，使用数组可以缩短和简化程序。

例如用户要记录一个班级的同学的名字，可以为每一个同学定义一个变量用来记录他们的名字，例如 student1、student2，依次类推，但是这样做显然是一种非常笨的方法，因为这些变量的类型都是相同的，并且功能相似，都是用来记录学生名字的。因此，采用数组的方法就会简单得多，利用数组只需要定义一个数组变量 student，然后利用数组的下标就可以识别数组中的每一个元素。如图 4-3 所示，显示了用一维数组存储学生姓名的方法。

张三	李四	王五		
Student1	Student2	Student3		

图4-3 一维数组存储

数组可以声明为任何基本数据类型的数组，包括用户自定义类型，但是一个数组中的所有元素一般具有相同的数据类型。当然，当数据类型为 Variant 时，各个元素能够包含不同种类的数据（对象、字符串、数值等）。

Visual Basic 6.0 中，数组按照长度可以分成两种类型的数组：一种是固定大小的数组，这种数组总是保持同样的大小，另一种是在运行时大小可以改变的动态数组。

4.6.1 声明固定大小的数组

有 3 种方法声明固定大小的数组，用哪一种方法取决于数组应有的有效范围。

- 建立全局数组，在模块的声明段用 Public 语句声明数组。
- 建立模块或窗体数组，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 Private 语句声明数组。

声明数组时，在数组名之后跟一个用括弧括起来的，其上界、下界不得超过 Long 数据类型的范围（-2 147 483 648~2 147 483 647）。

例如，下列数组声明可出现在模块的声明段：

```
Dim A(4) As Integer '同时定义了A(0)、A(1)、A(2)、A(3)、A(4)5个元素。
Dim S (20) As Double '定义了S(0)到S(20)21个元素。
```

为建立公用数组，直接用 Public 取代 Dim：

```
Public A(4) As Integer
Public S (20) As Double
```

过程之中同样的声明使用 Dim：

```
Dim A(4) As Integer
Dim S (20) As Double
```

第一个声明建立了一个有 5 个元素的数组，其索引号从 0~4。第二个声明建立了一个有 21 个元素的数组，其索引号从 0~20。缺省的下界为 0。

为了规定下界，可以使用关键字“To”显式提供下界（为 Long 数据类型）：

```
Dim A (1 To 5) As Integer
Dim S (100 To 120) As String
```

在前述声明中，“A”的索引值范围从 1~5，而“S”的索引值范围从 100~120。

有时候，可能需要知道数组的上界值和下界值，这可以通过 Lbound 和 Ubound 函数来测试，其格式如下：

```
Lbound(数组名) '返回数组的上界值
Ubound(数组名) '返回数组的下界值
```

例如：

```
Dim A(5) As Integer
Print Lbound(A)；
Print Ubound(A)
```

其结果为：

```
0  5
```

例如：

```
Dim B(2 to 15) As String
Print Lbound(B);
Print Ubound(B)
```

其结果为：

```
2  15
```

4.6.2 动态数组

数组到底应该有多大才合适，有时可能不得而知。所以希望数组能够在运行时具有改变大小的能力。动态数组就可以在任何时候改变大小。在 Visual Basic 6.0 中，动态数组是最灵活、最方便的，有助于有效管理内存。例如，可短时间使用一个大数组，然后，在不使用这个数组时，将内存空间释放给系统。如果不用动态数组，就要声明一个数组，它的大小尽可能达到最大，然后再抹去那些不必要的元素。但是，如果过度使用这种方法，会导致内存的操作环境变慢。要创建动态数组，请按照以下步骤执行。

1. 如果希望数组为公用数组，则用 Public 语句声明数组；如果希望数组为模块级，则在模块级用 Dim 语句声明数组；如果希望数组为局部数组，则在过程中用 Static 或 Dim 语句声明数组。给数组赋一个空数组，这样就将数组声明为动态数组。例如：

   ```
   Dim A ()
   ```

2. 用 ReDim 语句分配实际的元素个数。

   ```
   ReDim A (X + 1)
   ```

ReDim 语句只能出现在过程中。与 Dim 语句、Static 语句不同，ReDim 语句是一个可执

行语句，由于这一语句，应用程序在运行时执行一个操作。ReDim 语句支持这样的语法，它与固定数组中使用的语法相同。每个 ReDim 语句都能改变元素数目以及上下界。

例如，用第一次在模块级声明所建立的动态数组 M：

```
Dim M () As Integer
```

然后，在过程中给数组分配空间：

```
Sub A ()
...
ReDim M (20)
...
End Sub
```

这里的 ReDim 语句给 M 分配 20 个整数空间。

4.6.3 控件数组

控件数组是由一组相同类型的控件组成。它们共用一个控件名（Name），绝大部分的属性也相同，但有一个属性不同，即“ Index ”属性的值不同。当建立控件数组时，系统给每个元素赋一个惟一的索引号（ndex），通过【属性】窗口的“ Index ”属性，可以知道该控件的下标是多少，第 1 个元素下标是 0。例如，控件数组 Shape(6)表示控件数组名为 Shape 的第 7 个元素。

控件数组最大的特点是：控件数组共享同样的事件过程。所以适用于若干个控件执行的操作相似的场合，例如，控件数组 Command 有 10 个命令按钮，则不管单击哪个命令按钮，就会调用同一个单击事件过程。为了区分是控件数组中的哪个元素触发了事件，在程序运行时，通过传送给过程的索引值（即下标值）来确定。

一个控件数组至少包含一个元素，最多可达 32768 个。

控件数组是针对控件建立的，因此与普通数组的定义不一样。控件数组的建立有两种方式。

1. 第一种方式，在设计时建立，有两种方法。

（1）第一种方法，通过复制现存控件来添加控件数组元素，建立的步骤如下：

① 在窗体上画出某控件，可进行控件名的属性设置，这是建立的第一个元素。

② 选中该控件，进行【编辑】/【复制】操作。

③ 进行【编辑】/【粘贴】操作。Visual Basic 6.0 系统会出现一提示框询问是否建立控件数组，如图 4-4 所示（假设先画了一个“ Command ”命令按钮）。

图4-4 【提示建立控件数组】对话框

④ 单击对话框中的 

按钮，窗体上的左上角将出现一个控件，它就是控件数组的第二个元素。

⑤ 按照第③和第④步的方法，创建控件数组的其他元素。

⑥ 进行事件过程的编程。

（2）第二种方法，通过改变控件名称来添加控件数组元素，建立的步骤如下：

① 绘制控件数组中要添加的全部控件（必须为同一类型的控件）。决定哪一个控件

作为数组中的第一个元素，并设置其“ name ”属性。

② 选定控件并将其“ name ”设置值变成数组第一个元素的“ name ”设置值。

③ 在数组中为控件输入相同名称时，Visual Basic 6.0 系统会出现一提示框询问是否建立控件数组，如图 4-3 所示（假设控件数组名为“ Command ”）。

④ 按照第一种方法的④、⑤、⑥步操作，即可建立控件数组。

2. 第二种方式，在运行时添加控件数组。建立的步骤如下：

（1）先在窗体上画出某控件，设置该控件的“ index ”值为 0，表示该控件为数组；也可进行控件名的属性设置，这是建立的第一个元素。

（2）在编程时通过 Load 方法添加其余的若干个元素，也可以通过 Unload 方法删除某个添加的元素。

（3）每个新添加的控件数组通过设置“ left ”和“ top ”属性，确定其在窗体中的位置，并将“ Visible ”属性设置为“ True ”。

控件数组的使用方法及特征与普通的一些数组基本相似。对于控件数组的其他使用方法和它的一些其他属性特征将在下一章介绍。

4.7 基本流程结构

控制结构可控制程序执行的流程。如果未使用控制流语句，程序便从左至右、自顶向下地贯穿这些语句。有些简单程序可以只用单向流程来编写，有些流程可以依靠运算符的优先级来控制，但任何编程语言的效力和用途皆由其通过结构和循环改变语句顺序的能力而得。

4.7.1 顺序结构

顺序结构就是指程序按照语句的先后顺序一条一条地执行。使用这种结构只需要将合法语句按照合理的执行顺序排列好，即可一一执行。

在 Visual Basic 程序设计中，顺序结构是一类最简单的结构，这种结构的程序是按从上到下的顺序依次执行语句的，中间既没有调转性的语句，也没有循环语句。

在顺序程序设计中用到的典型语句是赋值语句、输入输出语句以及其他计算语句，如加、减、乘、除算术运算等。下面介绍一个例子，以说明顺序结构程序设计的特点。

例如：求二次方程 ax^2+bx+c=0 的根。

```
Private Sub Command1_Click()
    Dim a, b, c As Single   '定义变量
    Dim D As Single
    Dim x1, x2 As Single
    a = 1   '为变量赋初值
    b = 5
    c = 6
    D = b * b - 4 * a * c
    x1 = (-b +Sqr(D)) / (2 * a)'计算方程的根
    x2 = (-b - Sqr(D)) / (2 * a)
    Text1.text=Str$(x1)     '将结果在文本框中显示出来
    Text2.text=Str(x2)
```

```
End Sub
```

4.7.2 条件结构

Visual Basic 6.0 的过程能够测试条件表达式，然后根据测试结果执行不同的操作。

Visual Basic 6.0 支持的条件结构有：

- If...Then
- If...Then...Else
- IIf
- Select Case

1. If...Then

用 If...Then 结构有条件地执行一个或多个语句。单行语法和多行块语法都可以使用：

```
If condition Then statement
If condition Then statements
End If
```

“condition”通常是比较式，但它可以是任何计算数值的表达式。Visual Basic 6.0 将这个值解释为“True”或“False”。一个为零的数值为“False”，而任何非零数值都被看作“True”。若“condition”为“True”，则 Visual Basic 6.0 执行“Then”关键字后面的所有“statements”。

例如：设计一个求数 a 的绝对值命令按钮。

```
Private Sub Command1_Click()
    Dim a As Integer    '定义变量
    a = Val(Text1.Text)     '将文本框中输入的值赋给变量 a
    If a < 0 Then   '当 a 为负数时取其相反数
        a = -a
    End If
    Text2.Text = Str$(a)
End Sub
```

在 Visual Basic 6.0 中可以使用单行或多行语法有条件地执行一个语句（下面两个例子是等价的）：

语句 1：`If a < 0 Then a = -1`

语句 2：
```
If a < 0 Then
        a = -1
       End If
```

If...Then 的单行格式不用 End If 语句。如果“condition”为“True”时要执行多行代码，则必须使用多行块 If...Then...End If 语法。

```
If a < 0 Then a = -1
   Timer1.Enabled = False'定时器控制失效
End If
```

2. If...Then...Else

用 If...Then...Else 块定义几个语句块，执行其中一个语句：

```
If condition1 Then
   [statementblock-1]
[Else If condition2 Then
   [statementblock-2]]
   ...
[Else   [statementblock-n]]
End If
```

Visual Basic 6.0 首先测试“condition1”。如果它为“False”，Visual Basic 6.0 就测试“condition2”，依此类推，直到找到一个为“True”的条件。当它找到一个为“True”的条件时，Visual Basic 6.0 就会执行相应的语句块，然后执行 End If 后面的代码。作为一个选择，可以包含 Else 语句块，如果条件都不是“True”，则 Visual Basic 6.0 执行 Else 语句块。

If...Then...Else If 只是 If...Then...Else 的一个特例。注意，可以使用任意数量的 Else If 子句，或者一个也不用。可以有一个 Else 子句，而不管有没有 Else If 子句。

例如：$y=\begin{cases}1 & (x>0)\\ 0 & (x=0)\\ -1 & (x<0)\end{cases}$

输入 x 的值，输出 y 的值。

```
Private Sub Command1_Click()
    Dim x, y As Single
    x = Val(Text1.Text)
    If x> 0 Then
        y = 1
    ElseIf x = 0 Then
        y = 0
    Else
        y = -1
    End If
    Text2.Text = Str$(y)
End Sub
```

例如：判断某年是否为闰年。

闰年的条件是：

- 能被 4 整除，但不能被 100 整除的年份都是闰年。
- 能被 100 整除，又能被 400 整除的年份是闰年。

```
Dim x As Integer
Private Sub Command1_Click()
    x = Val(Text1.Text)
    If (x Mod 100) Then     '如果 X 不能被 100 整除
        If (x Mod 4 = 0) Then   '如果 x 能被 4 整除但不能被 100 整除
            Text2.Text = "yes"
```

```
        Else    '如果 x 不能被 4 和 100 整除
            Text2.Text = "no"
        End If
    ElseIf (x Mod 400 = 0) Then    '如果 x 能被 100 整除，又能被 400 整除
        Text2.Text = "yes"
    Else
        Text2.Text = "no"
        End If
    End If
End Sub
```

总是可以添加更多的 Else If **块到** If...Then **结构中去。但是，当每个** Else If **都将相同的表达式比作不同的数值时，这个结构编写起来很乏味。在这种情况下可以使用** Select Case **判定结构。**

3. IIf **函数**

实现一些简单的条件判断分支结构，其格式如下：

```
IIf(条件，条件为真时的值，条件为假时的值)
```

其作用是对条件进行测试，若条件成立（为真值），则取第一个值（即“条件为真时的值”），否则取第二个值（即“条件为假时的值”）。

例如：将 a、b **中的小数，放入** Min **变量中。**

```
Min=IIf(a<b,a,b)
```

4. Select Case

Visual Basic 6.0 **提供** Select Case **结构替代** If...Then...Else**，从而可在多个语句块中有选择地执行其中一个。**Select Case **语句的能力与** If...Then...Else **语句类似，但对多重选择的情况，**Select Case **语句使代码更加易读。**Select Case **在结构的上方便处理一个测试表达式并只计算一次。然后，**Visual Basic 6.0 **将表达式的值与结构中的每个** Case **的值进行比较。如果相等，就执行与该** Case **相关联的语句块。**

```
Select Case test expression
    [Case expressionlist1
        [statementblock-1]]
    [Case expressionlist2
        [statementblock-2]]
            ...
    [Case Else
        [statementblock-n]]
End Select
```

每一个“expressionlist**”是一个或几个值的列表。如果在一个列表中有多个值，就用逗号把值隔开。每一个“**statementblock**”中含有** 0 **个或多个语句。如果不止一个** Case **与测试表达式相匹配，则只对第一个匹配的** Case **执行与之相关联的语句块。如果在表达式列表中没有一个值与测试表达式相匹配，则** Visual Basic 6.0 **执行** Case Else **子句（此项是可选的）中的语句。例如，假定在** If...Then...Else **的例子中要向【编辑】菜单添加命令。为此可以另加一个**

Else If 子句，或用 Select Case 来写函数。

例如：根据一个同学的成绩，判断他的优、良、中、差等级。

```
Private Sub Command1_Click()
    Dim m As Long
    m = Val(Text1.Text)
    Select Case m
      Case 80 To 100     '80<=m<=100
         Text2.Text = "优"
      Case 70 To 79      '70<=m<=79
         Text2.Text = "良"
      Case 60 To 69      '60<=m<=69
         Text2.Text = "中"
      Case 0 To 59       '0<=m<=59
         Text2.Text = "差"
    End Select
 End Sub
```

这个应用程序也可以使用 If...Then...Else 结构编写，读者不妨试试。

Select Case 结构每次都要在开始处计算表达式的值。而 If...Then...Else 结构为每个 Else If 语句计算不同的表达式。只有在 If 语句和每一个 Else If 语句计算相同表达式时，才能用 Select Case 结构替换 If...Then...Else 结构。

4.7.3 循环结构

循环结构允许重复执行一行或数行代码。Visual Basic 6.0 **支持的循环结构有：**

- Do...Loop
- For...Next
- For Each...Next

1. Do...Loop

用 Do **循环重复执行一语句块，且重复次数不定。**Do...Loop **语句有几种演变形式，但每种都计算数值条件以决定是否继续执行。如同** If...Then condition **必须是一个数值或者值为“**True**”(非零）或“**False**”(零）的表达式。在下面的** Do...Loop **循环中，只要“**condition**”为“**True**”就执行“**statements**”。**

```
Do While (condition)
    statements
Loop
```

当 Visual Basic 6.0 **执行这个** Do **循环时会首先测试“**condition**”。如果“**condition**”为“**False**”(零)，则跳过所有语句。如果“**condition**”为“**True**”(非零)，则** Visual Basic 6.0 **执行语句，然后退回到** Do While **语句再测试条件。因此，只要“**condition**”为“**True**”或非零，循环可以随意执行多少次。如果“**condition**”一开始便为“**False**”，则不会执行语句。**

例如：我国现有人口约为 12 **亿，设年增长率为** 1%**，计算多少年后增加到** 20 **亿。**

```
Private Sub Form_Click()
```

```
    Dim a As Double
    Dim r As Single
    Dim i As Integer
    a = 12
    r = 0.01
    i = 0
    Do While a < 20     '当人口数大于等于 20 亿时结束循环
        a = a * (1 + r)
        i = i + 1
    Loop
    Print i ;"年后中国人口达到 20 亿"
End Sub
```

运行程序，单击窗体，程序输出为：

52 **年后中国人口达到** 20 **亿**

Do...Loop **语句的另一种演变形式是先执行语句，然后再在每次执行后测试“** condition **”。这种形式保证“** statements **”至少执行一次：**

```
Do
    statements
Loop While condition
```

其他两种演变形式类似于前两个，所不同的是，只要“ condition **”为“** False **”而不是“** True **”，它们就执行循环。**

将上例改写成这种格式如下：

```
Private Sub Form_Click()
    Dim a As Double
    Dim r As Single
    Dim i As Integer
    a = 12
    r = 0.01
    i = 0
    Do
        a = a * (1 + r)
        i = i + 1
    Loop  While a < 20       '当人口数大于等于 20 亿时结束循环
    Print i ;" 年后中国人口达到 20 亿"
End Sub
```

运行程序，单击窗体，程序输出为：

52 **年后中国人口达到** 20 **亿**

2. For...Next

在不知道循环内需要执行多少次语句时，宜用 Do **循环。但是，在知道要执行多少次时，则最好使用** For...Next **循环。与** Do **循环不同，**For **循环使用一个叫做计数器的变量，每重复一**

次循环之后，计数器变量的值就会增加或者减少。For 循环的语法如下：

```
For counter = start To end [Step increment]
    Statements
Next [counter]
```

参数“Counter”、“Start”、“end”和“increment”都是数值型的。

“increment”参数可正可负。如果“increment”为正，则“Start”必须小于等于“end”，否则不能执行循环内的语句。如果“increment”为负，则“Start”必须大于等于“end”，这样才能执行循环体。如果没有设置“Step”（步长），则“increment”缺省值为 1。每执行一个循环，“counter= counter+step”。

在执行 For 循环时，其执行过程如下：

① 设置“counter”等于“start”。

② 测试“counter”是否在“start”和“end”之间。若不是的话，则 Visual Basic 6.0 退出循环。若是的话执行语句。

③ 执行完“Statements”语句，执行 Next [counter]，counter= counter+step。

④ 重复步骤②到步骤④。

例如：求 m=1+2+3+4+……100 的值。

```
Private Sub Command1_Click()
    m = 0
    For i = 1 To 100    '步长省略，默认为 1
        m = m + i
    Next i
    Text1.Text = Str$(m)
End Sub
```

3. For Each...Next

For Each...Next 循环与 For...Next 循环类似，但它对数组中的每一个元素重复一组语句，而不是重复语句一定的次数。如果不知道一个集合有多少元素，For Each...Next 循环非常有用。

For Each...Next 循环的语法如下：

```
For Each element In group
    Statements
Next element
```

这里的“element”是一个变体型变量，它是为循环提供的，并在 For Each...Next 语句中重复使用，它实际上代表的是数组中的每个元素。“Group”是一个数组名，没有括号和上下界。

用 For Each...Next 语句可以对数组中的每个元素进行处理，它包括查询、显示和读取。他所重复的次数由数组中元素的个数决定，数组中有多少个元素，就自动重复执行多少次。例如：求数组 A 中的所有元素的和。

```
Dim sum As Integer
Dim A(1 To 10) As Integer
sum = 0
For Each x In A
    sum = sum + x
```

```
    Next x
```

因为数组中有 10 个元素，所以循环将执行 10 次。这里“x”是循环变量，但不需要为它提供初值和终值，而是根据数组元素个数决定循环次数。“x”的值在循环过程中是不断变化的，开始执行时，“x”是数组中第一个元素的值，执行完一次后“x”变为数组中第二个元素的值，当“x”为最后一个元素的值时，执行最后一次循环。

4.7.4 控制结构

前面介绍了几种基本的控制语句，而应用这几种基本的控制语句，可以产生各种各样的控制结构，下面讲解几种特殊的控制结构。

1. 嵌套控制结构

可以把控制结构放入另一个控制结构之内（例如在 For...Next 循环中的 If...Then 块）。一个控制结构内部包含另一个控制结构叫做 nest（嵌套）。在 Visual Basic 6.0 中，控制结构的嵌套层数没有限制。按一般习惯，为了使判定结构和循环结构更具可读性，总是用缩排方式书写判定结构或循环结构的正文部分。

例如：打印出两位数的素数。

```
Private Sub Form_DblClick()
    Dim i As Integer, j As Integer, m As Integer,p As Integer
    Print
    For i = 10 To 100
        m = 0
        p = Int(Sqr(i))
        For j = 2 To p      '内循环判断 i 是否是素数，若是则 m=0 否则 m=1
            If i Mod j = 0 Then
                m = 1
            End If
        Next j
        If m = 0 Then
            Print i;
        End If
    Next i
End Sub
```

执行程序的结果如图 4-5 所示。

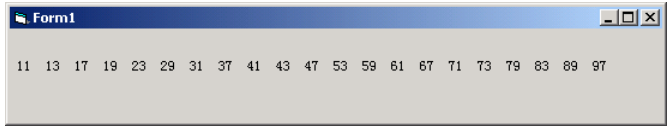


图4-5 两位数的素数

第一个 Next 关闭了内层的 For 循环，而最后一个 For 关闭了外层的 For 循环。同样，在嵌套的 If 语句中，End If 语句自动与最靠近的前一个 If 语句配对。嵌套的 Do...Loop 结构的工作方式也是一样的，最内圈的 Loop 语句与最内圈的 Do 语句匹配。

2. **退出控制结构**

用 Exit **语句可以直接退出** For **循环、**Do **循环、子过程或函数过程。**Exit **语句的语法很简单，**Exit For **在** For **循环中出现的次数没有限制，**Exit Do **在** Do **循环中出现的次数也没有限制。**

```
For counter = start To end [Step increment]
    [statementblock]
    [Exit For]
    [statementblock]
Next [counter[, counter] [,...]]

Do [{While | Until} condition]
    [statementblock]
    [Exit Do]
    [statementblock]
Loop
```

Exit Do **语句可以在** Do **循环语法的所有版本中使用。**Exit For **和** Exit Do **非常有用，因为它有时适用于立即退出循环，而且不再执行循环中的任何进一步迭代或者语句。**

例如：我国现有人口约为 12 **亿，设年增长率为** 1%**，计算多少年后增加到** 20 **亿。**

前面使用了 Do…Loop **语句求出了应用程序的结果，也可以在** For **循环中使用** Exit **语句也可以求出结果。例如：**

```
Private Sub Form_Click()
    Dim a As Double
    Dim r As Single
    Dim i As Integer
    a = 12
    r = 0.01
    For i=0 to 1000
        a = a * (1 + r)
            If a>=20 then       '当人口数大于等于 20 亿时跳出循环 Exit For
        End If
    Next i
    Print i; 年后中国人口达到 20 亿
End Sub
```

正如此例所表明的，Exit **语句几乎总是出现在** If **语句或** Select Case **语句内部，而** If **语句或** Select Case **语句在循环内嵌套。用** Exit **语句中断循环时，计数器变量的值会因退出循环方式的不同而不同：**

- **在完成循环时，计数器的值等于上限值加上步长值。**
- **在提前退出循环时，计数器变量保持其值，并遵从有关取值范围的一般规则。**

4.8 过程概述

将程序分割成较小的逻辑部件就可以简化程序设计任务，称这些部件为过程，它们可以变

成增强和扩展 Visual Basic 6.0 的构件。

过程可用于压缩重复任务或共享任务，例如，压缩频繁的计算、文本与控件操作和数据库操作。用过程编程有两大好处：

- 过程可使程序划分成离散的逻辑单元，每个单元都比无过程的整个程序容易调试。
- 一个程序中的过程，往往不必修改或只需稍作改动，便可以成为另一个程序的构件。

在 Visual Basic 6.0 中使用下列几种过程：

- Sub 过程不返回值。
- Function 过程返回值。
- property 过程返回并指定值以及设置对象引用。

4.8.1 Sub 过程

子过程是在响应事件时执行的代码块。将模块中的代码分成子过程后，在应用程序中查找和修改代码就变得更容易了。

4.8.1.1 定义子过程

子过程的语法是：

```
[Private|Public][Static]Sub procedurename (arguments)
    statements
End Sub
```

每次调用过程都会执行 Sub 和 End Sub 之间的“ statements ”。可以将子过程放入标准模块、类模块和窗体模块中。按照缺省规定，所有模块中的子过程为 Public（公用的），这意味着在应用程序中可随时调用它们。过程的“ arguments ”类似于变量声明，它声明了从调用过程传递进来的值。在 Visual Basic 6.0 中应区分通用过程和事件过程这两类子过程。

1. 通用过程

通用过程告诉应用程序如何完成一项指定的任务。一旦确定了通用过程，就必须专由应用程序来调用。反之，直到为响应用户引发的事件或系统引发的事件而调用通用过程时，事件过程通常总是处于空闲状态。

为什么要建立通用过程呢？理由之一就是几个不同的事件过程也许要执行同样的动作。可将公共语句放入一个分离开的过程（通用过程）并由事件过程来调用它，这样一来就不必重复代码，也容易维护应用程序。

2. 事件过程

当 Visual Basic 6.0 中的对象对一个事件的发生做出认定时，便自动用相应于事件的名字调用该事件的过程。因为名字在对象和代码之间建立了联系，所以说事件过程是附加在窗体和控件上的。

- 一个控件的事件过程将控件的（在“ Name ”属性中规定的）实际名字、下划线（_）和事件名组合起来。例如，如果希望在单击了一个名为“ cmdPlay ”的命令按钮之后，这个按钮会调用事件过程，则要使用 cmdPlay_Click 过程。
- 一个窗体事件过程将词汇“ Form ”、下划线和事件名组合起来。如果希望在单击窗体之后，窗体会调用事件过程，则要使用 Form_Click 过程。和控件一样，窗体也有惟一的名字，但不能在事件过程的名字中使用这些名字。如果正在使用 MDI 窗

体，则事件过程将词汇“ MDIForm ”、下划线和事件名组合起来，如 MDIForm_Load。

所有的事件过程都使用相同的语法。

控件事件的语法：

```
Private Sub controlname_eventname (arguments )
   Statements
End Sub
```

窗体事件的语法：

```
Private Sub Form_eventname (arguments)
   Statements
End Sub
```

虽然可以自己编写事件过程，但使用 Visual Basic 6.0 提供的代码过程会更方便，这个过程自动将正确的过程名包括进来。从【对象框】中选择一个对象，从【过程框】中选择一个过程，就可在【代码编辑器】窗口选择一个模板。在开始为控件编写事件过程之前先设置控件的“ Name ”属性，这不失为一个好主意。如果对控件附加一个过程之后又更改控件的名字，那么也必须更改过程的名字，以符合控件的新名字。否则，Visual Basic 6.0 无法使控件和过程相符。过程名与控件名不符时，过程就成为通用过程。

通用过程可以放在标准模块中，也可以放在窗体模块中，而事件过程只能放在窗体模块中，不同模块中的过程（包括事件过程和通用过程）可以相互调用。当过程名惟一时，可以直接通过过程名调用；如果两个或两个以上的标准模块中含有相同的过程名，则在调用是必须用模块名限定，其一般格式为：

模块名 . 过程名（参数列表）

4.8.1.2　建立 Sub 子过程

前面已经介绍如何建立事件过程，那么下面就介绍一下如何建立通用过程。通用过程不属于任何一个事件过程，因此不能放在事件过程中。通用过程可以在标准模块中建立，也可以在窗体模块中建立。

如果在标准模块中建立，可以是用以下两种方法。

- 第一种方法操作步骤如下：

1. 执行【工程】/【添加模块】命令，打开【添加模块】对话框，如图 4-6 所示，在该对话框总选择【新建】选项卡，然后双击【模块】图标，代开【代码】窗口。
2. 执行【工具】/【添加过程】命令，打开【添加过程】对话框，如图 4-7 所示。

图4-6 【添加模块】对话框

图4-7 【添加过程】对话框

3. 在【名称】文本编辑器框中输入要建立的过程的名字（例如“ Txt ”）。

4. 在【类型】栏内选择要建立的类型，如果建立子程序过程，则应选择【子程序】单选按钮；如果要建立函数过程，则应选择【函数】单选按钮。
5. 在【范围】栏内选择过程的适用范围，可以选择【公有的】单选按钮或【私有的】单选按钮。如果选择【公有的】单选按钮，则所建立的过程可用于本工程内的所有窗体模块；如果选择【私有的】单选按钮，则所建立的过程只能用于本标准模块。
6. 单击[确定]按钮，回到【模块代码】窗口，如图 4-8 所示。

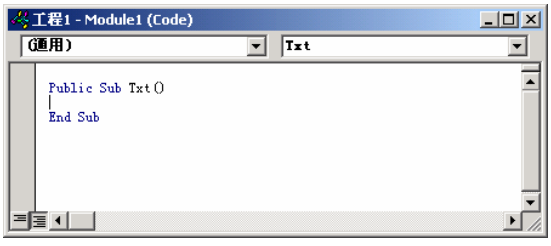


图4-8 【模块代码】窗口

此时，可以在光标跳动的地方键入程序代码（与事件过程代码键入方法相同）。

- 第二种方法：执行【工程】/【添加模块】命令，打开【代码】窗口，然后键入过程的名字。例如，键入“Public Sub Txt”后按回车键后显示如图 4-7 所示的程序。

在【模块代码】窗口中建立通用过程，则可双击窗体进入【代码】窗口，在【对象】框中选择“通用”，在【过程】框中选择“声明”，直接在窗体内键入“Sub Txt()”，然后按回车键，窗口内显示：

```
Sub Txt()
End Sub
```

此时即可键入代码。

4.8.1.3 调用 Sub 过程

在表达式中，Sub 过程不能用其名字调用。调用 Sub 过程的是一个独立的语句。Sub 过程还有一点与函数不一样，它不会用名字返回一个值。但是，与 Function 过程一样，Sub 过程也可以修改传递给它们的任何变量的值。

调用 Sub 过程有两种方法，以下两个语句都调用了名为“Txt”的 Sub 过程。

```
Call Txt (FirstArgument, SecondArgument)
Txt FirstArgument, SecondArgument
```

当使用“Call”语法时，参数必须在括号内。若省略“Call”关键字，则也必须省略参数两边的括号。

4.8.2 Function 过程

Visual Basic 6.0 包含内置的或内部的函数，如 Sqr、Cos 或 Chr。此外，还可用 Function 语句编写自己的 Function 过程。

4.8.2.1 定义 Function 过程

函数过程的语法是：

```
Private|Public][Static]Function procedurename (arguments) [As type]
```

```
    Statements
End Function
```

与 Sub 过程一样，Function 过程也是一个独立的过程，可读取参数，执行一系列语句并改变其参数的值。与子过程不同，Function 过程可返回一个值到调用的过程。在 Sub 过程与 Function 过程之间有 3 点区别：

- 一般说来，让较大的语句或表达式的右边包含函数过程名和参数(returnvalue=function)，这就调用了函数。
- 与变量完全一样，函数过程也有数据类型。这就决定了返回值的类型（如果没有 As 子句，缺省的数据类型为 Variant）。
- 给“ procedurename ”自身赋一个值，就可返回这个值。Function 过程返回一个值时，该值可成为较大表达式的一部分。

例如，下面是已知三角形三边长，计算三角形周长的函数：

```
Function P(A As Integer, B As Integer,C As Integer) As String
    P=A+B+C
End Function
```

在 Visual Basic 6.0 中调用 Function 过程的方法和调用任何内部函数的方法是一样的：

```
Text1.text = P(a,b,c)
```

4.8.2.2　建立 Function 过程

前一节提到建立 Sub 过程的 3 种方法也可以建立 Function 过程，只是当用第一种方法建立时，在对话框的【类型】栏内应选择【函数】单选框，另外两种方法中的 Sub 应换成 Function。

4.8.2.3　调用 Function 过程

通常，调用自行编写的函数过程的方法和调用 Visual Basic 6.0 内部函数过程（例如 Cos）的方法一样，即在表达式中写上它的名字。下面的语句都调用函数 Txt。

```
Print 10 * Txt
X = Txt
If Txt = 10 Then
Print "Out of Range";" X = ";10 * Txt
```

像调用 Sub 过程那样，也能调用函数。下面的语句都调用同一个函数：

```
Call Sin(x)
Sin x
```

4.8.3　参数使用

过程中的代码通常都需要某些关于程序状态的信息才能完成它的工作。信息包括在调用过程时传递到过程内的变量。当将变量传递到过程时，称变量为参数。参数分为形式参数和实际参数。

- 形式参数（简称形参）：在被调过程中的参数，出现在 Sub 过程和 Function 过程中。形式参数可以是变量名和数组名。
- 实际参数（简称实参）：在调用过程中的参数，过程调用时实参数据会传递给形参。

形参表和实参表中的对应变量名可以不同，但实参和形参的个数、顺序以及数据类型必须相同。

4.8.3.1 参数的数据类型

过程的参数被缺省为具有 Variant 数据类型。不过，也可以声明参数为其他数据类型。例如，下面的函数接受一个字符串和一个整数：

```
Function WhatsForLunch(WeekDay As String, Hour As Integer) As String
'根据星期几和时间，返回午餐菜单
If WeekDay = "Friday" then
    WhatsForLunch = "Fish"
Else
    WhatsForLunch = "Chicken"
End If
If Hour > 4 Then
    WhatsForLunch = "Too late"
End Function
```

4.8.3.2 参数的传递

参数传递可以实现调用过程和被调过程之间的信息交换，在过程的调用中，调用其他过程的过程称为主过程。被调用的过程称为子过程。当被调用的子过程要使用主过程中的数据时，那么就必须使用参数传递了。当以变量作为实参时，将会出现按值传递和按地址传递两种参数传递方式。

参数按何种方式传递，是在定义过程时决定的。如果在定义形参时前面加上“ByNal”关键字，就是传值方式；如果在定义形参时前面加上“ByRef”关键字，就是传址方式。如果在定义过程时缺省了这两个关键字，默认的是传址方式。

1. 按值传递参数

按值传递参数时，传递的只是变量的副本。如果过程改变了这个值，则所做的变动只影响副本而不会影响变量本身，即按传值方式时，只是把变量的值传递给形参。如果在过程中改变了这个形参的值，将不会影响到原变量。用“ByVal”关键字指出参数是按值来传递的。

例如：

```
Sub PostAccounts (ByVal intAcctNum as Integer)

'这里放语句

End Sub
```

按值传递参数时，过程内不会改变参数的值，如：

```
Private Sub Increase (ByVal x As Integer)
    x = x + 1
End Sub

Private Sub Decrease (ByVal x As Integer)
    x = x – 1
```

```
End Sub
```

在事件过程中调用：

```
Private Sub Form_Click ()
    Dim i As Integer
    i = 1000
    Increase i
    Print i
    Decrease i
    Print i
End Sub
```

其运行结果为：

1000
1000

例如交换两个变量值的过程为：

```
Public Sub Swap(ByVal x As Integer，ByVal y As Integer)
    Dim z As Integer
    z=x
    x=y
    Y=z
End Sub
```

如果再使用“swap a,b**”来调用过程，变量的值就不会再改变了。因为，**a、b **是以传值的方式传递的，在过程中形参再怎样改变都与它们无关了。**

因此，如果确定某个参数不需要在过程中改变，一定要在定义过程中在形参处加上“ByNa1**”关键字，以防止发生意外的错误。**

2. **按地址传递参数**

按传址方式时，则是把变量在内存中的地址传递给形参。这时，形参将与原变量使用内存中的同一地址。也就是说，如果在过程中改变了这个形参的值，原变量也会随之而改变。按地址传递参数在 Visual Basic 6.0 **中是缺省的，即不用“**ByVal**”关键字。**

如果给按地址传递参数指定数据类型，就必须将这种类型的值传给参数。可以给参数传递一个表达式，而不是数据类型。Visual Basic 6.0 **中的计算表达式，如果可能的话，还会按要求的类型将值传递给参数。**

把变量转换成表达式的最简单的方法就是把它放在括号内。例如，为了把声明为整数的变量传递给过程，该过程以字符串为参数，则可以用下面的语句：

```
Sub CallingProcedure ()
    Dim intX As Integerint
    X = 12 * 3
    Foo (intX)
End Sub
Sub Foo (Bar As String)
MsgBox Bar      'Bar 的值为字符串"36"
```

```
End Sub
```

按地址传递参数时，过程内改变了参数的值，例如：

```
Private Sub Increase (ByRef x As Integer)
    x = x + 1
End Sub
Private Sub Decrease (ByRef x As Integer)
    x = x – 1
End Sub
```

在事件过程中调用：

```
Private Sub Form_Click ()
    Dim i As Integer
    i = 1000
    Increase i
    Print i
    Decrease i
    Print i
End Sub
```

其运行结果为：

1001
1000

例如将前面介绍的交换两个变量值的过程改成如下形式：

```
Public Sub Swap(x As Integer，y As Integer)
    Dim z As Integer
    z=x
    x=y
    Y=z
End Sub
```

如果再使用“ swap a,b **”来调用过程，变量的值就会改变了。因为，**a、b **是以按地址的方式传递的。**

因此，如果确定某个参数不需要在过程中改变，一定要在定义过程中在形参处加上“ ByNa1 **”关键字，以防止发生意外的错误。**

4.8.4 数组参数的传递

数组参数的传递方法是，当需要把数组作为参数传递给过程时，只需要在定义形参时，形参名后面加一对括号就行。下面来看这样一个例子。

```
Private Sub Form Load()
    Dim x(7) As Inteter，i As integer，a As Integer  '定义数组及循环变量
    For i=1 To7         '为数组元素赋值
    X(i)=Inputbox("请输入一个整数：","输入数据",0)
    Next i
```

```
    A=Max(x())   '调用自定义函数
End Sub

Private Function Max(a() As Integer) As Integer   '定义函数过程
    Dim I As Integer, m As Integer
    M=a(1)
    For i=2 to 7
        If a(i)>m then
            m=a(i)   '将数组中的最大值存到变量 m 中
        End If
Next i
Max=m
End Sub
```

这个程序先定义了一个数组“a”，然后再用 1nputbox 函数输入 7 个整数存到该数组里，之后再调用自定义函数 Max 取出这 7 个数中最大的数。在这里传递的参数就是数组“x”。

当数组作为参数时，只能以传址方式传递，而不能使用传值方式。也就是说以数组作为参数时，可以在过程中改变数组元素的值。

另外，函数的返回值也可以是数组。只需在说明函数返回值类型时在后面加上一对括号即可。

4.8.5　可选参数与可变参数

Visual Basic 6.0 提供了十分灵活和安全的参数传递方式，允许使用可选参数和可变参数。在调用过程时，可以向过程传递可选的参数或者任意数量的参数。

4.8.5.1　可选参数

在前面介绍的函数中可以发现，过程中的形式参数是固定的，调用时使用的实参也是固定的。即如果一个过程中有两个形参，则调用时必须按相同的类型和顺序提供两个实参。但是在一些常用的函数中，有很多函数的某些参数是可选的，也就是说在调用函数时可以传递这些参数，也可以不传递这些参数。因为这些参数只是为了提供某项功能，但这项功能不是每次使用时都需要用到或使用默认就行了。

可以在定义过程时对可选的参数加上“Optional”关键字，并且可以在过程体中通过 IsMissing 函数测试调用是否传递可选参数。例如：

```
Sub Sum(N1 As Integer,N2 As Integer,Option N3)
    Dim I,S As Integer
    S=N1+N2
    If IsMissing(N3) Then
        S=S+N3
    End If
End Sub
```

在过程体中首先计算前两个参数的和，并把结果赋给变量“S”，然后测试第 3 个参数是

否存在，如果存在，则把第 3 个参数与前两个参数的乘积相乘，最后输出。

在上述过程中有 3 个参数，其中前两个参数与普通过程中的书写格式相同，最后一个参数没有制定类型（默认为变体形变量 Variant），而在前面加上了“ Optional ”，表明该参数是一个可选参数。可选参数必须放在参数表的最后，而且必须是变体形变量（Variant）。

IsMissing 函数有一个参数，它就是由“ Optional ”指定的形参的名字，其返回值为 Boolean 类型。在调用过程时，如果没有向可选参数传递实参，则 IsMissing 函数的返回值为“ True ”，否则返回值为“ False ”。

使用可选参数时要注意：若某个参数被指定为可选参数，则它后面所有的参数都必须是可选的。

4.8.5.2 可变参数

在定义过程时，有时可能不知道会传递多少个参数，这时就需要用到不定个数的参数。方法是在定义过程时用“ ParamArray ”关键字来定义一个可变参数，可以把这个参数看作是一个动态数组，其大小随参数的个数而定。

例如要定义一个求多个整数之和的函数过程，但又不知道可能会出现多少个数，可以使用如下的方法定义函数：

```
Private Function,MySum(ParamArray X()) As Long
    Dim i AS Integer，sum As Long
    Dim n AS Integer
    N=UBound(x)  '取出该数组的长度(即不定参数的个数)做为循环终值
    For i=0 To n
    Sum=sum+x(i) '求和
    Next I
    MySum=sum '设置函数返回值
End FunctiOn
```

在这个函数中就用到了可变参数，定义过该函数后，就可以在程序中调用该函数了。例如：a＝MySum(3,8,5)或 b=MySum(1,2,3,4,5,6)。执行过这两个语句后，变量 a、b 的值分别 16 和 21。

在使用“ ParamArray ”关键字时要注意以下几点：

- “ ParamArray ”关键字定义的动态数组只能是变体型的，且其下界总是 0。
- “ ParamArray ”关键字不能与“ ByVal ”、“ ByReF ”、“ Optional ”一起使用。
- “ ParamArray ”关键宇只能用于最后一个参数。
- 若要使用不定个数参数，则其他参数不能被指定为可选的。

4.9 小结

本章着重介绍了编写 Visual Basic 6.0 程序所需要掌握的基本知识和 Visual Basic 6.0 编程的基本思想方法。这主要包括对象和类、事件驱动的编程机制、主要数据类型、变量和数组、主要过程控制以及过程和函数。在学习本章之后，读者便具有了一定的 Visual Basic 6.0 编程的基础知识，为以后的编程打下基础

对象和类是 Visual Basic 6.0 面向对象编程的最显著的特点，在 Visual Basic 6.0 中，各种

编程要素实际上都是由类构成的，读者应很好地了解这种编程的思想。事件驱动的编程机制是 Visual Basic 6.0 应用程序，也是 Windows 应用程序运行的基本特点。

4.10　习题

一、 选择题

1. 以下叙述中错误的是（　）。

   A. 如果过程被定义为 Static 类型，则该过程中的局部变量都是 Static 类型

   B. Sub 过程中不能嵌套定义 Sub 过程

   C. Sub 过程中可以嵌套调用 Sub 过程

   D. 事件过程可以像通用过程一样由用户定义过程名

2. 一个工程中包含两个名称分别为 Form1、Form2 的窗体、一个名称为 Func 的标准模块。假定在 Form1、Form2 和 Func 中分别建立了自定义过程，其定义格式为：

   Form1 中定义的过程：

```
Private Sub Fun1( )
……
End Sub
```

   Form2 中定义的过程：

```
Private Sub Fun2( )
……
End Sub
```

   Func 中定义的过程：

```
Private Sub Fun3( )
……
End Sub
```

在调用上述过程的程序中，如果不指明窗体或模块的名称，则以下叙述中正确的是（　）。

   A. 上述 3 个过程都可以在工程中的任何窗体或模块中被调用

   B. Fun1 和 Fun2 过程能够在工程中各个窗体或模块中被调用

   C. 上述 3 个过程都只能在各自被定义的模块中调用

   D. 只在 Fun3 过程能够被工程中各窗体或模块调用

3. 以下叙述中错误的是（　）。

   A. 一个工程中可以包含多个窗体文件

   B. 在一个窗体文件中用 Private 定义的通用过程能被其他窗体调用

   C. 在设计 Visual Basic 6.0 程序时，窗体、标准模块、类模块等需要分别保存为不同类型的磁盘文件

   D. 全局变量必须在标准模块中定义

4. 下列变量名中，合法的变量名是（　）。

   A. C24　　B. A.B　　C. A：B　　D. 1+2

5. 可以同时删除字符串前导和尾部空白的函数是（　）。

   A. Ltrim　　B. Rtrim　　C. Trim　　D. Mid

6. 设 a="Visual Basic"，下面使 b="Basic"的语句是（　）。

A. b=Left(a,8,12)　　B. b=Mid(a,8,5)

C. b=Rigth(a,5,5)　　D. b=Left(a,8,5)

7. **设有如下声明：**

```
Dim X As Integer
```

如果 Sgn(X) **的值为**-1**，则** X **的值是（　）。**

A. **整数**　　B. **大于** 0 **的整数**　　C. **等于** 0 **的整数**　　D. **小于** 0 **的数**

8. **执行以下程序段后，变量** c$**的值为（　）。**

```
a$="Visual Basic Programing"
b$="Quick"
c$=b$ & UCase(Mid$(a$,7,6)) & Right $ (a$,11)
```

A. Visual BASIC Programing

B. Quick Basic Programing

C. QUICK Basic Programing

D. Quick BASIC Programing

9. **表达式** 4+5 \ 6 * 7 / 8 Mod 9 **的值是（　）。**

A. 4　　B. 5　　C. 6　　D. 7

10. **如果执行以下操作：**

```
a=8 <CR> (<CR>是回车键，下同)
b=9 <CR>
print a>b <CR>
```

则输出结果是（　）。

A. –1　　B. 0　　C. False　　D. True

11. **当** Visual Basic 6.0 **执行下面语句后，**A **的值为（　）。**

```
A=1
If A>0 Then A=A+1
If A>1 Then A=0
```

A. 0　　B. 1　　C. 2　　D. 3

12. **在窗体中添加一个命令按钮** Command1**，并编写如下程序：**

```
Private Sub Command1_Click( )
    x=InputBox(x)
    If x^2=9 Then y=x
    If x^2<9 Then y=1/x
    If x^2>9 Then y=x^2+1
    Print y
End Sub
```

程序运行后，在 InputBox **中输入** 3**，单击命令按钮，程序的运行结果是（　）。**

A. 3　　B. 0.33　　C. 17　　D. 0.25

13. **程序运行后，如果单击命令按钮，则窗体上显示的内容是（　）。**

```
Private Sub Form_Click()
score = Int(Rnd * 10) + 30
```

```
Select Case score
Case Is < 10
a$ = "F"
Case 10 To 19
a$ = "D"
Case 20 To 29
a$ = "C"
Case 30 To 39
a$ = "B"
Case Else
a$ = "A"
End Select
Print a$
End Sub
```

程序运行后，单击窗体，则在窗体上显示的是（　）。

A. A　　B. B　　C. C　　D. D

14. **执行下面的程序段后，**x **的值为（　）。**

```
x=5
For i=1 To 20 Step 2
x=x+i\5
Next i
```

A. 21　　B. 22　　C. 23　　D. 24

15. **阅读下面的程序段：**

```
For i=1 To 3
For j=1 To i
For k=j To 3
a=a+1
Next k
Next j
Next i
```

执行上面的三重循环后，a **的值为（　）。**

A. 3　　B. 9　　C. 14　　D. 21

16. **在窗体上画一个命令按钮，然后编写如下事件过程：**

```
Private Sub Command1_Click()
For i=1 To 4
x=4
For j =1 To 3
x=3
For k=1 To 2
x=x+6
```

```
Next k
Next j
Next i
Print x
End Sub
```

程序运行后，单击命令按钮，输出结果是（ ）。

A. 7　　B. 15　　C. 157　　D. 538

17. **在窗体上画两个文本框（其** Name **属性分别为** Text1 **和** Text2**）和一个命令按钮（其** Name **属性为** Command1**），然后编写如下事件过程：**

```
Private Sub Command1_Click()
x=0
Do While x<50
x=(x+2)*(x+3)
n=n+1
Loop
Text1.Text=Str(n)
Text2.Text=Str(x)
End Sub
```

程序运行后，单击命令按钮，在两个文本框中显示的值分别为（ ）。

A. 1 **和** 0　　B. 2 **和** 72　　C. 3 **和** 50　　D. 4 **和** 168

18. **用下面语句定义的数组的元素个数是（ ）。**

```
Dim A (-3 To 5) As Integer
```

A. 6　　B. 7　　C. 8　　D. 9

19. **以下程序的输出结果是（ ）。**

```
Private Sub Command1_Click()
Dim a(10),p(3) As Integer
k=5
For i=1 To 10
a(i)=i
Next i
For i=1 To 3
p(i)=a(i*i)
Next I
For i=1 To 3
k=k+p(i)*2
Next I
Print k
End sub
```

A. 33　　B. 28　　C. 35　　D. 37

20. **假定有如下的** Sub **过程：**

```
Sub S（x As Single,y As Single）
t = x
x =t/y
y =t Mod y
End Sub
```

在窗体上画一个命令按钮，然后编写如下事件过程：

```
Private Sub Commandl_Click ( )
Dim a As Single
Dim b As Single
a =5
b =4
Sub
Print a,b
End Sub
```

程序运行后，单击命令按钮，输出结果为（　）。

A. 5 4　　B. 1 1　　C. 1.25 4　　D. 1.25 1

21. **阅读程序：**

```
Function F(a As Integer)
b = 0
Static c
b = b+1
c = c+1
f = a+b+c
End Function
Private Sub Commandl_Click ()
Dim a As Integer
a =2
For i =1 To 3
Print F(a)
Next i
End Sub
```

运行上面的程序，单击命令按钮，输出结果为（　）。

A. 4　　B. 4　　C. 4　　D. 4

4　　5　　6　　7

4　　6　　8　　9

22. **阅读程序：**

```
Sub subP(b() As Integer)
For i =1 To 4
b(i)=2*i
Next i
```

```
End Sub
Private Sub Commandl_Click()
Dim a(1 To 4)As Integer
a (1)=5
a (2)=6
a (3)=7
a (4)=8
subP a ()
For i =1 To 4
Print a(i)
Next i
End Sub
```

运行上面的程序，单击命令按钮，输出结果为（ ）。

A. 2　　B. 5　　C. 10　　D. **出错**

4　　6　　12

6　　7　　14

8　　8　　16

23. **在窗体上画一个名称为** Commandl **的命令按钮，然后编写如下程序：**

```
Private Sub Command1_Click()
    Static X As Integer
    Static Y As Integer
    Cls
    Y=1
    Y=Y+5
    X=5+X
    Print X,Y
End Sub
```

程序运行时，三次单击命令按钮 Commandl **后，窗体上显示的结果为（ ）。**

A. 15　16　　B. 15　6　　C. 15　15　　D. 5　6

二、 填空题

1. **编写** Visual Basic 6.0 **程序代码需要在_____窗口进行。**
2. **为了在整个应用程序中用常量** Pi **来代替** 3.1416**，应在主窗体口的顶层声明中使用语句：_____。**
3. **设有如下的** Visual Basic **表达式：**

```
5 * x^2 - 3 * x - 2 * Sin(a)/3
```

它相当于代数式_____。

4. **表达式** Fix(-32)+Int(-24)**的值为_____。**
5. **随机生成一个** 1~6 **的随机整数的表达式是_____。**
6. **下面程序运行后，输出的结果为_____。**

```
Private Sub Command1_Click( )
```

```
A$="Beijing"
B$="dalian"
C$="shanghai"
C$=Instr(LeftA$,2)+Right$(B$,2),C$)
Print C$
End Sub
```

7. **执行下面的程序段后，s 的值为_____。**

```
s=5
For i = 2.6 To 4.9 Step 0.6
s=s+1
Next i
```

执行下面的程序段后，b 的值为_____。

```
a=300
b=20
a=a+b
b=a-b
a=a-b
```

8. **在窗体画一个命令按钮，然后编写如下事件过程：**

```
Private Sub Command1_Click()
    Dim a(1 To 10)
    Dim p(1 To 3)
    k=5
    For i=1 To 10
    a(i)=i
    Next i
    For i=1 To 3
    p(i)=a(i*i)
    Next i
    For i=1 To 3
    k=k+p(i)*2
    Next i
    Print k
End Sub
```

程序运行后，单击命令按钮，输出结果是_____。

9. **在窗体上画一个命令按钮，其名称为 Command1，然后编写如下程序：**

```
Function M(x As Integer,y As Integer)As Integer
    M=IIf(x>y,x,y)
End Function
Private Sub command1_Click()
    Dim a As Integer,b As Integer
```

```
        a=100
        b=200
        Print M(a,b)
    End Sub
```

程序运行后，单击命令按钮，输出结果为_____。

三、 编程题

1. **根据根据输入的** x **的值分别计算两个数的和、差、积、商。**
   X：1 **为求和，**2 **为差，**3 **为积，**4 **为商**
2. **计算表达式的值：**

```
y= x   (x<1)
y=2x-1   (1<=x<10)
y=3x-11  (x>=10)
```

3. **多位数分位显示（不能带有四舍五入）。**
   如：5468799 **显示为** 5 4 6 8 7 9 9
4. **求** 1+2+3+4+ ····· n。1/1+1/2+1/3+……1/n。
   要求：n **从一个文本框输入，**1/1 **到** 1/n **的和从另一文本框输出。**
5. **求从** a **到** b **的偶数和，当和数大于** 400 **是不再求和，最后输出结果。**
   要求：a 、b **从文本框输入，计算结果从文本框** 3 **中输出。**
6. **求** n!=1×2×3 ····· ×n。
   要求：n **从文本框输入，**1 **到** n **的和从另一文本框输出。**
   从键盘上输入 10 **个整数，并放入一个一维数组中，然后将其前** 5 **个元素与后** 5 **个元素对换，即第** 1 **个元素与第** 10 **个元素互换，第** 2 **个元素与第** 9 **个元素互换……第** 5 **个元素与第** 6 **个元素互换。分别输出数组原来的各元素的值和对换后各元素的值。**
7. **一个两位的正整数，如果将它的个位数字与十位数字对调，则产生另一个正整数，我们把后者叫做前者的对调数。现给定一个两位的正数，请找到另一个两位的正整数，使得这两位正整数之和等于它们积各自的对调数之和。例如，**12+32=3=23+21。**编写程序，把具有这种特征的一对两位正整数都找出来。下面是其中的一种结果：**

```
56+（10）=（1）+65        56+（65）=（56）+65
56+（21）=（12）+65       56+（76）=（67）+65
56+（32）=（23）+65       56+（87）=（78）=65
56+（43）=（34）+65       56+（98）=（89）+65
56+（54）=（45）=65
```

8. **编写一个计算矩形面积的** Sub **过程，然后调用过程计算面积。**
9. **编写一个求最大公约数的函数过程。要求在** text1、text2 **中输入** a、b **值，然后在** text3 **中输出最大公约数。**
10. **编写一个** Function **过程，求数组的最大值。**
11. **编写一个** Function **过程求前** n **个数的和，即** m=1+2+3+……+n**，调用次函数求出** S=11+（1+2）+（1+2+3）+…（1+2+3+4…+N）**的值。**
12. **编写一个** Function **过程，判断一个数是否是素数。调用这个函数，在窗体上打印出** 100 **以内的所有素数。（说明：只能被** 1 **和本身整除的数为素数。）**

第5章　Visual Basic 6.0 常用控件

通过前面几章基础知识的学习，读者已经知道，控件是用户界面最基本的组成元素，每个控件都有自己的属性、事件和方法，灵活地使用这些属性、事件和方法，便可以实现一些特定的任务。在 Visual Basic 6.0 中，控件主要有 3 类，即常用控件、ActiveX 控件、可插入对象控件。在工具箱中所显示的控件都是常用控件，也叫做标准控件，如图 5-1 所示。本章将介绍如何使用这些标准控件。

本章学习目标

- 控件的添加。
- 控件的共有属性。
- 控件的共有事件。
- 标签控件。
- 文本框控件。
- 命令按钮控件。
- 单选按钮、复选按钮控件和框架控件。
- 列表框、组合框控件。
- 滚动条控件。
- 定时器控件。
- 控件命名约定。

图5-1　工具箱

5.1 控件的添加

在用 Visual Basic 6.0 设计应用程序时，必须先向窗体中添加控件，以便完成应用程序界面的设计。要将某一控件添加到窗体中，有以下两种方法：

- 在工具箱中，双击对应的控件图标。

- 在工具箱中，单击对应的控件图标，然后将鼠标移到窗体上，这时鼠标变为“＋”字形，在窗体上按住鼠标左键，拖动鼠标，在鼠标拖动一定范围后，松开鼠标左键。

除了可以在窗体上添加控件外，还可以向容器类控件（图片框控件、框架控件）中添加控件。向容器类控件中添加控件时，必须先在工具箱中单击对应的控件图标，然后将鼠标移到容器类控件上，在容器类控件上拖动添加控件。拖动鼠标时，鼠标不要拖到容器类控件的外面去。

为窗体添加控件之后，在窗体上单击控件，便可以选中控件，在窗体上拖动控件，便可以改变控件的位置；拖动控件的边角，便可以改变控件的大小。在窗体上选中 1 个控件，按住 Shift 键，单击其他控件，这时便可以同时选中多个控件。当选中多个控件后，单击【格式】/【对齐】菜单等可以选择控件间的对齐方式；单击【格式】/【统一尺寸】/【高度相同】菜单等可以调整控件间的大小关系；单击【格式】/【水平间距】/【相同间距】菜单等可以调整控件间的水平间距；单击【格式】/【垂直间距】/【相同间距】菜单等可以调整控件间的垂直间距。

调整控件的位置及大小时，要学会灵活使用【格式】菜单的各个子菜单。【格式】菜单只有在选中多个控件时才可用，对于单个控件而言不可用，并且在调整控件的间距时，至少要有 3 个控件被选中。

5.2 控件的公共属性

在 Visual Basic 6.0 中，每个控件都有自己的属性，并且这些属性一般都会显示在【属性】窗口中。在控件的众多属性中，有一部分属性是大部分控件都有的，主要包括：“Name”属性、“Appearance”属性、“BackColor”属性、“Caption”属性、“Enabled”属性、“ForeColor”属性、“Font”属性、“Height”属性、“Left”属性、“Top”属性、“Width”属性、“Visible”属性等。

设置某个控件的属性时，我们必须先在窗体中选中该控件。

(1) “Name”属性

功能：访问和区分控件。

说明：每个控件都必须有 1 个名称。名称属性就如同控件的“姓名”，“姓名”就直接代表着控件本身。

(2) “Appearance”属性

功能：返回或设置控件的外观样式。

说明：“Appearance”属性有 2 个取值：0 或 1。“Appearance”属性为 0 时，表示将控件的外观设为平面的样式；“Appearance”属性为 1 时，表示将控件的外观设为三维的样式。

(3) “BackColor”属性

功能：返回或设置控件背景的颜色。

说明：设置“BackColor”属性将会直接改变控件的底色。

(4) “Caption”属性

功能：返回或设置控件上所显示的文本。

说明：只有那些不能接受用户输入的控件才有该属性，比如说标签控件、命令按钮控件

等。

(5)　“ Enabled ”属性

功能：返回或设置控件是否可用。

说明：“ Enabled ”属性有两个取值：“ True ”或“ False ”；“ Enabled ”属性为“ True ”（默认值）时，表示控件可用，可以响应用户的操作；“ Enabled ”属性为“ False ”时，控件为灰色，表示控件不可用，不能响应用户的操作。

(6)　“ ForeColor ”属性

功能：返回或设置控件的前景颜色。

说明：设置“ ForeColor ”属性将会影响图形及文本的颜色。

(7)　“ Font ”属性

功能：返回或设置控件文本所用的字体名、字体样式及字体大小。

(8)　“ Height ”属性、“ Width ”属性

功能：“ Height ”属性返回或设置控件的高度；“ Width ”属性返回或设置控件的宽度。

说明：控件的大小可以通过拖动控件边角来改变，也可以通过设置“ Height ”、“ Width ”属性来改变。

(9)　“ Left ”属性、“ Top ”属性

功能：“ Left ”属性返回或设置控件左边与其容器（窗体、图片框控件或框架控件）左边间的距离；“ Top ”属性返回或设置控件顶部与其容器（窗体、图片框控件或框架控件）顶部之间的距离。

说明：控件的位置可以通过拖动控件来改变，也可以通过设置“ Left ”、“ Top ”属性来改变。

(10)　“ Visible ”属性

功能：返回或设置控件是否可视。

说明：“ Visible ”属性有两个取值：“ True ”或“ False ”，“ Visible ”属性为“ True ”（默认值）时，表示控件可见；“ Visible ”属性为“ False ”时，表示控件不可见。

除了以上 10 种公共属性之外，另外还有一些常见的公共属性，详见附表一。

5.3　控件的公共事件

通过前面基础知识的学习，读者知道任何控件都可以响应一定的事件。控件不同，可以响应的事件也不同，并且对于同一事件，不同的控件可有不同的响应方式。控件可以响应的事件有很多，可以在这些事件中编写相应的代码，让控件执行不同的响应事件，从而实现特定的效果或任务。在众多事件中，有一部分事件是大部分控件都可以响应的公共事件，主要包括鼠标事件、拖动事件、键盘事件、焦点事件等。在为控件添加事件之前，必须先明确事件激发的顺序，以确保操作互不冲突。

在【代码】窗口，从对象列表框中选择相应的控件名称，然后在事件/过程列表框中可以查看该控件所能响应的事件，单击某个事件便可以为控件添加相应的事件。在窗体上双击控件，便可以为控件添加最常用的事件。

5.3.1　鼠标事件

鼠标是应用程序中最常用的输入设备，因此在设计应用程序时必须灵活地控制鼠标。要控

制鼠标，必须先要了解与鼠标有关的事件。鼠标事件主要包括：单击事件（Click 事件）、双击事件（DblClick 事件）、鼠标按下事件（MouseDown 事件）、鼠标弹起事件（MouseUp 事件）、鼠标移动事件（MouseMove 事件）。

单击事件（Click 事件）是鼠标事件中应用最广的事件，大多数控件都能响应该事件。在窗体上单击某个控件，便会激发 Click 事件，其语法结构如下：

```
Private Sub 控件名_Click()


End Sub
```

单击控件除了激发 Click 事件之外，还将激发 MouseDown 事件和 MouseUp 事件。Click、MouseDown、MouseUp 这 3 个事件所发生的顺序因控件的不同而不同。例如，对于列表框控件和命令按钮控件，这 3 个事件按以下顺序发生：MouseDown、Click、MouseUp。对于文件列表控件、标签控件、图片框控件，这 3 个事件按以下顺序发生：MouseDown、MouseUp、Click。

【例5-1】 编写一个程序，实现以下功能：

- 在窗体上按下鼠标时，窗体的标题为“鼠标被按下”。
- 松开鼠标时，窗体的标题为“鼠标被松开”。

程序编制步骤及相应代码如下。

1. 启动 Visual Basic 6.0，新建 1 个工程。
2. 单击窗体资源管理器的查看代码图标，打开【代码】窗口，如图 5-2 所示。

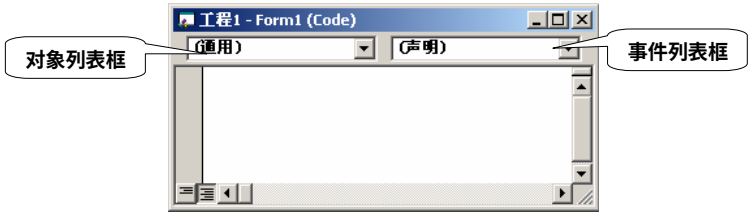


图5-2 【代码】窗口

3. 单击对象列表框右端的箭头，打开对象列表，然后单击“Form1”项，这时就会为窗体添加了默认的加载事件（Load 事件）。
4. 单击事件/对象列表框右端的箭头，打开事件列表，然后单击“MouseDown”项，这时便为窗体添加了鼠标按下事件（MouseDown 事件），在事件中添加如下代码：

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, _
X As Single, Y As Single)
    Form1.Caption = "鼠标被按下"
End Sub
```

5. 以同样的方法为窗体添加鼠标弹起事件（MouseUp 事件），并在 MouseUp 事件中添加如下代码：

```
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, _
X As Single, Y As Single)
    Form1.Caption = "鼠标被松开"
```

```
End Sub
```

6. 单击工具栏上的【启动】按钮或直接按 F5 键，运行程序。
7. 在窗体上，按住鼠标左键，这时窗体的标题变为“鼠标被按下”，然后松开鼠标左键，这时窗体的标题变为“鼠标被松开”。

当在某个控件上按下鼠标时，便会激发鼠标按下事件，即 MouseDown 事件；如果松开鼠标，便会激发鼠标弹起事件，即 MouseUp 事件；在控件上移动鼠标，便会激发鼠标移动事件，即 MouseMove 事件。

上述 3 种鼠标事件的语法结构如下。

- 鼠标按下事件

```
Private Sub 控件名_MouseDown(Button As Integer,Shift As Integer, _
X As Single,Y As Single)
End Sub
```

- 鼠标弹起事件

```
Private Sub 控件名_MouseUp(Button As Integer, Shift As Integer, _
X As Single, Y As Single)
End Sub
```

- 鼠标移动事件

```
Private Sub 控件名_MouseMove(Button As Integer,Shift As Integer, _
X As Single,Y As Single)
End Sub
```

以上 3 个鼠标事件过程都具有相同的参数，即“Button”、“Shift”、“X”、“Y”，这 4 个参数是由系统给出的，而不需用户去给定，各个参数的说明如下。

- “Button”参数

“Button”是一个整型参数，用来获取用户所按下的鼠标键，其取值见表 5-1。

表 5-1　　“Button”参数值

Button 值	常量	说明
000（十进制 0）		未按任何键
001（十进制 1）	vbLeftButton	左键被按下（默认值）
010（十进制 2）	vbRightButton	右键被按下
011（十进制 3）	VbLeftButton+ vbRightButton	同时按下左键和右键
100（十进制 4）	vbMiddleButton	中键被按下
101（十进制 5）	VbMiddleButton+ vbLeftButton	同时按下中键和左键
110（十进制 6）	VbMiddleButton+ vbRightButton	同时按下中键和右键
111（十进制 7）	VbMiddleButton+ vbLeftButton+ vbRightButton	3 个键同时被按下

对于 MouseDown、MouseUp 事件，“Button”参数的取值只能有 3 种，即 001（十进制 1）、010（十进制 2）和 100（十进制 3），而对于 MouseMove 事件，“Button”参数可取表 5-1 中任何值。

- “Shift”参数

“Shift”是一整型参数，用于获取 Shift、Ctrl、Alt 键的状态，其取值见表 5-2。

表 5-2 “Shift”参数值

Shift 值	常量	说明
000（十进制 0）		未按任何键
001（十进制 1）	vbShiftMask	按下 Shift 键
010（十进制 2）	vbCtrlMask	按下 Ctrl 键
011（十进制 3）	vbShiftMask+ vbCtrlMask	同时按下 Shift 键和 Ctrl 键
100（十进制 4）	vbAltMask	按下 Alt 键
101（十进制 5）	vbAltMask+ vbShiftMask	同时按下 Alt 键和 Shift 键
110（十进制 6）	vbAltMask+ vbCtrlMask+ vbShiftMask	同时按下 Alt 键和 Ctrl 键
111（十进制 7）	vbAltMask+ vbCtrlMask+ vbShiftMask	3 个键同时被按下

“Shift”参数反映了在按下鼠标的同时，Shift、Ctrl、Alt 这 3 个键的状态。

- “X”,“Y”参数

“X”,“Y”参数用于记录鼠标指针所在的位置，其中参数“X”记录指针的横坐标，参数“Y”记录指针的纵坐标。参数“X”,“Y”随着鼠标的移动而改变。

【例5-2】 编写 1 个程序，测试当前在窗体上所单击的鼠标键、鼠标的位置以及 Shift、Ctrl、Alt 键的状态。

程序编制步骤及相应代码如下。

1. 新建 1 个工程。
2. 按【例 5-1】所讲的方法为窗体添加 MouseDown 事件，并在事件中添加如下代码：

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, _
X As Single, Y As Single)
    Form1.AutoRedraw = True
    Print "鼠标指针横坐标："; X; "纵坐标："; Y
    If Button = 1 And Shift = 0 Then
        Print "只按下鼠标左键"
    ElseIf Button = 2 And Shift = 0 Then
        Print "只按下鼠标右键"
    ElseIf Button = 1 And Shift = 1 Then
        Print "同时按下鼠标左键和 Shift 键"
    ElseIf Button = 2 And Shift = 1 Then
        Print "同时按下鼠标右键和 Shift 键"
    End If
End Sub
```

3. 单击工具栏上的【启动】按钮或直接按 F5 键，运行程序。
4. 在窗体上单击鼠标左键或右键或单击鼠标的同时按下 Shift 键，则窗体上便会显示相应的提示信息，如图 5-3 所示。

图5-3　程序运行后的界面

5.3.2　键盘事件

键盘也是应用程序中常用的输入设备之一，在用键盘完成输入功能时，同样会激发与键盘有关的事件。与键盘相关的事件主要有按键事件（KeyPress 事件）、键按下事件（KeyDown 事件）、键弹起事件（KeyUp 事件）。

【例5-3】　编写 1 个程序，测试所按的键是数字键还是字母键。

程序编制步骤及相应代码如下。

1. 新建 1 个工程。
2. 单击窗体资源管理器的查看代码图标，打开【代码】窗口。
3. 在对象列表框中选择窗体“ Form1 ”，在事件列表框中选择“ KeyPress ”事件，为窗体添加 KeyPress 事件，并在事件中添加如下代码：

```
Private Sub Form_KeyPress(KeyAscii As Integer)
    Form1.AutoRedraw = True
    If Chr(KeyAscii) >= "0" And Chr(KeyAscii) <= "9" Then
        Print "按下的是数字键"
    ElseIf Chr(KeyAscii) >= "A" And Chr(KeyAscii) <= "z" Then
        Print "按下的是字母键"
    End If
End Sub
```

4. 运行程序，在键盘上按下任意 1 个数字键或字母键，这时窗体上便会显示所按键的类型，如图 5-4 所示。

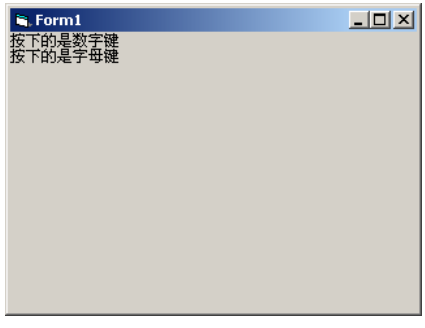


图5-4　程序运行后的界面

在程序运行的过程中，当按下键盘中的某个键时，除了激发 KeyPress 事件之外，还会激

发 KeyDown 事件；松开所按下的键时，便会激发 KeyUp 事件，各个事件的语法结构如下。

- KeyPress 事件

```
Private Sub 控件名_KeyPress(KeyAscii As Integer)

End Sub
```

- KeyDown 事件

```
Private Sub 控件名_KeyDown(KeyCode As Integer, Shift As Integer)

End Sub
```

- KeyUp 事件

```
Private Sub 控件名_KeyUp(KeyCode As Integer, Shift As Integer)

End Sub
```

在以上 3 个事件中，“ KeyAscii ”、“ KeyCode ”都是整数型参数，用来获取当前所按键的键码，“ KeyAscii ”获取的是按键上字符的 ASCII 码，“ KeyCode ”获取的是按键的扫描码，这两个参数都是由系统自动传递过来的，不需要用户自己另外去设置。例如，在【例 5-3】中，便是通过参数“ KeyAscii ”来区分所按键的类型。

键盘的每个键都有 1 个 ASCII 码和扫描码，扫描码反映的是按键的位置信息，而 ASCII 码反映的是标准的字符信息，因此“ KeyCode ”参数不能区分大小写，即大写 A 和小写 a 所对应的“ KeyCode ”值是一样的，都为 65，而“ KeyAscii ”参数则可以区分大小写。

在默认情况下，控件的键盘事件优先于窗体的键盘事件，因此一旦发生键盘事件，则总是控件先响应键盘事件。如果希望窗体先响应键盘事件，则必须将窗体的“ KeyPreview ”属性设为“ True ”。

参数“ KeyAscii ”返回的是整型数值，使用 Chr 函数便可以将“ KeyAscii ”参数转化为相应的字符，语法结构如下：

Chr(KeyAscii)

5.3.3 焦点事件

焦点（Focus）是用于表示控件具有接受输入的能力，只有当控件具有焦点时，控件才可以被激活，才可以响应事件。在活动的窗体中，任何时刻都只能有一个控件具有焦点，并且只有当控件的“ Enabled ”和“ Visible ”属性都为“ True ”时，才能获得焦点。当控件获得焦点之后，便会以特殊的外观显示出来。例如，当命令按钮、单选按钮或复选按钮获得焦点之后，便会在控件之上显示 1 个虚的方框；当文本框获得焦点之后，便会在文本框中出现 1 个闪动的光标。控件获得焦点最简单的方式便是直接单击该控件，除了这种方式之外，控件还可以通过按 Tab 键来获得焦点。连续按 Tab 键，焦点就会按控件添加的顺序在各个控件之间切换。

当控件获得焦点之后，便会激发 GotFocus 事件；反之，当控件失去焦点便会激发 LostFocus 事件。焦点事件的语法结构如下。

- GotFocus 事件

```
Private Sub 控件名_GotFocus()
```

```
End Sub
```

- LostFocus 事件

```
Private Sub 控件名_LostFocus()

End Sub
```

除了以上几种公共事件，另外还有一些常用的公共事件，详见附表二。

5.4 标签控件

标签控件主要是用来显示文本，如图 5-5 所示，但用户不能编辑所显示的文本，常用来说明或标示其他不具有“Caption”属性的控件，如文本框控件、列表框控件、组合框控件。所显示的文本是通过“Caption”属性来设置的，并且还可以选择字体的样式、大小和颜色。

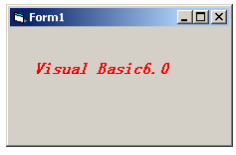


图5-5　文本框用来显示文本

【例5-4】　向窗体中添加 1 个标签控件，并在标签控件中以粗体显示“Visual Basic 6.0”，文字的颜色为红色。

程序编制步骤及相应代码如下。

1. 新建 1 个工程。
2. 在工具箱中双击标签控件A，向窗体中添加 1 个标签控件。
3. 在窗体上，单击标签控件，选中标签控件，这时标签控件周围会出现由蓝色小方框组成的边界线。
4. 将鼠标移动到蓝色的小方框上，鼠标变为双向的箭头，按住鼠标左键，然后拖动鼠标，这时标签控件的大小随着鼠标地拖动而发生改变，当鼠标拖到适当位置后，松开鼠标左键，这时标签控件的大小便发生相应地改变。
5. 将鼠标移动到标签控件上，按住鼠标左键，然后拖动鼠标，这时标签控件的位置随着鼠标地拖动而发生改变，当鼠标拖到适当位置后，松开鼠标左键，这时标签控件的位置便发生相应地改变。

在以后的例子中，如果没有特别说明，控件的大小及位置，都是通过以上方式来改变的，并且控件实际的大小和位置，用户可根据实际需要，自己选择。

6. 在【属性】窗口，单击【Caption】属性值栏，然后在该栏中输入“Visual Basic 6.0”。
7. 在【属性】窗口，单击【Font】属性栏，这时在属性栏右端会出现 1 个…按钮，单击该按钮打开【字体】选择窗口，如图 5-6 所示。
8. 在该窗口的【字体】栏选择字体的样式，在【字形】栏选择字形，在【大小】栏选择字体大小，按图 5-6 选择相关的值。
9. 单击【字体】选择窗口的 确定 按钮，回到主窗体。
10. 在【属性】窗口，单击【ForeColor】属性栏，这时在属性栏右端会出现 1 个向

下的箭头，单击该箭头打开【颜色】选择窗口。

11. 单击【颜色】选择窗口的【调色板】选项卡，将窗口切换到【调色板】窗口，然后在该窗口单击红色框。

在以后的例子中，控件的属性值直接给出，读者按上面介绍的方法设置。

12. 运行程序，窗体上便会显示如图 5-5 所示的文本。

除了“ Caption ”、“ Font ”、“ ForeColor ”等公共属性之外，标签控件还有一些其他常用的属性，主要包括：“ AutoSize ”属性、“ Alignment ”属性、“ BackStyle ”属性、“ BorderStyle ”属性等。

- “ AutoSize ”属性

功能：返回或设置标签是否自动改变大小以显示全部的内容。

说明：“ AutoSize ”属性有两个取值：“ True ”或“ False ”。“ AutoSize ”属性取“ True ”时，表示自动改变标签控件的大小以便显示全部的文本内容；“ AutoSize ”属性取“ False ”时（缺省值），表示不调整标签控件的大小，控件的大小保持不变，超出控件范围的内容将被剪掉。在【例 5-4】中，如果将标签控件的“ AutoSize ”属性设为“ True ”，则就没必要去调整标签控件的大小。

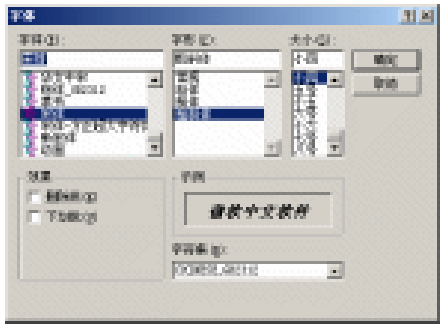


图5-6 【字体】选择窗口

- “ Alignment ”属性

功能：返回或设置标签控件中文本的对齐方式。

说明：“ Alignment ”属性有 3 个取值：0、1 或 2。“ Alignment ”属性取 0 时（默认值），表示标签控件中的文本左对齐显示；“ Alignment ”属性取 1 时，表示标签控件中的文本右对齐显示；“ Alignment ”属性取 2 时，表示标签控件中的文本居中显示。

- “ BackStyle ”属性

功能：返回或设置标签控件是否透明。

说明：“ BackStyle ”属性有两个取值：0 或 1。“ BackStyle ”属性取 0 时，表示标签控件透明，此时“ BackColor ”属性无效；“ BackStyle ”属性取 1 时（默认值），表示标签控件不透明，此时“ BackColor ”属性才有效。

- “ BorderStyle ”属性

功能：返回或设置标签控件的边框样式。

说明：“ BorderStyle ”属性有两个取值：0 或 1。“ BorderStyle ”属性为 0 时（默认值），表示标签控件无边框；“ BorderStyle ”属性为 1 时，表示标签控件有固定的单线边框。

由于标签控件主要是起标示作用的，因此在设计程序时，很少为其添加事件，但标签控件还是能够响应绝大多数的事件，如 Click 事件、Change 事件、MouseDown 事件、MouseUp、MouseMove 事件等，其中 Change 事件是在标签控件所显示的内容发生改变时才被激发。由于标签控件所显示的文本不能被编辑，因此标签控件是不能获得输入焦点的，也就不能响应键盘事件（KeyPress 事件、KeyDown 事件、KeyUp 事件）及焦点事件了（GotFocus 事件、LostFocus 事件）。

5.5 文本框控件

文本框控件是标准控件中最常用的控件之一，其在工具箱中的图标为  ，主要用于建立文本的输入或编辑区，以实现数据的输入、编辑、显示等。文本框控件实际上是一个文字编辑器，是显示和输入文本的重要工具之一。

5.5.1 文本框控件的常用属性

文本框控件除了有“Left”、“Top”、“Height”、“Width”及“Enabled”等公共属性之外，文本框还有一些特有的属性，主要包括：“MaxLenght”属性、“MultiLine”属性、“PasswordChar”属性、“ScrollBar”属性、“SelLength”属性、“SelStar”属性、“SelText”属性及“Text”属性等。

(1) “MaxLenght”属性

功能：返回或设置文本框控件中输入的最大字符量。

说明：“MaxLenght”属性值为整型数值，其默认值为 0，表示不限制输入的字符数，用户可以随意地输入字符。如果将“MaxLenght”属性值设为非 0 的整数，则用户所输入的字符数便有限制，不能超出所设定的值，超出的字符将被删除。

(2) “MultiLine”属性

功能：返回或设置文本框控件是否能够允许多行输入或显示。

说明：“MultiLine”属性有两个取值：“True”或“False”。取“True”时，表示允许多行输入或显示；取“False”时（默认值），表示不允许多行输入或显示，所有的字符都显示在一行中。

(3) “ScrollBar”属性

功能：返回或设置文本框控件是否有水平滚动条或垂直滚动条。

说明：“ScrollBar”属性只有在“MultiLine”属性为“True”时才有效，共有 4 个取值：0、1、2 或 3。“ScrollBar”属性为 0（默认值）时，表示不添加任何滚动条；取 1 时，表示添加水平滚动条；取 2 时，表示添加垂直滚动条；取 3 时，表示同时增加水平和垂直滚动条，如图 5-7 所示。

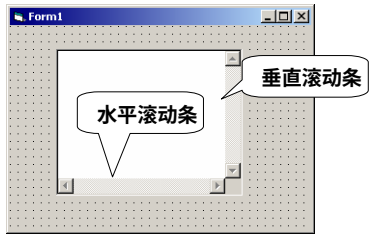


图5-7　添加滚动条后的文本框控件

(4) “PasswordChar”属性

功能：返回或设置替代符。

说明：设置该属性，所输入的字符将被所设置的符号所代替。例如，如果将“PasswordChar”属性设为“*”，则在文本框所输入的字符都将被符号“*”所代替，如图 5-8 所示。在实际应用中，该属性主要用来设置密码。

图5-8 设置“PasswordChar”为“*”的文本框

(5) “Text”属性

功能：返回或设置文本框控件中的文本。

说明：文本框控件无“Caption”属性，文本框中所显示的内容是通过“Text”属性来获取的。

(6) “SelLength”属性、“SelStar”属性、“SelText”属性

功能：这 3 个属性用于对文本内容进行选定等操作。其中“SelLength”属性返回或设置所选择的字符数；“SelStar”属性返回或设置选定文本的起始点，如果无选定的文本，则指出插入点的位置；“SelText”属性返回或设置当前被选定的字符，如果无选定字符，则返回空字符串。

说明：这 3 个属性不显示在【属性】窗口，要设置这 3 个属性必须用代码来完成，具体语法结构如下。

```
文本框控件名.SelLength＝长度值
文本框控件名.SelStar＝位置值
文本框控件名.SelText＝字符串
```

例如，要删除文本框控件 Text1 中的所有文本，可通过以下代码来完成。

```
Text1.SelStar＝0
Text1.SelLength＝Len(Text1.text)
Text1.SelText = ""
```

5.5.2 文本框控件的常用事件

文本框控件虽能响应鼠标事件、键盘事件、焦点事件等常用事件，但使用最多的还是 Change 事件，在窗体上双击文本框控件便可以为其添加 Change 事件。

【例5-5】 编写 1 个程序，用于验证用户所输入的密码是否正确。

程序编制步骤及相应代码如下。

1. 新建 1 个工程。
2. 向窗体中添加 2 个标签控件和 1 个文本框控件。
3. 调整各个控件的位置及大小至如图 5-9 所示。

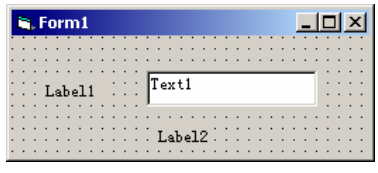


图5-9 调整后的窗体

4. 按表 5-3 设置有关控件的属性。

表 5-3　　控件属性

控件名	属性	属性值
Label1	AutoSize	True
	Caption	请输入密码：
Label2	AutoSize	True
	Alignment	2－Center
	Visible	False
Text1	MaxLength	4
	PasswordChar	*
	Text	删除 Text1

5. 在窗体上双击文本框控件，为文本框控件添加 Change 事件，并在事件中添加如下代码：

```
Private Sub Text1_Change()
    '当输入的密码达到 4 位，检验密码是否正确
    '并在标签控件上显示相应的提示
    If Len(Text1.Text) = 4 Then
        Label2.Visible = True
            If Text1.Text = "1234" Then
                Label2.Caption = "密码正确！"
            Else
                Label2.Caption = "密码错误，重新输入！"
            End If
    Else
        Label2.Visible = False
    End If
End Sub
```

6. 在【代码】窗口的对象列表框中选择“Text1”，然后在事件/过程列表框中选中“KeyPress”事件，为文本框控件添加 KeyPress 事件，并在事件中添加如下代码：

```
Private Sub Text1_KeyPress(KeyAscii As Integer)
    '限制只能在文本框中输入数字
    If Chr(KeyAscii) < "0" Or Chr(KeyAscii) > "9" Then
        Beep
        Text1.SelStart = 0
        Text1.SelLength = Len(Text1.Text)
        Text1.SelText = ""
    End If
End Sub
```

7. 运行程序，在文本框中输入密码，如果输入的密码为“1234”，则在标签控件中显示“密码正确!”；如果输入的密码不是“1234”，则在标签控件中显示“密码错误，重新输入!”。并且文本框中只能输入数字。

Change 事件是文本框最常用的事件，当文本框中的内容发生改变时，便会激发该事件，在【例 5-5】中，便是通过 Chang 事件来检验密码是否正确。KeyPress 事件常用来识别用户所键入的字符，当用户按下某个键时便会激发该事件，在【例 5-5】中，通过 KeyPress 事件来限制用户只能输入数字。GotFocus 事件、LostFocus 事件主要用来检查或确认用户的输入，当文本框获得焦点时，便会激发 GotFocus 事件，失去焦点时，便会激发 LostFocus 事件。

除了以上常用属性和常用事件之外，用户还可以通过 SetFocus 方法将焦点赋给文本框控件，语法结构如下：

文本框控件名.SetFocus

5.6 命令按钮控件

命令按钮控件是所有控件中最常用的控件之一，几乎所有的 Visual Basic 6.0 应用程序中都要用到命令按钮控件，常用于发布执行命令，其在工具箱中的图标为  。

5.6.1 命令按钮的常用属性

(1) “Caption”属性

功能：返回或设置命令按钮上所显示的文本。

说明：利用该属性还可以为命令按钮添加访问键，如果某个字母被定义成访问键时，用户便可以直接通过“Alt 键＋该字母键”来访问命令按钮。在设置“Caption”属性时，在要定义为访问键的字母前加上符号“&”，便可以将该字母设为访问键。例如，如果将命令按钮的“Caption”属性设为了“取消（&C）”，就可以通过按“Alt＋C 键”来直接访问命令按钮。

(2) “Style”属性

功能：返回或设置控件的外观样式。

说明：“Style”属性有两个取值，分别为 0 或 1。“Style”属性取 0 时（默认值），表示以标准样式显示命令按钮，按钮上不能显示图片；“Style”属性取 1 时，表示以图形样式显示命令按钮，此时可在命令按钮上显示图片。

(3) “Value”属性

功能：返回或设置按钮状态。

说明：“Value”属性有两个取值，分别为“True”或“False”。“Value”属性取“True”时，表示按钮被按下；取“False”时，表示按钮未被按下。“Value”属性不显示在【属性】窗口，要设置该属性只能通过代码来完成，语法结构如下：

控件名.Value＝True 或 False

5.6.2 命令按钮常用事件

单击事件（Click 事件）是命令按钮最常用的事件，在窗体上双击命令按钮便可以为命令按钮添加 Click 事件。程序运行时，在窗体上单击命令按钮，便会激发 Click 事件。

【例5-6】 设计 1 个简单的计算器，如图 5-10 所示。

程序编制步骤及相应代码如下：

1. 新建 1 个工程。
2. 向窗体中添加 5 个命令按钮控件和 3 个文本框控件，并调整控件的位置及大小至图 5-11 所示。

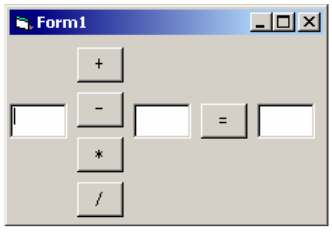


图5-10　简单计算器

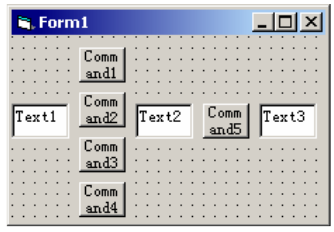


图5-11　调整后的窗体

3. 按表 5-4 设置相关控件的属性。

表 5-4　　控件属性

控件名	属性	属性值
Commond1	Caption	+
Commond1	Caption	−
Commond1	Caption	*
Commond1	Caption	/
Commond1	Caption	=
Text1	Text	删除 Text1
Text2	Text	删除 Text2
Text3	Text	删除 Text3

4. 在窗体上设计文本框 Text1，为文本框添加 Change 事件，并在程序的开始添加如下代码：

```
Dim num1 As Integer
Dim num2 As Integer
Dim result As Integer
```

5. 在文本框的 Change 事件中添加如下代码：

```
Private Sub Text1_Change()
    num1 = Val(Text1.Text)
End Sub
```

6. 单击窗体资源管理器的查看对象图标，回到窗体设计窗口。
7. 在窗体上双击命令按钮，为该命令按钮添加 Click 事件，并在事件中添加如下代码：

```
Private Sub Command1_Click()
    result = 1
End Sub
```

8. 以同样的方式为其他文本框控件添加 Change 事件，为命令按钮控件添加 Click 事件，并在相应的事件中添加响应的代码，程序最终代码如下（加底纹部分为新增代码）：

```
Dim num1 As Integer
```

```
Dim num2 As Integer
Dim result As Integer
Private Sub Command1_Click()
    result = 1
End Sub

Private Sub Command2_Click()
    result = 2
End Sub

Private Sub Command3_Click()
    result = 3
End Sub

Private Sub Command4_Click()
    result = 4
End Sub

Private Sub Command5_Click()
    Select Case result
         Case 1
            Text3.Text = Str(num1 + num2)
         Case 2
            Text3.Text = Str(num1 - num2)
         Case 3
            Text3.Text = Str(num1 * num2)
         Case 4
            Text3.Text = Str(num1 / num2)
     End Select
End Sub

Private Sub Text1_Change()
    num1 = Val(Text1.Text)
End Sub

Private Sub Text2_Change()
    num2 = Val(Text2.Text)
End Sub
```

9. **运行程序，在第 1 个文本框中输入第 1 个数，在第 2 个文本框中输入第 2 个数，在 4 个运算符号按钮中选择要进行的运算，然后单击 = 按钮，最终的计算**

结果便会显示在第 3 个文本框中。

命令按钮除了可以响应 Click 事件之外，还可以响应键盘、鼠标等其他公共事件，但命令按钮不支持鼠标双击事件（DblClick 事件）。

5.7 单选按钮控件、复选框控件及框架控件

单选按钮控件、复选框控件都是选择型按钮，主要是为了实现选择功能。如果要求用户进行单一选择时，一般使用单选按钮控件；如果允许用户进行多个选择时，一般选用复选框控件。框架控件是一个容器类控件，可以向框架中添加其他控件，一般的框架控件主要是用来将控件进行分组，最常见的情况便是将单选按钮进行分组。

5.7.1 单选按钮控件

单选按钮常成组出现，在一组单选按钮中，用户只能选中其中的一个单选按钮，如图 5-12 所示。单击某个单选按钮，则该单选按钮被选中；单击其他单选按钮，则该按钮不被选中。在工具箱中单选按钮控件的图标为。

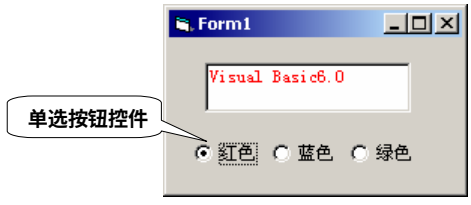


图5-12　单选框

单选按钮控件上所显示的文本是由“ Caption ”属性来设置的，所处的状态是由“ Value ”属性来获得的。“ Value ”属性主要是用来设置或返回单选按钮控件的状态，其有两个取值：“ True ”或“ False ”，“ Value ”属性取“ True ”时，表示单选按钮控件被选中；“ Value ”属性取“ False ”时，表示单选按钮控件未被选中。“ Value ”属性是不显示在【属性】窗口的，用户只能通过代码来设置该属性，语法结构如下：

单选按钮控件名.Value＝True 或 False

【例5-7】　编写 1 个用单选按钮控件来选择字体的应用程序。

程序编制步骤及相应代码如下。

1. 新建 1 个工程。
2. 向窗体中添加 3 个单选按钮控件和 1 个文本框控件。
3. 调整控件的位置及大小至适当位置，如图 5-13 所示。

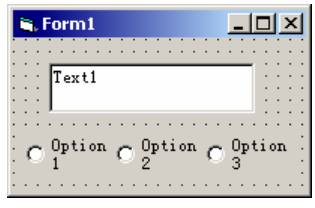


图5-13　调整后的窗体

4. 将 3 个单选按钮的“ Caption ”属性依次设为“ 红色 ”、“ 蓝色 ”、“ 绿色 ”，并删除文本框“ Text ”属性中的“ Text1 ”。
5. 双击窗体的空白部分，为窗体添加 Load 事件，并在事件中添加如下代码：

```
Private Sub Form_Load()
    '选中“红色”单选按钮
    Option1.Value = True
End Sub
```

6. 单击窗体资源管理器的查看对象图标，回到窗体设计窗口。
7. 在窗体双击“红色”【单选】按钮，为其添加 Click 事件，并在事件中添加如下代码：

```
Private Sub Option1_Click()
    '将文本框的文字变为红色
    Text1.ForeColor = vbRed
End Sub
```

8. 以同样的方式为其他【单选】按钮添加 Click 事件，并在事件中添加代码，最终的代码如下（加底纹部分为新增代码）：

```
Private Sub Form_Load()
    '选中“红色”单选按钮
    Option1.Value = True
End Sub

Private Sub Option1_Click()
    '将文本框的文字变为红色
    Text1.ForeColor = vbRed
End Sub

Private Sub Option2_Click()
    '将文本框的文字变为蓝色
     Text1.ForeColor = vbBlue
End Sub

Private Sub Option3_Click()
    '将文本框的文字变为绿色
     Text1.ForeColor = vbGreen
End Sub
```

9. 运行程序，“红色”单选按钮被选中，在文本框中输入“Visual Basic 6.0”，字体颜色为红色；单击“蓝色”单选按钮，选中该单选项，这时文本框中的文字为蓝色；单击“绿色”单选按钮，选中该单选项，这时文本框中的文字为绿色。

单选按钮控件虽可以响应绝大多数的事件，但单击事件（Click 事件）是最常用的事件。除了单击单选按钮可以激发 Click 事件之外，将单选按钮控件的“Value”属性设为“True”也同样可以激发 Click 事件。【例 5-7】便是通过单选按钮的 Click 事件来实现对字体颜色的选择。

5.7.2　复选框控件

复选框控件相当于一个开关，用于表明某个特定状态是否被选中。单击复选框控件，则该复选框控件被选中☑；再次单击该复选框控件，则该复选框不被选中☐。复选框控件成组出现时，用户可以同时选中一个或多个选项，如图 5-14 所示。在工具箱中复选框的图标为☑。

图5-14　复选框

和单选按钮控件一样，复选框控件上所显示的文本由“ Caption ”属性来设定，复选框控件的状态由“ Value ”属性来返回或设置，“ Value ”属性有 3 个取值：0、1 或 2。取 0 时，表示复选框控件没有被选中；取 1 时，表示复选框控件被选中；取 2 时，表示复选框控件不可用，此时复选框以灰色显示。

【例5-8】　编写一个用复选框控件来选择字体样式的应用程序。

程序编制步骤及相应代码如下。

1. 新建 1 个工程。
2. 向窗体中添加 3 个复选框控件和 1 个文本框控件，并调整控件的位置及大小至适当位置，如图 5-15 所示。

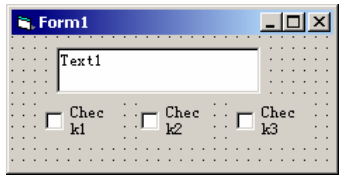


图5-15　调整后的窗体

3. 将 3 个复选框控件的“ Caption ”属性依次设为“ 粗体 ”、“ 斜体 ”、“ 下划线 ”，并将文本框的“ Text ”属性设为“ ”。
4. 双击窗体的空白部分，为窗体添加 Load 事件，并在事件中添加如下代码：

```
Private Sub Form_Load()
    '选中“粗体”复选框
    Check1.Value = 1
End Sub
```

为窗体添加 Load 事件时，不要在有控件的位置或窗体的标题上双击窗体。

5. 单击窗体资源管理器的查看对象标签，回到窗体设计窗口。
6. 在窗体双击 红色【单选】按钮，为其添加 Click 事件，并在事件中添加如下代码：

```
Private Sub Check1_Click()
```

```
    '将文本框中文字加粗或不加粗
    Text1.FontBold = Check1.Value
End Sub
```

7. 以同样的方式为其他【单选】按钮添加 Click 事件，并在事件中添加代码，最终的代码如下（加底纹部分为新增代码）：

```
Private Sub Check1_Click()
    '将文本框中文字加粗或不加粗
    Text1.FontBold = Check1.Value
End Sub

Private Sub Check2_Click()
    '将文本框中文字变为斜体或不变为斜体
    Text1.FontItalic = Check2.Value
End Sub

Private Sub Check3_Click()
    '为文本框中文字加上或去掉下划线
    Text1.FontUnderline = Check3.Value
End Sub

Private Sub Form_Load()
    '选中“粗体”复选框
    Check1.Value = 1
End Sub
```

8. 运行程序，粗体复选框被选中☑ 粗体，在文本框中输入“Visual Basic 6.0”，字体为粗体；单击复选框☐ 斜体，选中该复选框，这时文本框中文字变为斜体，再次单击复选框☐ 斜体，该复选框不被选中，此时文本框中文字又恢复为标准的字体；以同样的方式选中复选框☐ 下划线，看看有什么效果。

复选框被选中或其“Value”值为 1 时，便会激发复选框的 Click 事件。【例 5-8】便是通过复选框的 Click 事件来实现字体样式的选择的。

5.7.3 框架控件

在窗体上添加多个单选按钮时，所有的单选按钮都将被作为 1 组，因此用户只能选中多个单选按钮中的 1 个。如果想选中多个单选按钮，则必须使用框架控件将单选按钮分组。单选按钮被分组后，便可以在每组中都选中 1 个单选按钮，如图 5-16 所示。在工具箱中，框架的图标为。

框架作为一种容器类控件，可为其他控件提供可标识的分组，所标识的文本由“Caption”属性来设置。为了让框架起到分组的作用，在添加控件时，必须将控件添加到框架中去，具体添加的方法如下：

- 向窗体中添加 1 个框架控件，然后将框架控件调整到适当的大小。

- 在工具箱中单击要添加的控件图标，然后将鼠标移到框架上，按住鼠标左键，拖动鼠标至适当位置，松开鼠标左键。注意：鼠标拖动的范围不能超出框架的边界。

向框架中添加控件后，所添加的控件将被作为框架的一部分，随着框架的改变而改变，随着框架的消失而消失。

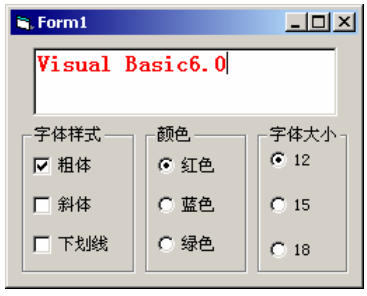


图5-16　用框架来分组

【例5-9】　编写一个能选择字体样式、字体颜色和字体大小的应用程序。

程序编制步骤及相应代码如下。

1. 新建 1 个工程。
2. 向窗体中添加 3 个框架控件和 1 个文本框控件，并调整控件大小及位置至适当位置，如图 5-17 所示。

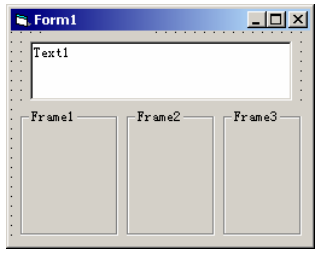


图5-17　调整后的窗体

3. 将 3 个框架控件的“ Caption ”属性对应地设为“ 字体样式 ”、“ 颜色 ”、“ 大小 ”，并删除文本框“ Text ”属性中的“ Text1 ”。
4. 在工具箱中单击复选框控件的图标，然后将鼠标移到框架【字体样式】上。
5. 按住鼠标左键，在框架上拖动鼠标，在适当位置松开鼠标，向框架【字体样式】中添加 1 个复选框控件，如图 5-18 所示。

图5-18　向框架中添加复选框

6. 以同样的方式再向【字体样式】框架中添加 2 个复选框控件，向【颜色】框架中添加 3 个单选按钮，向【字体大小】框架中添加 3 个单选按钮，并调整控件大小及位置至如图 5-19 所示。

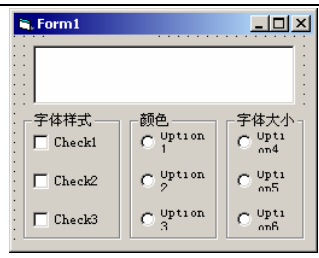


图5-19　最终的窗体

7. 按表 5-5 设置控件的“ Caption ”属性。

表 5-5　　控件的“ Caption ”属性

控件	Caption 属性值
Check1	粗体
Check2	斜体
Check3	下划线
Option1	红色
Option2	蓝色
Option3	绿色
Option4	12
Option5	15
Option6	18

8. 在窗体无控件的部位双击窗体，为窗体添加 Load 事件，并打开【代码】窗口，在【代码】窗口添加如下代码：（这里省略了事件的添加过程，读者可按相应的方法为各个控件添加相关的事件。）

```
Private Sub Check1_Click()
    Text1.FontBold = Check1.Value
End Sub

Private Sub Check2_Click()
    Text1.FontItalic = Check2.Value
End Sub

Private Sub Check3_Click()
    Text1.FontUnderline = Check3.Value
End Sub

Private Sub Form_Load()
    '选中“粗体”复选框
    Check1.Value = 1
    '选中“红色”单项按钮
    Option1.Value = True
```

```
    '选中“12”单项按钮
    Option4.Value = True
End Sub

Private Sub Option1_Click()
    Text1.ForeColor = vbRed
End Sub

Private Sub Option2_Click()
    Text1.ForeColor = vbBlue
End Sub

Private Sub Option3_Click()
    Text1.ForeColor = vbGreen
End Sub

Private Sub Option4_Click()
    '将文本框中的字体大小变为 12 号
    Text1.FontSize = 12
End Sub

Private Sub Option5_Click()
    '将文本框中的字体大小变为 15 号
    Text1.FontSize = 15
End Sub

Private Sub Option6_Click()
    '将文本框中的字体大小变为 18 号
    Text1.FontSize = 18
End Sub
```

9. **运行程序，【粗体】复选框、【红色】单选按钮、【12】单选按钮被选中，在文本框中输入“Visual Basic 6.0”，然后在各个框架中选择字体的样式、颜色及大小。**

5.8 列表框控件、组合框控件

列表框控件和组合框控件都是列表选择型控件，主要是用来向用户提供一系列的列表项目，用户可以从这些列表项目中选择自己所需的项目。

列表框控件以列表的形式向用户提供一系列项目，如图 5-20 所示，用户可以从中选择一个或多个项目。用户在列表框中单击某一项目时，该项目会以蓝色光条的形式显示，表示该项目被选中。当列表框中的项目超出了列表框所能显示的范围，便会自动在列表框中增加 1 个垂

直滚动条，便于用户进行上下翻动，如图 5-20 所示。在工具箱中，列表框控件的图标为。

组合框控件以下拉列表或组合列表的形式向用户提供一系列项目，如图 5-20 所示，它兼有列表框控件和文本框控件的功能。用户可以在列表框部分选择所需的项目，也可以在文本框部分输入所需的项目。在工具箱中，组合框控件的图标为。

图5-20　列表框、组合框控件

5.8.1　列表框控件

列表框的常用属性如下。

(1)　“List”属性

功能：返回或设置列表框中某一列表项的内容。

说明：“List”属性是 1 个字符串类型的数组，列表框中所有的列表项都被保存在该数组中，因此要访问或设置列表框中的某一项时，必须按以下语法结构来访问：

```
列表框控件名.List(列表项的索引值)[=字符串表达式]
```

例如，可以通过以下语句来设置列表框 List1 中的第一项：

```
List1.List(0)="武汉"
```

(2)　“ListCount”属性

功能：返回列表框控件中所有项目的个数。

说明：“ListCount”属性是不显示在【属性】窗口的，只能通过代码来访问该属性，具体语法结构如下：

```
[整型变量＝] 列表框控件名.ListCount
```

(3)　“ListIndex”属性

功能：返回或设置当前被选中的列表项的索引值。

说明：“ListIndex”属性是不显示在【属性】窗口的，只能通过代码来访问或设置该属性，具体语法结构如下：

```
列表控件名.ListIndex[＝索引值]
```

例如，以下代码可以选中列表框 List1 中的第 2 项。

```
List1.ListIndex＝1
```

(4)　“Sorted”属性

功能：返回或设置列表框控件的列表是否按字母升序来排列。

说明：“Sorted”属性有两个取值：“True”或“False”。取“True”时，表示按字母升序排列列表项；取“False”时（默认值），表示按列表加入的顺序排列列表。

(5)　“Text”属性

功能：返回列表框控件中最后被选中的列表项。

说明：“Text”属性是不显示在【属性】窗口的，只能通过代码来访问该属性，具体语法

结构如下：

```
[字符串变量]=列表框控件名.Text
```

(6) “MultiSelect”属性

功能：设置是否可以在列表框控件中选择多个列表项。

说明：“MultiSelect”属性有 3 个取值，分别是 0、1 或 2。取 0 时，表示只能选择一项；取 1 时，表示允许用户进行多项选择，在进行多项选择时，用户只需单击所要选择的各个项即可，如果某项已被选中，再单击该项时，该项将被取消；取 2 时，也表示允许用户进行多项选择，但用户在进行多项选择时，必须同时按住 Shift 键，同样如果某项已被选中，再单击该项时，则该项被取消。

下面介绍列表的添加与删除。

列表框中的列表并不是自动添加上去的，需要用户用特定的方法来添加。要想删除列表框中的某一列表，同样也需要使用特定的方法来完成。

【例5-10】 向列表框添加列表，并删除选定的列表。

程序编制步骤及相应代码如下。

1. 新建 1 个工程。
2. 向窗体中添加 1 个标签控件、1 个文本框控件、1 个列表框控件和 2 个命令按钮控件，并调整控件的位置及大小至图 5-21 所示。

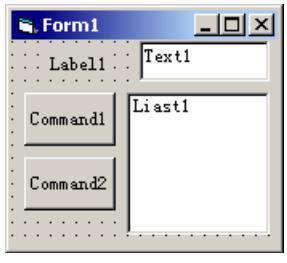


图5-21　调整后的窗体

3. 在窗体上单击列表框控件选中该控件，然后在【属性】窗口单击【List】属性栏，这时在该栏的右端会出现 1 个向下的箭头，单击该箭头，打开下拉列表，删除“List1”列表。
4. 按表 5-6 设置有关控件的属性。

表 5-6　　控件属性

控件	属性	属性值
Label1	Caption	输入学生姓名：
Command1	Caption	添加
Command2	Caption	删除
Text1	Text	删除 Text1

5. 在窗体上分别单击两个命令按钮控件及列表框控件，为相应的控件添加 Click 事件，并在相应的事件中添加如下代码：

```
Dim i As Integer

Private Sub Command1_Click()
```

```
    '增加列表项
    List1.AddItem Text1.Text
End Sub

Private Sub Command2_Click()
    '删除选定的列表项
    If i >= 0 Then
        List1.RemoveItem i
    End If
End Sub

Private Sub List1_Click()
    '选定列表项
    i = List1.ListIndex
End Sub
```

6. 运行程序，在文本框中输入姓名，然后单击 添加 按钮，便可以将文本框中所输入的姓名添加到列表框中；在列表框中单击某个列表项选中该列表项，然后单击 删除 按钮，便可以将选中的列表项删除。

向列表框中增加列表可以使用 AddItem 方法，具体的语法结构如下：

```
列表框控件名.AddItem 字符串变量或表达式,[索引值]
```

如果省略“索引值”，则列表项总是添加到列表框最后；如果指定“索引值”，则在所指定的位置添加列表项，并将该位置以后的列表项都向后移动 1 个位置。另外，如果将列表框控件的“Sorted”属性设为“True”，则“索引值”失去作用。【例 5-10】通过使用 AddItem 方法向列表框添加列表项：

```
List1.AddItem  Text1.text
```

所添加的列表项总是添加到列表框的最后。如果将该行代码改为如下形式看看会有什么效果，读者不妨试一试：

```
List1.AddItem  Text1.text,0
```

如果想删除列表框中的列表项，则可以使用 RemoveItem 方法，具体语法结构如下：

```
列表框控件名.RemoveItem  列表项索引值
```

【例 5-10】便是通过 RemoveItem 方法来删除选定的列表项：

```
List1.RemoveItem i
```

其中变量“i”便是用来指定列表项的索引值。如果想删除列表框中的所有列表项，可以使用 Clear 方法，具体语法结构如下：

```
列表控件名.Clear
```

在列表框中单击某个列表项，便可以选定该列表项，该列表项的索引值被“ListIndex”属性所记录，列表项的内容被“Text”属性所保存。在单击列表项的同时，便会激发列表框控件的 Click 事件。【例 5-10】便是通过列表框的 Click 事件来获取选中列表项的索引值的。

5.8.2　组合框控件

组合框控件的常用属性：

(1)　“ Style ”属性

功能：返回或设置组合框的样式。

说明：“ Style ”属性可以有以下 3 种取值，分别为 0、1 或 2。取 0 时，表示组合框的样式为组合下拉式，如图 5-22 所示，用户通常是看不到所有列表项的，只有通过单击右端的箭头才可以看到全部的列表项，在这种样式下用户即可以在文本框部分输入列表项，也可以在下拉列表框部分选择列表项；取 1 时，表示组合框的样式为组合式，如图 5-23 所示，用户即可以在文本框中输入列表项，也可以在列表框中选择列表项，要想看到列表框，在设计组合框时，必须将组合框的高度尽量拉长；取 2 时，表示组合框的样式为简单下拉式，如图 5-24 所示，在样式上和第 1 种样式没什么区别，但在此种样式下，用户不能在文本框中输入列表项。

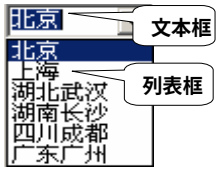


图5-22　组合下拉式组合框

图5-23　组合式组合框

图5-24　简单下拉式组合框

(2)　“ Text ”属性

功能：返回或设置组合框被选中的列表项。

说明：如果列表项是在文本框中直接输入的，则“ Text **”属性返回的是在文本框中所输入的列表项；如果列表项是从列表框中选择的，则“** Text **”属性返回的是在列表框中所选定的列表项。**

除了“ Text **”属性有所不同之外，组合框的其他属性基本上和列表框一样，如“** List **”属性、“** ListCount **”属性、“** ListIndex **”属性等，这里不再详细论述，读者可参看列表框有关属性的说明。另外，和列表框控件一样，也可以使用** AddItem **方法为组合框添加列表项，用** RemoveItem **方法来删除选定的列表项，用** Clear **方法来删除全部列表项，具体语法结构和列表框的完全一样。**

下面介绍组合框控件的常用事件。

组合框所能响应的事件与“ Style **”属性有关，但无论在哪种样式下，用户只要单击组合框中的列表项，便会激发** Click **事件。**

【例5-11**】 编写一个在文本框中显示从组合框中所选择的内容。**

程序编制步骤及相应代码如下。

1. **新建** 1 **个工程。**
2. **向窗体中添加** 1 **个组合框和** 1 **个文本框，并调整控件的位置及大小如图** 5-25 **所示。**

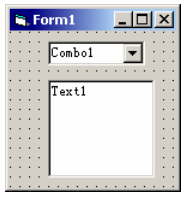


图5-25　调整后的窗体

3. 删除文本框的“Text”属性中的“Text1”，并将“MultiLine”属性设为“True”。
4. 在窗体的空白处双击窗体，为窗体添加 Click 事件，并在事件中添加如下代码：

```
Private Sub Form_Load()
    '向组合框中添加列表项
    With Combo1
        .AddItem "北京"
        .AddItem "上海"
        .AddItem "湖北武汉"
        .AddItem "湖南长沙"
        .AddItem "四川成都"
        .AddItem "广东广州"
        .ListIndex = 0
    End With
End Sub
```

5. 在【代码】窗口的对象列表框中选择“Combo1”，在事件列表框中选择“Click”，为组合框添加 Click 事件，并在事件中添加如下代码：

```
Private Sub Combo1_Click()
    '在文本框中分行显示所选中的列表项
    Text1.Text = Text1.Text + Combo1.Text + Chr(13) + Chr(10)
End Sub
```

6. 运行程序，组合框中的列表项“北京”被选中，同时也被显示在文本框中。在组合框中单击列表项选中该列表项，所选中的列表项同时也被显示在文本框中。

除了可以响应 Click 事件之外，组合框还能响应其他一些常用事件，但与“Style”属性有关。当“Style”属性为 0 或 1 时，如果直接在文本框中输入列表项或通过代码设置了“Text”属性，则会激发 Change 事件；而当“Style”属性为 2 时，则不能响应 Change 事件；当“Style”属性为 1 时，如果在列表框中双击列表项，则会激发双击事件（DblClick 事件），而在另外两种样式下，组合框不能响应 DblClick 事件；当“Style”属性为 0 或 2 时，如果单击下拉箭头，则会激发 Dropdown 事件，而“Style”属性为 1 时，不能响应该事件。

5.9 滚动条控件

滚动条一般附在窗口的边上，用来滚动窗口，以方便查看数据。在 Visual Basic 6.0 中，滚动条还常常用来进行数据的输入。Visual Basic 6.0 为用户提供了两种样式的滚动条，一种为水平的，一种为垂直的，如图 5-26 所示。单击滚动条两端的箭头或滚动条的空白处，滚动框便会在滚动条上移动。这两种控件除了在方向上有所不同之外，属性和所能响应的事件都一样，在工具箱中所对应的图标分别为  （水平）和 （垂直）。

5.9.1 滚动条的常用属性

(1) “Max”属性、“Min”属性

功能：返回或设置滚动条所能表示的范围。其中“Max”属性用于设置最大值，“Min”属性用于设置最小值。

说明：“ Max ”属性表示的是当滚动框处于最右端或最下端时，滚动条所对应的值；“ Min ”属性表示的是当滚动框处于最左端或最上端时，滚动条所对应的值，并且一般的“ Min ”属性值不能大于“ Max ”属性值。

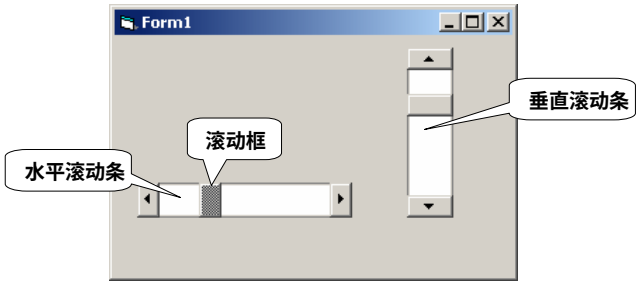


图5-26　滚动条

(2)　“ SmallChange ”属性、“ LargeChange ”属性

功能：返回或设置滚动框每次滚动的范围，即滚动框每次移动的距离。

说明：“ SmallChange ”属性用于设置滚动框每次移动的最小距离；“ LargeChange ”属性用于设置滚动框每次移动的最大距离。当单击滚动条两端的箭头时，滚动框便按“ SmallChange ”属性所设定的值滚动；当单击滚动条的空白处时，滚动框便按“ LargeChange ”属性所设定的值滚动。

(3)　“ Value ”属性

功能：返回或设置滚动条所代表的值。

说明：当滚动框处于不同的位置时，滚动条所代表的值也不一样，具体所代表的值由“ Value ”属性返回。当滚动框处于最左端或最上端时，“ Value ”的值为“ Min ”属性所设定的最小值；当滚动框处于最右端或最下端时，“ Value ”的值为“ Max ”属性所设定的最大值；当滚动框处于其他位置时，“ Value ”的值介于最大值和最小值之间。当单击滚动条两端的箭头时，“ Value ”值便按“ SmallChange ”属性所设定的最小改变量递增或递减；当单击滚动条的空白处时，“ Value ”值便按“ LargeChange ”属性所设定的最大改变量递增或递减。

5.9.2　滚动条的常用事件

滚动条能响应的常用事件有滚动事件（Scroll 事件）和改变事件（Change 事件）。当在滚动条上拖动滚动框时，便会激发滚动事件（Scroll 事件），而滚动框的位置发生改变后便会激发改变事件（Change 事件）。因此，可以用滚动事件（Scroll 事件）来跟踪滚动框的动态变化，改变事件（Change 事件）可用来得到滚动框的位置。

【例5-12】 设计一个简单的调色板。

程序编制步骤及相应代码如下。

1. 新建 1 个工程。
2. 向窗体中添加 3 个水平滚动条控件和 4 个标签控件，并调整控件的位置及大小如图 5-27 所示。
3. 按表 5-7 设置有关控件的属性。

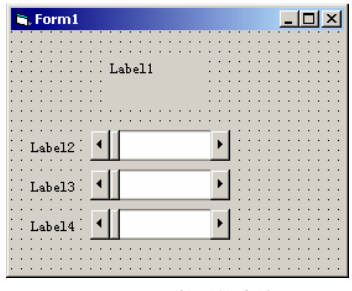


图5-27　调整后的窗体

表 5-7　　控件属性

控件	属性	属性值	说明	控件	属性	属性值	说明
Label1	Caption	“”	显示	HScroll1	Max	255	用于调
	AutoSize	True	颜色		Min	0	整绿色
HScroll1	Max	255	用于调		LargeChange	20	所占的
	Min	0	整红色		SmallChange	15	比列
	LargeChange	20	所占的	Label1	Caption	红色	
	SmallChange	15	比列		ForeColor	&H000000FF&（即红色）	
HScroll2	Max	255	用于调	Label2	Caption	蓝色	
	Min	0	整蓝色		ForeColor	&H00FF0000&（即蓝色）	
	LargeChange	20	所占的	Label3	Caption	绿色	
	SmallChange	15	比列		ForeColor	&H0000FF00&（即红色）	

4. **在窗体上分别双击 3 个滚动条控件，为滚动条添加 Change 事件，并在事件中添加如下代码：**

```
Private Sub HScroll1_Change()
    Label1.BackColor = RGB(HScroll1.Value, _
    HScroll2.Value, HScroll3.Value)
End Sub

Private Sub HScroll2_Change()
    Label1.BackColor = RGB(HScroll1.Value, _
    HScroll2.Value, _HScroll3.Value)
End Sub

Private Sub HScroll3_Change()
    Label1.BackColor = RGB(HScroll1.Value, _
    HScroll2.Value, HScroll3.Value)
End Sub
```

5. **运行程序，单击滚动条两端的箭头或滚动条的空白处，标签控件 1 的颜色也随着滚动条的改变而改变。**

5.10　定时器控件

定时器控件是专门用于计时的控件，它的大小是不可以改变的，并且在程序运行时是不可见的。定时器控件可以周期性地完成某项工作，利用这一点可以用定时器来设计简单的动画。在工具箱中，定时器控件的图标为。

定时器控件的属性只有 7 种，其中最为重要的属性为“ Interval ”（事件间隔）属性。“ Interval ”属性以毫秒（即千分之一秒）为单位，它决定着每隔多长时间激发一次计时事

件。当“ Interval ”属性所设定的时间一旦达到，系统便会自动激发定时器的惟一事件 Timer 事件。只要将所要做的事件放在定时器的 Timer 事件中，便可以实现定时做某件事的功能。

【例5-13】 设计 1 个倒计时器。

程序编制步骤及相应代码如下。

1. 新建一个工程。
2. 向窗体上添加 1 个定时器控件、1 个标签控件和 1 个命令按钮，并调整控件的大小及位置至图 5-28 所示。

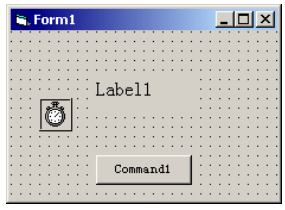


图5-28　调整后的窗体

3. 将标签控件的“ AutoSize ”属性设为“ True ”，命令按钮的“ Caption ”属性设为“ 开始 ”。
4. 在窗体上双击定时器控件，为定时器控件添加 Timer 事件，并打开【代码】窗口，在【代码】窗口添加如下代码：

```
Dim i As Integer

Private Sub Command1_Click()
    '根据命令按钮所显示的文本的不同而执行不同的操作
    Select Case Command1.Caption
        '当按钮为“开始”，便开始倒计时，并将按钮切换到“暂停”
        Case "开始"
            Command1.Caption = "暂停"
            Timer1.Interval = 100
        '当按钮为“暂停”，便暂停倒计时，并将按钮切换到“开始”
        Case "暂停"
            Command1.Caption = "开始"
            Timer1.Interval = 0
    End Select
End Sub

Private Sub Form_Load()
    i = 10
    Label1.FontSize = 30
    Label1.FontBold = True
    Label1.Caption = i
End Sub
Private Sub Timer1_Timer()
```

```
        '循环倒计时
        If i = 0 Then
            i = 10
        End If
        i = i - 1
        Label1.Caption = i
        '改变文字的颜色
        Label1.ForeColor = Rnd * RGB(255, 0, 0)
    End Sub
```

5. 运行程序，单击 

按钮，开始倒计时，并且按钮变为【暂停】按钮。单击 

按钮，便暂停倒计时，并且按钮变为【开始】按钮。

如果定时器的“Interval”属性的值大于 0，则按所设定的时间定时地做某件事；如果定时器的“Interval”属性的值等于 0，则定时器不起作用，因此也就不能激发定时器的 Timer 事件了。【例 5-13】便是通过设置定时器的“Interval”属性来实现倒计时及暂停的效果。

5.11 控件命名的约定

在前面编程的过程中，所有控件的名称都是采用系统默认的名称，如 Command1、Label1 等，这种命名的方法既不利于书写，而且也很不直观，相互之间极易混淆。为了便于编写程序，Visual Basic 6.0 对控件的命名有个约定，即控件名要具有可读性，也就是说，一看到控件的名称便知道该控件的类型及其功能。为了表明控件的类型，在给控件命名时，必须在名称前面加上控件类型的缩写前缀，如“cmd”代表着命令按钮，“lbl”代表着标签控件等，常用控件名称的缩写如表 5-8 所示。

表 5-8　　控件命名约定缩写

控件	名称缩写	示例
窗体	frm	frmDraw
标签控件	lbl	lblName
文本框控件	txt	txtName
命令按钮控件	cmd	cmdOK
单选按钮控件	opt	optMan
框架控件	fra	fraColor
复选框控件	chk	chkFont
列表框控件	lst	lstCity
组合框控件	cbo	cboCity
水平滚动条	hsb	hsbRed
垂直滚动条	vsb	vsbRed
图片框控件	pic	picCat
图像框控件	img	picCat
菜单	mnu	mnuFile

5.12　小结

本章是 Visual Basic 6.0 最基础的一章，本章主要介绍了 Visual Basic 6.0 的各种标准控件，具体学习了以下内容：

- Visual Basic 6.0 标准控件的添加。
- 各种标准控件的常用属性及其功能。
- 各种标准控件的常用事件及其激发的条件。
- 各种标准控件的常用方法。

5.13　习题

一、 填空题

1. 控制控件是否可见的属性为_____；控制控件是否可用的属性为_____；控件的位置是由_____和_____属性来确定的；控件的大小是由_____和_____属性来确定；控件上所显示的文本是由_____属性来设定的；控件的外观样式是由_____来设定，该属性有_____和_____两个取值。
2. 与鼠标有关的事件包括_____、_____、_____、_____、_____，其中_____在单击控件时被激发，_____在鼠标键被按下时被激发，_____在鼠标被松开时被激发，_____事件在双击控件时被激发。
3. 与键盘有关的事件包括_____、_____、_____，其中_____在单击键盘键时被激发，_____在按下键盘键时被激发，_____在松开键盘键时被激发。
4. 要想标签能根据所显示的内容来自动调整大小，则必须将“ AutoSize ”属性设为_____。
5. 要想在文本框中输入多行内容，则必须将“ MultiLine ”属性设为_____。当文本框中的内容发生改变时，便会激发_____事件，文本框中所输入的内容由_____属性返回。
6. 进行多选一时，可使用_____控件，进行多选多时，可使用_____控件。
7. 向列表框和组合框中添加列表时，可使用_____方法；删除选定的列表，可使用_____方法；删除全部列表项，可使用_____方法。
8. 滚动条所能代表的范围是由_____和_____属性来确定的，滚动条当前所代表的值由_____属性返回。当单击滚动条两端的箭头时，滚动条的增量值是由_____属性决定的，当单击滚动条的空白处时，滚动条的增量值是由_____属性决定的。
9. 定时器控件能够响应的惟一事件为_____，并且该事件被激发的时间间隔由_____属性来给定。

二、 选择题

1. 以下事件中，标签控件不能响应的事件为（ ）。

   A. Click 事件　　B. MouseDown 事件　　C. MouseUp 事件　　D. GotFocus 事件

2. 以下事件中，命令按钮不能响应的事件为（ ）。

   A. Click 事件　　B. MouseDown 事件　　C. Change 事件　　D. GotFocus 事件

3. 使用单选按钮控件或复选框控件来实现选择功能时，选择项是否被选中可由（ ）属性来获得。

A. Enabled　　B. Visible　　C. Value　　D. Caption

4. **要使组合框的样式为简单组合式，则“** Style **”属性设为（　）。**

A. 0　　B. 1　　C. 2　　D. 3

5. **如果只允许在列表框中选择一个列表项，则“** MultiSelect **”属性必须设为（　）。**

A. 0　　B. 1　　C. 2　　D. 3

6. **在列表框、组合框中，当前被选中的列表项由（　）返回。**

A. List　　B. ListIndex　　C. Text　　D. ListCount

7. **在列表框、组合框中，如果“** Sorted **”属性为“** True **”，则以下哪个属性失效（　）。**

A. ListCount　　B. ListIndex　　C. Text　　D. List

8. **当单击组合框的下拉箭头时，便会激发（　）事件。**

A. Click　　B. Change　　C. DropDown　　D. DblClick

9. **组合框所能响应的事件与下面哪个属性有关（　）。**

A. List　　B. ListIndex　　C. Text　　D. Style

10. **当拖动滚动条的滚动框时，便会激发（　）事件**

A. Scroll　　B. Change　　C. DropDown　　D. Click

三、 程序设计题

1. **设计** 1 **个能将列表框中所选定的列表项添加到组合框中去的程序。要求：窗体上要有** 1 **个列表框、**1 **个组合框、**1 **个【确定】按钮和** 1 **个【取消】按钮。列表框中的列表项先添加，然后在列表框中选定列表项，单击【确定】按钮，列表框中被选定的列表项便会被添加到组合框中，并且列表框中被选定的列表项被删除；单击【取消】按钮，组合框中被选定的列表项重新回到列表框中。**

2. **设计** 1 **个统计学生爱好的程序，界面如图** 5-29 **所示。要求：单击【确定】按钮，所选择的内容将按顺序显示在右边的文本框中，并且该文本框自带滚动条，可以多行显示。**

图5-29　**程序界面**

第6章　菜单栏、工具栏、状态栏的设计

前面几章涉及到了设计简单的界面，但设计出来的界面极其单调，为此还必须向窗体添加能够美化界面的菜单栏、工具栏和状态栏。菜单栏、工具栏不仅可以美化、简化用户界面，而且还可以使得应用程序的功能加强；状态栏主要是用来显示工作状态或一些系统信息。Visual Basic 6.0 提供了专门的工具来设计菜单栏、工具栏和状态栏，使设计工作变得简单容易。

本章学习目标

- 菜单基本知识。
- 菜单编辑器的使用。
- 菜单栏的设计。
- 弹出式菜单的设计。
- 工具栏的设计。
- 状态栏的设计。

6.1 菜单基本知识

在 Visual Basic 6.0 中，一个完整的菜单项一般由菜单标题、访问键、快捷键 3 项组成，如图 6-1 所示，菜单标题就好像是每个人的名字，访问键、快捷键就好像是这个人的别名。每个人都必须有一个名字，但不一定有别名，因此对于菜单项而言，菜单标题是必须的，访问键、快捷键可有可无。

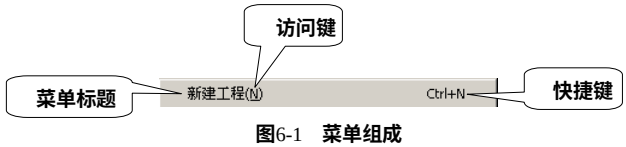


图6-1　菜单组成

菜单按出现位置的不同可分为菜单栏和弹出式菜单两种，菜单栏出现在窗体的标题栏下面，包括每个菜单的标题，例如，在 Visual Basic 6.0 主界面上，标题栏下面的一栏便是菜单栏，如图 6-2 所示；弹出式菜单只有在按下鼠标右键时才出现，是一个上下文相关的菜单，例如，在窗体上单击鼠标右键所弹出的菜单便是弹出式菜单，如图 6-3 所示。

图6-2　Visual Basic 6.0 的菜单栏

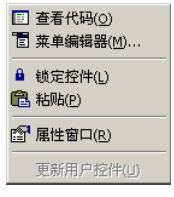


图6-3　弹出式菜单

在菜单栏中，各个菜单项又有主菜单和子菜单之分，依此可以将菜单分为不同级别的菜单。在窗体上直接显示出来的菜单级别最高，为一级菜单，又称为标题菜单，一级菜单的子菜单级别次之，为二级菜单，依次类推，将菜单分为不同级别的菜单，如图 6-4 所示，在 Visual

Basic 6.0 中，能将菜单分为 6 个级别。如果一个菜单（除一级菜单外）是主菜单，含有子菜单时，则其右端会显示一个标示符 ▸，如图 6-4 所示。

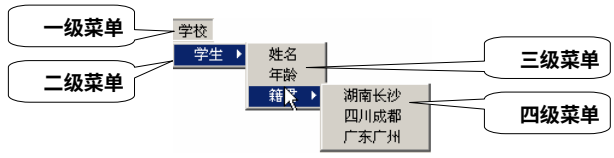


图6-4　不同级别的菜单

菜单的主次之分是相对而言的，一个菜单可以是一个主菜单，同时也可以是另外一个菜单的子菜单。例如，在图 6-4 所示的菜单栏中，【学生】菜单是【姓名】、【年龄】、【籍贯】菜单的主菜单，同时它又是【学校】菜单的子菜单。

6.2 【菜单编辑器】

在 Visual Basic 6.0 中，【菜单编辑器】可以用来创建任何菜单，同时还可以对菜单的属性进行设置。启动如图 6-5 所示的【菜单编辑器】有以下 3 种方法：

- 依次单击【工具】/【菜单编辑器】菜单。
- 单击工具栏中的按钮。
- 在窗体空白处单击鼠标右键，在弹出的菜单中单击【菜单编辑器】或直接使用 Ctrl+E 快捷键。

在启动【菜单编辑器】之前，必须先选中一个窗体，否则【菜单编辑器】菜单、按钮都为为灰色，表示不可用。

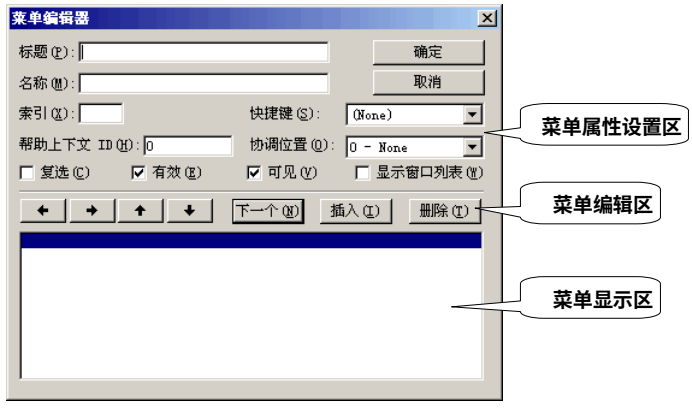


图6-5　【菜单编辑器】

【菜单编辑器】由 3 个部分组成，由上到下依次分为菜单属性设置区、菜单编辑区、菜单显示区 3 个区域，如图 6-5 所示。

(1) 菜单属性设置区

菜单属性设置区主要是用来设置菜单的相关属性，主要组成部分包括：

- 【标题】：用来输入菜单的标题文字，相当于控件的 Caption 属性。
- 【名称】：用于输入菜单的名称，相当于控件的 Name 属性，每个菜单的名称必须是惟一的。

- 【快捷键】：设置与菜单功能等价的快捷键，用户可以单击下拉列表右端的箭头，(None)，从下拉列表中选择相应的快捷键。快捷键一般也是惟一的。默认值为“None”。
- 【复选】：该属性决定是否在菜单前面加上选中符号。当选择该选项时，☑ 复选(C)，即将【复选】属性设为“True”，则在相应菜单的前面显示一个“✓”；当不选择该选项时，☐ 复选(C)，即将【复选】属性设为 False，则在相应菜单的前面不显示一个“✓”。默认值为“False”，☐ 复选(C)。
- 【有效】：该属性决定菜单是否可用，相当于控件的 Enabled 属性。当选择该选项时，☑ 有效(E)，即将【有效】属性设为“True”，则菜单可用；当不选择该选项时，☐ 有效(E)，即将“有效”属性设为“False”，则菜单不可用，菜单为灰色；默认值为“True”，☑ 有效(E)。
- 【可见】：该属性决定菜单是否可见。当选择该选项时，☑ 可见(V)，即将【可见】属性设为“True”，则菜单可用；当不选择该选项时，☐ 可见(V)，即将【可见】属性设为“False”，则菜单不可见，不被显示在菜单栏上，此时菜单变为弹出式菜单；默认值为【True】，☑ 可见(V)。

(2) 菜单编辑区

菜单编辑区主要是由一些按钮组成，用来编辑菜单和设置菜单级别。

- ← 按钮：将当前菜单的级别升一级，级别最高为 1 级。
- → 按钮：将当前菜单的级别降一级，级别最低为 6 级。
- ↑ 按钮：将当前菜单向上移动一个位置。
- ↓ 按钮：将当前菜单向下移动一个位置，如果当前菜单为最后一个菜单，则在当前菜单的位置新建一个一级菜单，并将当前菜单向下移动一个位置。
- 下一个(N) 按钮：将光标从当前菜单移动到下一个菜单，如果当前菜单为最后一个菜单，则在最后新建一个与当前菜单级别一样的菜单，并且光标停留在最后一个菜单上。
- 插入(I) 按钮：在当前菜单的前面插入一个和当前菜单级别一样的菜单。
- 删除(T) 按钮：删除当前菜单。

(3) 菜单显示区

所有已建的菜单都会在此区域显示，蓝色光条所在的位置就是当前被选中的菜单，如图 6-6 所示。菜单前面的内缩符号“…”是用来区分菜单级别的，菜单前面无内缩符号，表示此菜单为一级菜单；菜单前面有 1 个内缩符号“…”表示此菜单为二级菜单；菜单前面有两个内缩符号“…”表示菜单为三级菜单，依次类推，将菜单分为 6 个级别。菜单前面多一个内缩符号，表示菜单的级别降了一级，如图 6-6 所示。

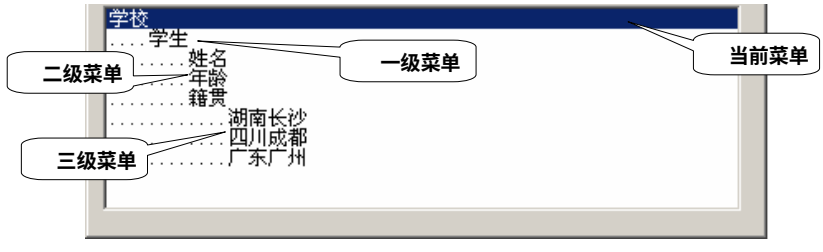


图6-6　菜单列表

6.3 菜单栏的设计

设计菜单栏，不仅是为了美化窗体，更重要的是为了让用户能够更加简单地进行操作，因此在设计菜单时，必须很好地控制菜单的状态，整体规划菜单的结构。菜单状态的控制，可以通过在【菜单编辑器】中设置属性来实现，也可以通过编写代码来实现；菜单结构的规划，主要是规划好各个菜单的级别，并且要规划好设计的先后顺序。在默认情况下，有多少个一级菜单，在菜单栏上就有多少个标题菜单，并按设计的先后顺序，依次排列在菜单栏上。

6.3.1 菜单栏的设计

在设计菜单栏之前，必须先规划好菜单栏的结构，对各个菜单的级别要做到心中有数。对于含有子菜单的菜单，其标题必须要有提示的作用，能够概括其子菜单的公共功能或属性。例如，在图 6-7 所示的菜单栏中，【字体】、【样式】、【颜色】菜单都含有子菜单，因此它们的标题都概括了其子菜单的公共属性。除了规划好菜单的级别之外，还要考虑一下是否为菜单设置访问键和快捷键，无论是主菜单还是子菜单都可以设置访问键，但只有不含有子菜单的菜单才能设置快捷键。例如，在图 6-7 所示的菜单栏中，【字体】、【样式】、【大小】主菜单及【16】、【24】子菜单都有自己的访问键，但只有不含有子菜单的【16】、【24】菜单才能设置快捷键。

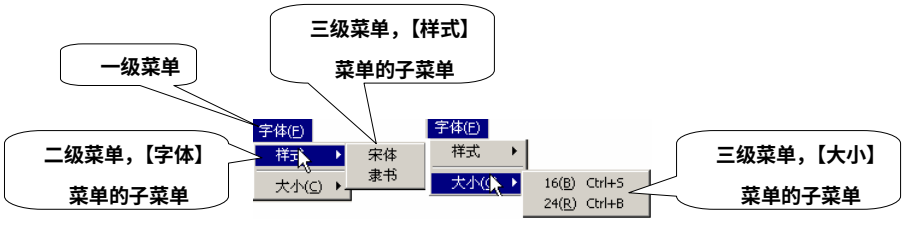


图6-7 菜单栏及其菜单系统

当菜单栏的菜单系统规划好后，使用【菜单栏编辑器】来设计菜单栏就非常容易。

【例6-1】 使用【菜单编辑器】设计如图 6-7 所示的菜单栏及其子菜单。

1. 单击【工具】/【菜单编辑器】菜单，打开如图 6-5 所示的【菜单编辑器】对话框。
2. 在【菜单编辑器】对话框的【标题】栏中输入“ 字体 ”，在【名称】栏中输入“ mnuFont ”。
3. 单击【菜单编辑器】对话框的 下一个(N) 按钮，新建下一个菜单，并设置菜单的有关属性。在【标题】栏中输入“ 样式 ”，在【名称】栏中输入“ mnuFontStyle ”。
4. 重复第 3 步，按表 6-1 的顺序新建所有的菜单，并设置相关属性，属性值见表 6-1。新建所有菜单后，【菜单编辑器】的格式如图 6-8 所示，在菜单显示区，单击某个菜单，选中该菜单，该菜单的有关属性就会相应地显示在属性区，如图 6-8 所示。

表 6-1 各菜单项属性值

序号	标题	名称
1	宋体	mnuFontStyle1
2	隶书	mnuFontStyle2
3	-（上划线）	MnuFontSeprator
4	大小（）	MnuFontSize

续表

序号	标题	名称
5	16（）	mnuFontSize1
6	24（）	mnuFontSize2

图6-8　生成所有菜单后的【菜单编辑器】对话框

5. 在菜单显示区中单击【字体】这一栏，选中【字体】菜单项，在【标题】栏的括号（）中加入“ &F ”，如图 6-9 所示。以同样的方法在【大小】菜单项的【标题】栏中加入“ &C ”，【16】菜单项的【标题】栏中加入“ &B ”，【24】菜单项的【标题】栏中加入“ &S ”。
6. 在菜单显示区中单击【16】这一栏，选中【16】菜单项，然后单击【快捷键】组合框右端的箭头，打开下拉列表，在下拉列表中单击“ Ctrl＋S ”选项，这时“ Ctrl＋S ”就会显示在组合框中，如图 6-10 所示。以同样的方法在菜单项【24】的【快捷键】组合框中选择“ Ctrl＋B ”选项。

图6-9　标题栏输入框

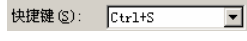


图6-10　快捷键组合框

7. 完成第 6 步之后，所有菜单项的属性已经设置完毕，这时【菜单编辑器】对话框的样式如图 6-11 所示。

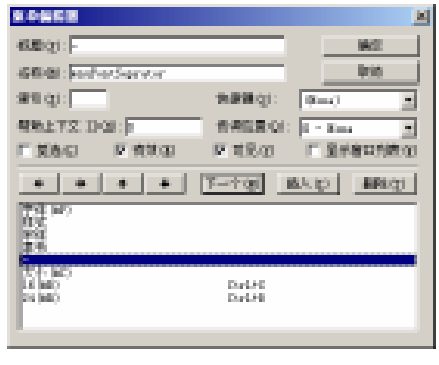


图6-11　属性设置完毕后的【菜单编辑器】对话框

8. 在菜单显示区中选中【样式】菜单项，单击→按钮，将【样式】菜单项的级别降一级，成为二级菜单。
9. 单击↓按钮，将光标移到下一个菜单，即【宋体】菜单项，单击→按钮两

次，将【宋体】菜单项的级别降二级，成为三级菜单。

10. 以同样的方法调整其他菜单项的级别，菜单级别见表 6-2，调整菜单级别后的【菜单编辑器】如图 6-12 所示。

表 6-2　　菜单级别

菜单	级别
隶书	三级菜单，【样式】菜单的子菜单
-	二级菜单，分隔线
大小	二级菜单，【字体】菜单的子菜单
16	三级菜单，【大小】菜单的子菜单
24	三级菜单，【大小】菜单的子菜单

11. 单击【菜单编辑器】对话框上的 确定 按钮，便会在主窗体上生成如图 6-7 所示的菜单栏。

从【例 6-1】中，可以看出在规划好了各个菜单的级别后，用【菜单编辑器】设计菜单一般有以下 5 个步骤：

1. 打开【菜单编辑器】，即【例 6-1】的第 1 步。
2. 先不管菜单的级别，按顺序新建所有的菜单，即【例 6-1】的第 2～4 步。
3. 为相关菜单设置访问键、快捷键，即【例 6-1】的第 5～7 步。
4. 编辑调整菜单级别，即【例 6-1】的第 8～10 步。
5. 单击【菜单编辑器】对话框上的 确定 按钮，生成菜单栏，即【例 6-1】的第 11 步。

图6-12　调整菜单级别后的【菜单编辑器】

以上步骤不一定要按顺序严格执行，也可以第 2、3、4 步同时进行，对于初学者，最好是按上面的步骤来设计菜单栏，有一定基础后，可以不按以上步骤来设计菜单栏。

和设计基本控件一样，在用【菜单编辑器】设计菜单的同时，也可以设置菜单的有关属性。

- 【标题】和【名称】

菜单的【标题】就相当于一个人的名字，【名称】就像是一个人的身份证，每个人都有自己的名字，几个人可以取同一个名字，但每个人的身份证都必须是惟一的，一个人只能有一个身份证号，一个身份证号只能属于一个人。因此每个菜单都必须有一个【标题】和一个【名

称】，【标题】不一定是惟一的，而【名称】必须是惟一的。

- 【快捷键】和【访问键】

菜单的【快捷键】是通过用 Ctrl 和其他键的组合来设置的，菜单设置快捷键后，可以直接使用快捷键来执行菜单的操作，比如说，在 Windows 系统中，使用 Ctrl＋C 来执行拷贝操作，使用 Ctrl＋V 来执行粘贴操作。快捷键的设置是在【快捷键】组合框中完成的，只要从组合框的下拉列表中选择相应的项就可以为菜单设置快捷键。例如，在【例 6-1】中，就为【16】菜单设置了 Ctrl＋S 的快捷键，可以直接通过 Ctrl＋S 来执行相应的操作。

【访问键】是通过 Alt 和字母键的组合来设置的，菜单设置访问键后，可以直接使用访问键来访问菜单，但不执行菜单的操作，比如说，在 Windows 系统中，可以使用 Alt＋E 来访问【编辑】菜单，打开其下拉菜单。菜单访问键的设置是在菜单的【标题】栏中完成的，只要在相应的字母前加上“ & ”，便可以使其成为访问键。例如，在【例 6-1】中，在【字体】菜单的【标题】栏中，“ &F ”便是为菜单设置了 Alt＋F 访问键，用户可以直接通过 Alt＋F 打开其下拉菜单。

【快捷键】、【访问键】的设置最好都是惟一的，不能重复使用。另外，含有子菜单的菜单项不能设置快捷键。

当一个菜单有多个子菜单时，并且各个子菜单的功能又有所不同时，这时有必要在子菜单之间加入间隔线，以示区别，如图 6-13 所示。分隔线也是菜单的一种，只是菜单的“ 标题 ”必须是“ - ”（上划线），如图 6-12 所示。

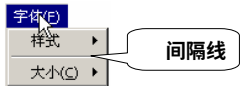


图6-13　加入间隔线

6.3.2　菜单事件

设计菜单的目的是用它来组织和管理应用程序中的窗体或者是代替命令按钮来完成某项任务，为了达到此目的，菜单设计好后，还必须为菜单添加相应的事件。菜单只有惟一的 Click 事件，但并不是所有的菜单都能响应 Click 事件，只有那些没有子菜单的菜单才有响应 Click 事件的能力，分隔线也不能响应 Click 事件。当菜单设计完毕后，在窗体设计窗口单击某个菜单，就为该菜单添加了 Click 事件，例如，在【例 6-1】中，在窗体设计器中单击【字体】/【样式】/【宋体】菜单，就为【字体】菜单添加了 Click 事件，如图 6-14 所示。

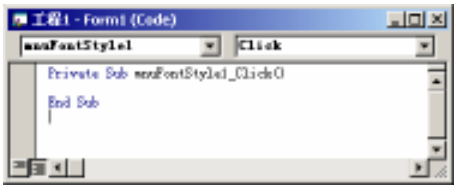


图6-14　添加 Click 事件

【例6-2】　在【例 6-1】的基础上，向窗体添加一个文本框控件。通过单击【字体】/【样式】/【宋体】菜单或【字体】/【样式】/【隶书】菜单来改变文本框中汉字的样式；通过单击【字体】/【大小】/【16】菜单或【字体】/【大小】/【24】菜单来改变文本框中汉字的大小。另外当【样式】菜单的某子菜单被单击时，在其前面显示选中

符号“ ✔ ”。

1. 打开【例 6-1】所建的工程。
2. 向窗体添加一个文本框控件，将控件的【名称】属性设为“ txtText ”，删除“ Text ”属性中的“ Text1 ”，“ MultLine ”属性设为“ True ”，并调整文本框的大小，调整后的窗体如图 6-15 所示。

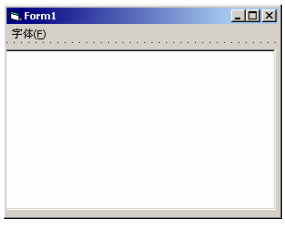


图6-15　添加文本框控件和菜单栏后的窗体

3. 在窗体设计窗口，单击【字体】/【样式】/【宋体】菜单，为【宋体】菜单添加 Click 事件。以同样的方法，为【隶书】菜单、【16】菜单、【24】菜单分别添加 Click 事件，并在代码窗口添加如下代码：

```
Private Sub mnuFontSize1_Click()
    '单击菜单【16】，文本框中文字的大小为 16
    txtText.FontSize = 16
End Sub

Private Sub mnuFontSize2_Click()
    '单击菜单【24】，文本框中文字的大小为 24
    txtText.FontSize = 24
End Sub

Private Sub mnuFontStyle1_Click()
    '单击菜单【宋体】，文本框中文字的样式为宋体
    txtText.FontName = "宋体"
    '单击菜单【宋体】，在其前面添加选中符号
    mnuFontStyle1.Checked = True
    '单击菜单【宋体】，去掉菜单【隶书】前面的选中符号
    mnuFontStyle2.Checked = False
End Sub

Private Sub mnuFontStyle2_Click()
    '单击菜单【隶书】，文本框中文字的样式为隶书
    txtText.FontName = "隶书"
    '单击菜单【隶书】，去掉菜单【宋体】前面的选中符号
    mnuFontStyle1.Checked = False
    '单击菜单【隶书】，在其前面添加选中符号
    mnuFontStyle2.Checked = True
End Sub
```

4. 运行程序，在文本框中输入“ Visual Basic 6.0 可视化编程 ”后，分别单击【字体】/【样式】/【隶书】菜单、【字体】/【样式】/【宋体】、【字体】/【大小】/

【16】、【字体】/【样式】/【24】菜单，测试各个菜单事件。

6.4 弹出式菜单的设计

弹出式菜单又称上下文菜单或快捷菜单，一般不在窗体上直接显示出来，只有在用户单击鼠标右键时才会显示，例如在 Visual Basic 6.0 中，在窗体上单击鼠标右键，就会弹出如图 6-16 所示的菜单。弹出式菜单又称为快捷菜单，是独立于菜单栏而显示在窗体上的浮动菜单，和菜单栏上的菜单一样，弹出式菜单也是用【菜单编辑器】来设计的。在 Visual Basic 6.0 中，任何含有子菜单的菜单都可以作为弹出式菜单而显示在窗体上，不管该菜单是菜单栏上已经存在的菜单还是不存在的，如果要使某个菜单成为弹出式菜单，只需要使用 PopupMenu 的方法来显示该菜单就可以了。

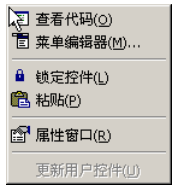


图6-16　弹出式菜单

【例6-3】　新建一个工程，当用户在窗体上单击鼠标右键时，弹出如图 6-17 所示的菜单。

1. 新建一个工程。
2. 选中窗体，单击【工具】/【菜单编辑器】菜单打开【菜单编辑器】对话框。
3. 按表 6-3 的顺序及属性值新建所有菜单。

图6-17　弹出式菜单

表 6-3　　各菜单的属性值

序号	【标题】栏	【名称】栏	级别
1	颜色	mnuColor	一级菜单
2	红色	mnuColorRed	二级菜单，【颜色】菜单的子菜单
3	蓝色	mnuColorBlue	二级菜单，【颜色】菜单的子菜单
4	绿色	mnuColorGreen	二级菜单，【颜色】菜单的子菜单

4. 在菜单显示区中单击【颜色】栏，选中【颜色】菜单项，将【颜色】菜单项的【可见】属性设为“ False ”。
5. 菜单设计完毕后，【菜单编辑器】如图 6-18 所示，单击【菜单编辑器】的 确定 按钮，回到主窗体。

图6-18　【菜单编辑器】

6. 在窗体资源管理器中单击图标 Form1 (Form1.frm) 选中主窗体，单击窗体资源管理

器最上面的图标，打开【代码】窗口，在【代码】窗口的对象名列表中选择窗体名“ Form ”，在事件/过程列表中选择“ MouseDown ”事件，为窗体添加 MouseDown 事件，如图 6-19 所示。

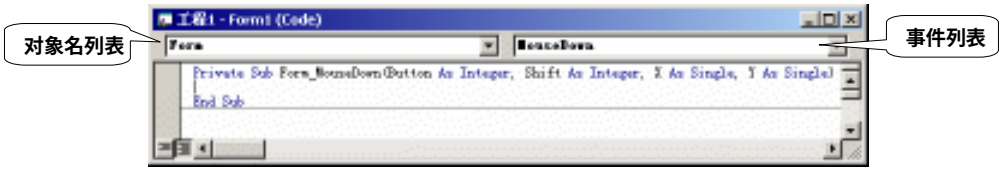


图6-19　添加 MouseDown 事件

7. 在窗体的 MouseDown 事件中添加如下代码：

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, _
X As Single,Y As Single)
    '判断单击的是否是右键
    If Button = 2 Then
    '单击右键，显示菜单【颜色】的子菜单
        Form1.PopupMenu mnuColor
    End If
End Sub
```

8. 运行程序，在窗体上单击鼠标右键，便弹出 6-17 所示的菜单。

在【例 6-3】中，【颜色】菜单便是弹出式菜单，由于其【可见】属性为“ False ”，因此【颜色】菜单及其所属的子菜单便不会显示在窗体上，为了将【颜色】菜单的子菜单显示出来，在【例 6-3】中使用了 PopupMenu 方法来显示其子菜单。PopupMenu 方法的语法结构如下：

```
[object] .PopupMenu menuname [,flags [,X [,Y [,boldcommand]]]]
```

参数的说明见表 6-4。

表 6-4　　PopupMenu 方法参数说明

	选择性	说明
Object	可选项	对象名，表示打开弹出式菜单的对象，缺省时为当前被选中的对象
menuname	必选项	弹出式菜单的名称，必须含有子菜单
flags	可选项	设定弹出式菜单的位置及条件，由位置参数和条件参数组成
X，Y	可选项	弹出式菜单显示的位置，缺省时为鼠标单击的位置
boldcommand	可选项	指定以粗体显示的菜单控件的名称，缺省时弹出式菜单中没有以粗体显示的内容

flags 参数由位置参数和条件参数组成，位置参数决定弹出式菜单显示的位置，条件参数决定弹出式菜单显示的条件。位置参数的取值见表 6-5，条件参数取值见表 6-6。指定一个位置常数和一个条件常数，中间用“ or ”操作符相连，即可为 flags 参数指定一个值，缺省值由位置参数、条件参数各自的缺省值共同组成。

由于 PopupMenu 方法只有一个必选的参数，因此可以使用以下最简单的形式：

对象名称.PopupMenu　菜单名称

例如，在【例 6-3】中，所采用的形式便是最简单的形式（Form1.PopupMenu mnuColor），“Form1”便是窗体的名称，“mnuColor”是菜单的名称。

object 参数的选择可以实现在不同的区域单击鼠标右键时，弹出不同内容的菜单。例如，在 Visual Basic 6.0 的不同窗口单击鼠标右键时，弹出的菜单各不一样。

表 6-5　flags 的位置参数

常数	值	说明
vbPopupmenuLeftAlign	0	向左对齐，缺省值
vbPopupmenuCenterAlign	4	向中间对齐
vbPopupmenuRightAlign	8	向右边对齐

表 6-6　flags 的条件参数

常数	值	说明
vbPopupMneuLeftButton	0	单击鼠标左键才显示弹出式菜单，缺省值
vbPopupMneuRightButton	2	单击鼠标右键才显示弹出式菜单

PopupMenu 方法显示的是菜单 menuname 的子菜单，但菜单 menuname 本身并不被显示，并且 PopupMenu 方法每次只能打开一个弹出式菜单。

弹出式菜单通常是在用户单击鼠标右键时才出现的，因此弹出式菜单的显示一般是在对象的 MouseDown 事件中完成。在【例 6-3】中，弹出式菜单的显示便是在窗体的 MouseDown 事件中完成的。

如果将菜单【颜色】的【可见】属性设为“True”，再执行程序，看看有什么效果，读者不妨试一试。

为了让弹出式菜单发挥其功能，还必须为弹出式菜单添加 Click 事件。由于弹出式菜单不显示到窗体上，因此不能像菜单栏上的菜单那样，在窗体上单击某个菜单，就为其添加 Click 事件，而是要在【代码】窗口中为其添加 Click 事件。

【例6-4】　为【例 6-3】所设计的弹出式菜单添加 Click 事件，在单击某个颜色菜单项后，窗体的底色变为相应的颜色。例如，在弹出菜单中单击【蓝色】菜单，窗体变为如图 6-20 所示。

1. 打开【例 6-3】所建的工程。
2. 双击窗体，打开【代码】窗口，在【代码】窗口的对象名列表中选择菜单名“mnuBlue”，在事件列表中选择“Click”事件，为【蓝色】菜单添加 Click 事件。以同样的方法为【红色】、【绿色】菜单添加 Click 事件。
3. 在代码窗口增加如下代码：

```
Private Sub mnuColorBlue_Click()
    Form1.BackColor = vbBlue
End Sub
```

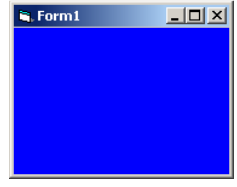


图6-20　程序运行后的主窗体

```
Private Sub mnuColorGreen_Click()
    Form1.BackColor = vbGreen
End Sub

Private Sub mnuColorRed_Click()
    Form1.BackColor = vbRed
End Sub
```

4. 运行程序，在窗体上单击鼠标右键，在弹出的菜单中选择【蓝色】菜单，主窗体就会变为如图 6-20 所示。单击其他弹出式菜单，测试相应的事件。

6.5 工具栏的设计

在 Visual Basic 6.0 中，工具栏一般显示在菜单栏下面，由一些命令按钮组成，并且每个按钮上都有图像，如图 6-21 所示，每个命令按钮都有相应的菜单项与之对应，可看作是相应菜单项的快捷方式。例如在 Visual Basic 6.0 中，工具栏上的按钮便是【工具】/【菜单编辑器】菜单的快捷按钮，点击按钮也可以直接打开【菜单编辑器】对话框。

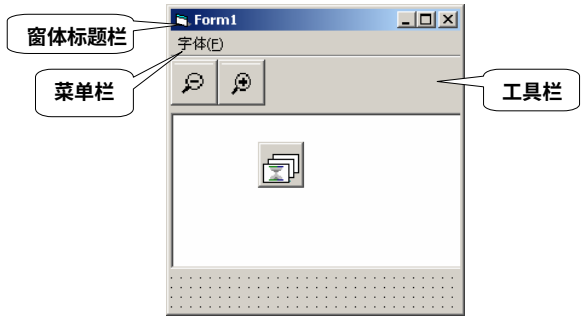


图6-21 工具栏的位置

要创建如图 6-21 所示的工具栏，必须先向工具箱中添加工具条控件和图像列表控件。可按以下方法向工具箱中添加工具条控件和图像列表控件。

1. 单击【工程】/【部件】菜单打开【部件】对话框，如图 6-22 所示。

图6-22 【部件】对话框

2. 单击【部件】对话框最上面的【控件】选项卡，在【控件】的列表中选中

“Microsoft Windows Common Control 6.0”，并单击左边的小方框，这时【控件】对话框变为如图 6-22 所示。

3. 单击 确定 按钮，关闭【部件】对话框，这时工具箱中就新增如图 6-23 所示的控件。

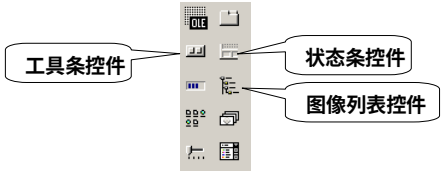


图6-23　工具箱新增的控件

在工具箱中有了工具条控件和图像列表控件后，便可以使用它们来创建工具栏了。工具栏是通过用工具条控件来设计，但在向窗体添加工具条之前，必须先向窗体添加图像列表控件。窗体添加工具条控件后，工具条控件会自动加到菜单栏下面，如果窗体上没有菜单栏，便直接加到窗体标题栏下面，其宽度是不可变的，和窗体的宽度一样，其高度随按钮图像大小的改变而自动调整，不需要去设定。

【例6-5】　为【例 6-2】创建一个如图 6-21 所示的工具栏。

1. 打开【例 6-2】所建的工程。
2. 按以上所讲的步骤向工具箱中添加如图 6-23 所示的控件。
3. 在工具箱中，双击图像列表控件，向窗体添加图像列表控件，窗体的样式如图 6-24 所示。

图像列表控件的大小是不能改变的，并且在程序运行的过程中，不被显示在窗体上。

4. 在窗体上单击图像列表控件，选中图像列表控件，然后在图像列表控件上单击鼠标右键，从弹出的菜单中单击【属性】菜单，便弹出如图 6-25 所示的【属性页】对话框。
5. 单击【32×32（3）】前面的圆圈，选中该项，如图 6-25 所示。

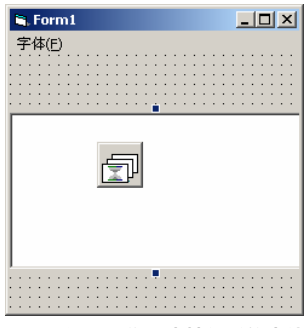


图6-24　添加图像列表控件后的窗体

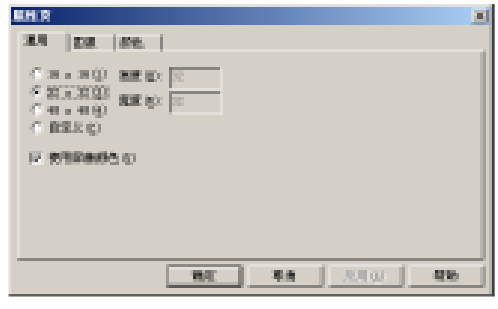


图6-25　【属性页】对话框的【通用】属性页

6. 单击【属性页】对话框最上面的【图像】选项卡，将【属性页】切换到【图像】属性页，如图 6-26 所示，单击 插入图片(P)... 按钮，便弹出如图 6-27 所示的【选定图像】对话框，在文件列表中单击“放大图标.bmp”文件，然后按住 Ctrl 键，单击“缩小图标.bmp”文件，这时这两个文件都被选中，如图 6-27 所示（这两个文件在本书的教学辅助光盘中提供）。

7. 单击 打开(O) 按钮，这是在“图像显示区”就会显示“放大图标.bmp”和“缩小小图标.bmp”两个文件所存储的图像，如图 6-28 所示。

图6-26 【属性页】对话框的【图像】属性页

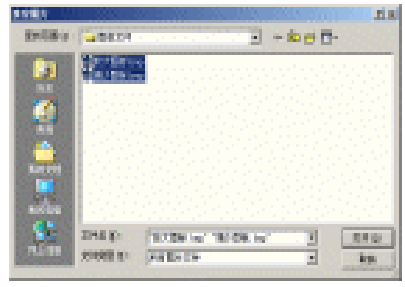


图6-27 【选定图像】对话框

图6-28 添加图像后的【属性页】对话框

在向图像列表控件中添加图像时，系统会按添加的顺序为每个图像赋一个“索引”值，如图 6-28 所示。

8. 单击【属性页】对话框上的 确定 按钮，回到主窗体。
9. 在工具箱中双击工具条控件，向窗体中添加工具条控件，如图 6-29 所示。
10. 在窗体上单击工具条控件，选中工具条控件，并在工具条控件上点击鼠标右键，从弹出的菜单中单击【属性】菜单，便弹出如图 6-30 所示的【属性页】对话框。单击【图像列表】栏右端的箭头，从下拉列表中选择“ImageList1”。

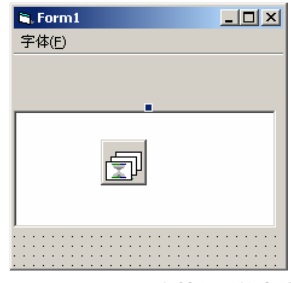


图6-29 添加工具条控件后的窗体

图6-30 【属性页】对话框的【通用】属性页

11. 单击【属性页】对话框最上面的【按钮】选项卡，将【属性页】对话框切换到【按钮】属性页，如图 6-31 所示。
12. 单击 插入按钮(N) 按钮，便向工具栏中加入一个按钮，在【工具提示文本】栏中输入“16 号字体”，在【图像】栏中输入“2”，如图 6-32 所示。

每插入一个按钮，系统便会按插入的先后顺序为每个按钮赋一个索引值，如图 6-31 所示。

13. 重复第 12 步，向工具栏新加入一个按钮，然后在【工具提示文本】栏输入“24 号字体”，在【图像】栏中输入“1”。
14. 单击 确定 按钮，便创建了如图 6-21 所示的工具栏。
15. 运行程序，将鼠标靠近工具栏的 按钮，这时在鼠标右下方会显示提示字符“16 号字体”，如图 6-33 所示。鼠标靠近工具栏的 按钮时，“24 号字体”提示字符也会显示鼠标的右下方。

图6-31　【属性页】对话框【按钮】属性栏

图6-32　添加按钮后的【属性页】对话框

图6-33　程序运行后的窗体

从【例 6-5】中可以看出，创建工具栏的过程比较复杂，这主要是由于工具栏上的按钮是含有图像的按钮所造成的，为此在创建工具栏之前，必须先创建一个装图像的“容器”以便装下按钮所要使用的图像。

(1) 在 Visual Basic 6.0 中，创建一个工具栏大概要经历以下 3 个过程：

- 向工具箱中添加图像列表控件和工具条控件，如【例 6-5】的第 2 步。
- 用图像列表控件创建一个装图像的“容器”，如【例 6-5】的第 3～8 步。
- 用工具条创建工具栏，如【例 6-5】的第 9～14 步。

(2) 在用图像列表控件创建图像“容器”的过程中，又要经历以下 4 个步骤：

- 向窗体中添加图像列表控件，如【例 6-5】的第 3 步。

- 打开图像列表控件的【属性页】对话框，如【例 6-5】的第 4 步。
- 设置图像的大小，如【例 6-5】的第 5 步。
- 添加图像，生成图像“容器”，如【例 6-5】的第 6～8 步。

(3) “图像”容器创建好后，便可以创建工具栏，并为工具栏上的按钮加载图像，具体步骤如下。

- 向窗体添加工具条控件，【例 6-5】的第 9 步。
- 打开工具条控件的【属性页】对话框，并将“图像”容器加载到工具条中，【例 6-5】的第 11 步。
- 创建工具栏按钮，并设置有关属性见表 6-7，【例 6-5】的第 12～13 步。
- 生成工具栏，【例 6-5】的第 14 步。

表 6-7　　按钮属性

属性	说明
工具提示文本	返回或设置按钮提示。程序运行时，鼠标靠近某个工具栏按钮 该属性所设的值就会被显示
图像	指定按钮所使用的是“图像”容器中的哪个图像，其值为“图像” 容器中图像的“索引”值

为了让工具栏上的按钮和具体的菜单对应起来，必须在工具栏创建完毕后，为工具栏控件添加 ButtonClick 事件。

【例6-6】　为【例 6-5】所设计的工具栏添加 ButtonClick 事件，使工具栏上的按钮和【字体】/【大小】/【16】菜单对应起来，按钮和【字体】/【大小】/【24】菜单对应起来。

1. 打开【例 6-5】所保存的工程。
2. 在窗体上双击工具栏，这是在代码窗口为工具栏添加了 ButtonClick 事件，如图 6-34 所示。

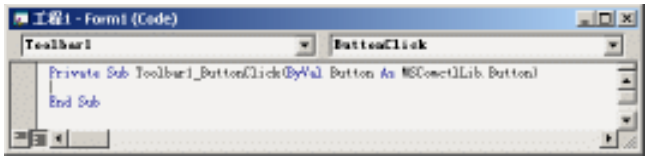


图6-34　为工具栏添加 ButtonClick 事件后的窗口

3. 在代码窗口新增如下代码：

```
Private Sub Toolbar1_ButtonClick(ByVal Button As MSComctlLib.Button)
    Select Case Button.Index
        Case 1
        '单击工具栏上的第一个按钮，执行【16】菜单的 Click 事件
            mnuFontSize1_Click
        Case 2
        '单击工具栏上的第一个按钮，执行【24】菜单的 Click 事件
            mnuFontSize2_Click
    End Select
```

```
End Sub
```

运行程序，在文本框中输入“Visual Basic 6.0 可视化编程”，然后单击工具栏的按钮，看看是否和单击【字体】/【大小】/【16】菜单的效果一样。

在【例 6-6】中，由于工具栏上含有多个按钮，因此在为其添加的 ButtonClick 事件中，还带有一个 Button 参数，该参数表示用户所单击的按钮，用户可根据 Button 参数来选择所要执行的操作，所使用的语法结构如下所示：

```
Select Case Button.Index
Case 1
    所要执行的操作代码
Case 2
    所要执行的操作代码
      ┆
End Select
```

由于工具栏上的按钮可以看作是菜单栏上某项菜单的快捷方式，因此单击工具栏上某个按钮所要执行操作便是单击相应菜单所要执行的操作，例如，在【例 6-6】中，单击工具栏上的第一个按钮所执行的代码，便是【16】菜单的 Click 事件所要执行的代码。

6.6 状态栏的设计

状态栏通常位于窗体的底部，由一些窗格组成，如图 6-35 所示，用于显示各类状态信息，比如说，系统时间、键盘键的状态等系统状态信息。状态栏在显示系统状态信息时，不会干扰主程序的执行，并且会自动的更新信息。

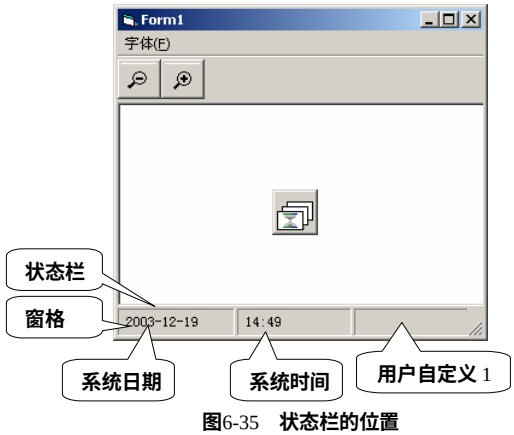


图6-35　状态栏的位置

状态栏的设计是通过状态条控件来完成的，在向工具箱中添加工具条按钮的同时，也向工具箱中添加了状态条控件，如图 6-23 所示。工具箱中有了状态条控件之后，便可以使用它来设计状态条了。在工具箱中，双击状态条控件，状态条控件便会自动的加到窗体的底部，其宽度是不可以改变的，和窗体一样宽，但高度可以改变。

【例6-7】　为【例 6-6】添加如图 6-35 所示的状态栏，并且在工具栏的【用户自定义 1】窗格显示字体的样式。

1. 打开【例 6-6】保存的工程。

2. 双击工具箱中状态条控件，向窗体添加状态条控件，添加状态条后的窗体如图 6-36 所示。

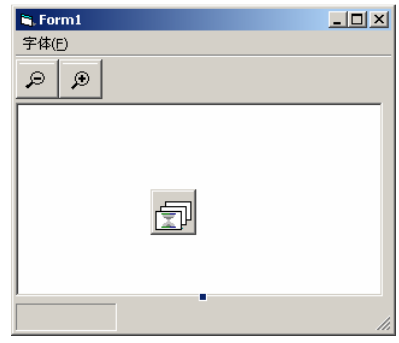


图6-36　添加状态条后的窗体

3. 在窗体上单击状态条控件，选中状态条控件，并在状态条上单击鼠标右键，在弹出的菜单中单击【属性】菜单，打开状态条的【属性页】对话框，如图 6-37 所示。
4. 单击【属性页】对话框的最上面的【窗格】选项卡，将【属性页】对话框切换到【窗格】属性页，如图 6-38 所示，这是在【窗格】属性页中显示的是第 1 个窗格的属性。
5. 单击【样式】栏右端的箭头，打开下拉列表，从下拉列表中单击“6－sbrDate”项，如图 6-39 所示。

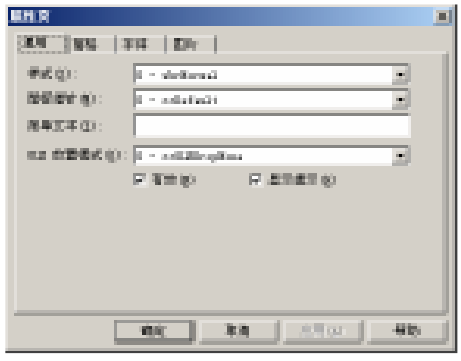


图6-37　【属性页】对话框

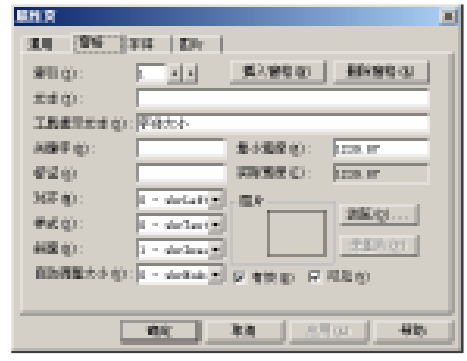


图6-38　【属性页】对话框的【窗格】属性页

图6-39　设置窗格属性后的【属性页】对话框

6. 单击 插入窗格(N) 按钮，新插入一个窗格，然后单击【索引】栏右端的箭头，将【索引】值变为“2”，这是【窗格】属性页中显示的是第 2 个窗格的属性，在【样式】栏的下拉列表中选中“5－sbrTime”项。

每插入一个窗格，系统便会按插入的先后顺序为每个窗格赋一个索引值，如图 6-38 所示。

7. 单击 插入窗格(N) 按钮，再插入一个窗格，然后单击【索引】栏右端的箭头，将【索引】值变为“3”，这是在【窗格】属性页中显示的是第 3 个窗格的属性，

在【样式】栏的下拉列表中选中“0－sbrText”项。

8. 单击【属性页】对话框的[确定]按钮，回到主窗体，添加状态栏的窗体如图 6-40 所示的状态栏。

9. 双击窗体，打开代码窗口，在代码窗口新增如下加底纹的代码：

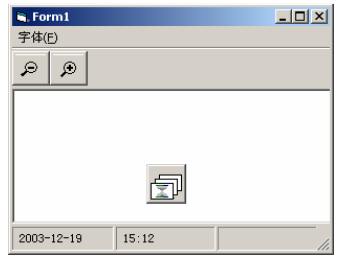


图6-40　添加状态栏后的窗体

```
Private Sub mnuFontStyle1_Click()
    txtText.FontName = "宋体"
    ’单击【菜单】,状态栏的第 3 个窗格显示“宋体”
    StatusBar1.Panels(3).Text = "宋体"
End Sub

Private Sub mnuFontStyle2_Click()
    txtText.FontName = "隶书"
    ’单击【菜单】,状态栏的第 3 个窗格显示“隶书”
    StatusBar1.Panels(3).Text = "隶书"
End Sub
```

10. 运行程序，在文本框中输入“Visual Basic 6.0 可视化编程”，然后单击【字体】/【样式】/【宋体】菜单，这是主窗体如图 6-41 所示。当单击【字体】/【样式】/【隶书】菜单，状态栏的第 3 个窗格相应显示“隶书”。

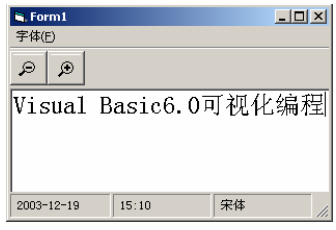


图6-41　程序运行后的窗体

在【例 6-7】中，状态栏中的【窗格】（Panels）不仅可以用来显示系统时间、系统日期等系统信息，而且还可以用来显示用户自定义的状态信息，用户所要做的只是将【窗格】的【样式】设为不同值而已，【样式】的常用属性值见表 6-8。表中第 1 栏属性值是用来显示用户自定义的状态信息，其余的属性值是用来显示各种不同的系统状态信息。

表 6-8　　【样式】属性值

属性值	说明
0_sbrText	缺省值，显示文本或位图，通过 Text 属性设置
1_sbrCaps	显示 Caps Lock 键的状态，有效时显示 CAPS，无效时变为灰色
2_sbrNum	显示 Num Lock 键的状态，有效时显示 NUM，无效时变为灰色
3_sbrIns	显示 Insert 键的状态，有效时显示 INS，无效时变为灰色
4_sbrScrl	显示 Scroll Lock 键的状态，有效时显示 SCRL，无效时变为灰色
5_sbrTime	以系统格式显示当前时间
6_sbrDate	以系统格式显示当前日期

当窗格用来显示系统的状态信息时，只需要将窗格的【样式】属性设为相应的值即可，系统会自动的更新窗格中的状态信息。例如，在【例 6-7】中，第 1 个所显示的日期，第 2 个窗格所显示的时间信息。当窗格用来显示自定义的状态信息时，必须将【样式】属性设为“0_sbrText”，并且还要使用下面的语法结构在相应的事件中将状态信息赋给窗格的 Text 属性：

状态条名.Panels(窗格索引值).Text=状态值

例如，在【例 6-7】中，便是在【宋体】、【隶书】菜单的 Click 事件中，将“宋体”、“隶书”两种不同的状态值赋给窗格的 Text 属性后，才会在第 3 个窗格中显示文本框中字体的样式。

6.7 小结

本章所介绍的内容是美化窗体不可缺少的部分，菜单栏、工具栏、状态栏的建立，不仅使得窗体得以美化，而且还使得窗体的功能进一步加强。

在本章主要介绍了以下内容：

- 菜单栏的设计和使用方法。
- 弹出式菜单的设计和使用方法。
- 图像列表控件、工具条控件、状态条控件的添加过程。
- 使用图像列表控件、工具条控件设计工具栏的方法即工具栏的使用方法。
- 状态栏的设计和使用方法。

6.8 习题

一、 填空题

1. 在 Visual Basic 6.0 中一个完整的菜单包括____、____、____3 项，其中____是每个菜单必须有的。
2. 菜单按出现的位置的不同可分为____和____两种，其中____一般显示在窗体标题栏下面，而____只有在单击鼠标右键的时候才出现。
3. 菜单编辑器由____、____、____3 部分组成，所有设计好的菜单都会在____中显示出来，并且通过____来区分菜单的级别。
4. 弹出式菜单一般不直接显示在窗体上，既可以是菜单栏中的菜单项，也可以是____属性设为 False 的菜单项。要显示弹出式菜单可以用____方法。
5. 工具栏一般显示在____下面，由____组成；状态栏一般显示在窗体的____，由____组成。

二、 选择题

1. 直接显示在窗体上的菜单项是（ ）。
   A. 一级菜单　　B. 二级菜单　　C. 三级菜单　　D. 四级菜单
2. 要使一个菜单项变为分隔线，必须将其标题属性设为（ ）。
   A. 下划线　B. &　C. 上划线　D. 减号
3. 含有子菜单的菜单不能设置（ ）。
   A. 访问键　　B. 快捷键　　C. 菜单标题　　D. 菜单名称
4. 下面哪些事件是菜单能响应的（ ）。
   A. Change 事件　　B. MouseDown 事件　　C. MouseUp 事件　　D. Click 事件

三、 程序设计题

设计 1 个可以改变字体大小及颜色的应用程序，程序的界面如图 6-42 所示。具体要求如下：

1. 单击【字体】/【大小】/【16】、【字体】/【大小】/【24】菜单，文本框的字体变为相应的大小。

2. **单击【字体】/【颜色】/【红色】、【字体】/【颜色】/【蓝色】菜单，文本框的字体变为相应的颜色，并且所选的颜色显示在状态栏的第 3 个窗格中。**
3. **单击【查看】/【工具栏】、【查看】/【状态栏】菜单，状态栏、工具栏在可见和不可见之间切换，并且相应的菜单在有无选中符号 ✔ 之间切换。（提示：参照例 6-7。）**

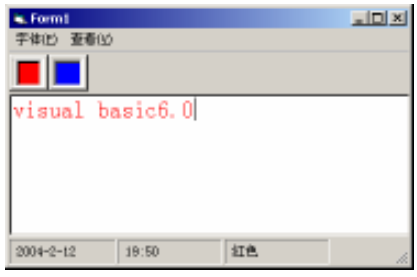


图6-42　程序界面

第7章　对话框的使用

对于对话框，大家并不陌生。在使用 Windows 系统时，如果执行了一次非法的操作，便会弹出一个提示对话框。在 Visual Basic 6.0 中，对话框是一种特殊的窗体，它通过一个或多个简单的控件与用户交互，获取用户简单的输入或向用户提示有关信息。【预定义】对话框、【自定义】对话框、【通用】对话框是 Visual Basic 6.0 中最基本的 3 种对话框，这 3 种对话框的创建过程和调用方法，将是本章主要学习的内容。

本章学习目标

- 对话框的调用和显示。
- 【预定义】对话框。
- 【通用】对话框。
- 【自定义】对话框。

7.1 对话框的调用和显示

在 Visual Basic 6.0 中，对话框一般都是一种弹出式的窗口。例如，启动 Visual Basic 6.0 后，在主界面上单击【文件】/【保存工程】菜单后，便弹出如图 7-1 所示的窗口，这个窗口便是对话框中的一种。

在 Visual Basic 6.0 中，对话框分为模态对话框和非模态对话框两种类型。

模态对话框比较常用，显示重要信息的对话框一般都是模态对话框。模态对话框要求必须先对对话框作出响应，才能继续其他工作，这样在未关闭对话框之前，就不能继续应用程序其他操作。例如：启动 Visual Basic 6.0 后，在主界面上单击【文件】/【保存工程】菜单，弹出的【文件另存为】对话框，如图 7-1 所示，【文件另存为】对话框便是模态的。在未完成或取消文件另存任务之前，Visual Basic 6.0 主界面上的任何控件、按钮、菜单都不能被选中。

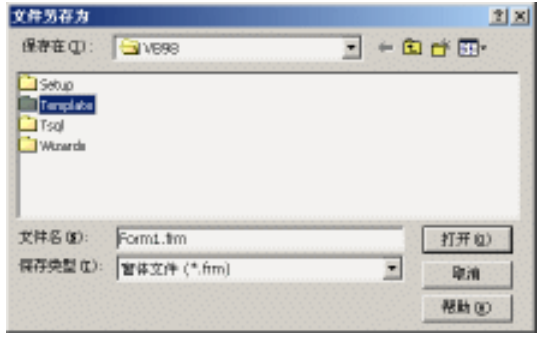


图7-1　【文件另存为】对话框（模态对话框）

非模态一般比较少用，只是用来显示频繁使用的命令或信息。非模态对话框允许不一定要对对话框作出响应，才能继续应用程序其他操作，这样在不用关闭对话框的情况下，还可以继续进行其他工作。例如：启动 Visual Basic 6.0 后，在主界面上单击【帮助】/【搜索】菜单，弹出如图 7-2 所示的【MSDN Library Visual Studio】对话框，此对话框便是非模态的。可以在使用“MSDN Library Visual Studio 6.0”的同时，继续在 Visual Basic 6.0 编程环境上的进行其他操作，两者互不干扰。

图7-2　【MSDN Library Visual Studio】对话框（非模态对话框）

对话框设计完后，用户要显示对话框，则必须使用 Show 方法，Show 方法的语法结构如下：

```
对话框名.Show[样式]
```

“样式”为可选项，是用来指定对话框是模态还是非模态的整数值。若“样式”为 0，则对话框为非模态的；若“样式”为 1，则对话框为模态的；如果不给“样式”赋值，则取默认值 0。

对话框被使用完后，如果想把对话框隐藏起来，这时就要用 Hide 方法来实现，其语法结构如下：

```
对话框名.Hide
```

隐藏对话框后，对话框变为不可见，和将其 Visible（可见）属性设置为 False（否）的效果一样。

Hide 方法只能将对话框隐藏起来，但不能使对话框卸载，要使对话框卸载的话，只能通过 Uload 方法。

7.2　【预定义】对话框

【预定义】对话框是 Visual Basic 6.0 为用户已经设计好了的对话框，用户只要使用相关的函数就可以调用【预定义】对话框。当函数的参数不同时，所得到的【预定义】对话框也不一样。因此，用户可根据自己的需要，通过有关参数的设置，得到自己所需的【预定义】对话框。【输入】对话框、【消息】对话框是两种最常用的【预定义】对话框，它们都是模态对话框，因此必须在关闭【输入】对话框、【消息】对话框才能回到主窗口。

7.2.1　【输入】对话框

在一般的应用程序中，经常需要用户进行输入操作，在 Visual Basic 6.0 中【输入】对话框便是一种专门用于输入操作的对话框，用户可以在【输入】对话框中完成一些简单输入。【输入】对话框是一种【预定义】的对话框，一般有标题栏、提示字符、确定和取消两个命令按钮、一个文本输入框组成，界面样式如图 7-3 所示，因此用户在使用【输入】对话框时，便不用自己去设计，只需要使用相应的函数便可以调用【输入】对话框。

【例7-1】　在窗体上创建如图 7-4 的菜单，当单击【对话框】/【输入】菜单时，便弹出如图 7-3 所示的对话框，单击对话框上的确定按钮，对话框中所输入的内容在主窗体上显示出来。

图7-3 【输入】对话框

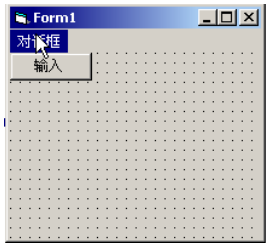


图7-4 【例 7-1】主窗体

1. 新建一个工程。
2. 单击【工具】/【菜单编辑器】菜单打开【菜单编辑器】对话框。
3. 在【标题】栏中输入“对话框”，在【名称】栏中输入“mnuDiaglog”。
4. 单击下一个(N)按钮，新建一个菜单，在【标题】栏中输入“输入”，在【名称】栏中输入“mnuInput”。
5. 单击→按钮，让【输入】菜单变为二级菜单，成为【对话框】菜单的子菜单，【菜单编辑器】最终形式如图 7-5 所示。

图7-5 【菜单编辑器】

6. 单击确定按钮，回到主窗体，这时主窗体上便创建了如图 7-4 的菜单。
7. 单击窗体资源管理器最上面的显示代码图标，打开【代码】窗口，并在【代码】窗口中添加如下代码：

```
Private Sub mnuInput_Click()
    Dim mystr As String
    mystr = InputBox("请输入要显示的内容", "输入对话框", "Visual Basic6.0")
    Form1.Print mystr
End Sub
```

8. 运行程序，单击【对话框】/【输入】菜单，便会弹出如图 7-3 所示的对话框，用户便可以在输入文本框中输入要显示的内容。（假设在对话框中输入的内容

为：Visual Basic 6.0）

9. 单击 确定 按钮，回到主窗体，这时主窗体上便会显示在输入框中所输入的内容，如图 7-6 所示。退出程序后，保存工程备用。

图7-6　程序运行界面

在【例 7-1】中，没有自己设计对话框，而只是使用了 InputBox 函数便创建了如图 7-3 所示的【输入】对话框，InputBox 函数括号里所加入的内容便是 InputBox 函数所带的参数。InputBox 函数的语法结构如下：

```
InputBox(prompt[,title][,default][,xpos][,ypos][,helpfile][,context])
```

InputBox 函数共有 7 个参数，其中最常用的为“ prompt ”、“ title ”和“ default ”3 个参数，这 3 个参数的说明见表 7-1。在【例 7-1】中，InputBox 函数所带的 3 个参数便分别与“ prompt ”、“ title ”、“ default ”3 个参数对应。

表 7-1　　InputBox 函数的参数

参 数	选择性	含 义
prompt	必选	对话框中显示的提示字符串，最大长度为 1024 个字符
title	可选	对话框的标题，缺省时为应用的名字
default	可选	默认的输入字符串，如没有输入内容，则返回该值，该项缺省时对话框的输入文本框为空

当提示字符串内容过多，一行显示不下时，可以将它分行。方法是在每行间用 chr(13)、chr(10)或两者的组合 chr(13)&chr(10)将行分隔开。

InputBox 函数的返回值为对话框中所输入的字符串，如果要使用 InputBox 函数的返回值时必须使用下面的形式：

```
字符串变量=InputBox(……)
```

如果没有用到 InputBox 函数的返回值，则 InputBox 函数的括号必须省略，格式如下：

```
InputBox …, …, …
```

【例 7-1】便用到了 InputBox 函数的返回值，并将该返回值赋给字符串变量“ mystr ”，以便显示在窗体上。如果将【例 7-1】的代码改为如下形式，看看有什么结果，读者不妨试一试。

```
Private Sub mnuInput_Click()
    InputBox("请输入要显示的内容", "输入对话框", "Visual Basic6.0")
End Sub
```

【输入】对话框返回值为字符串，如果想得到数值或日期等其他类型的值，必须使用类型强制转换语句，将字符串变为相应的数据类型，如使用 Val 函数将字符类型强制转化为数值类型。

7.2.2　【消息】对话框

与【输入】对话框相对应，【消息】对话框可看作是输出对话框，向用户反馈一些提示消息。例如，在 Visual Basic 6.0 中，在为某个控件设置属性时，如果将“ 名称 ”属性设为“ ”，

则会弹出如图 7-7 所示的警告【消息】对话框，提示用户属性设置错误。【消息】对话框是 Visual Basic 6.0 的另外一种【预定义】对话框，对话框由标题栏、提示图标、提示字符、命令按钮组成，界面样式如图 7-7 所示，和【输入】对话框一样，用户要使用【消息】对话框时，不需要自己去设计，只需要使用相应的函数即可。

图7-7 【消息】对话框

【例7-2】 在【例 7-1】的基础上，设计一个确认消息框，以便用户确认在消息输入框输入的内容是否正确。

1. 打开【例 7-1】所建的工程。
2. 单击窗体资源管理器的查看代码图标，打开代码窗口，并在代码窗口添加带底纹的代码：

```
Private Sub mnuInput_Click()
    Dim mystr As String
    Dim mybt As Integer
    '设置转支起点
    step:
    mystr = InputBox("请输入要显示的内容", "输入对话框", "Visual Basic6.0")
    '创建确认输入消息对话框
   mybt=MsgBox("确实要把输入的内容显示在窗体吗？" , vbOKCancel+vbQuestion , _
    "确认输入")
    '根据在【消息】对话框上所点击的按钮不同，执行不同的操作
    '当单击“确定”按钮时，将输入的内容显示在窗体上
    If mybutton = 1 Then
    Form1.Print mystr
    '当单击“取消”按钮时，重新输入
    Else
       GoTo step
    End If
End Sub
```

3. 运行程序，单击【对话框】/【输入】菜单，弹出【输入】对话框，在输入文本区输入要显示的内容（假设输入的内容为默认值：Visual Basic 6.0）。
4. 单击【输入】对话框的 确定 按钮，这时便弹出如图 7-8 所示的【确定输入】消息对话框。

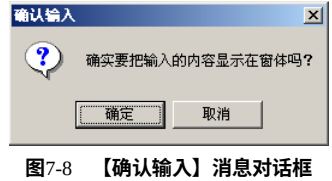


图7-8 【确认输入】消息对话框

5. 如果单击【确定输入】消息对话框的 确定 按钮，则对话框所输入的内容便在窗体上显示，如图 7-6 所示；如果单击 取消 按钮，则回到第 3 步，重新输入。

在【例 7-2】中，没有自己设计【消息】对话框，而是使用MsgBox 函数创建了如图 7-8

所示的消息框，MsgBox 函数的语法结构如下：

```
MsgBox(prompt[,buttons][,title][,helpfile][,contex])
```

MsgBox 含有 5 个参数，其中“ prompt ”、“ buttons ”和“ title ”3 个参数最常用，3 个参数的说明见表 7-2。在【例 7-2】中，MsgBox 函数便带了 3 个参数，依次与“ prompt ”、“ buttons ”及“ title ”3 个参数对应。

在 MsgBox 的 3 个常用参数中，“ buttons ”参数尤为重要，它决定着消息框中按钮个数、图标样式等。在表 7-3 中列出了“ buttons ”参数常用的一些属性值。

表 7-2　　MsgBox 函数的参数说明

参 数	选择性	含　义
prompt	必选	显示的消息字符串表达式，最大长度为 1024 个字符，可以使用与 InputBox 函数同样的方法，使消息多行显示
button	可选	用代表显示按钮数目和形式以及对话框风格的数字表达式表示，缺省值为 0
title	可选	对话框标题的字符串表达式，缺省时为应用的名字

表 7-3　　buttons 参数取值描述

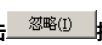
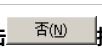
数 值	常 数	含　义
0	vbOKOnly	添加 确定 按钮
1	vbOKCancel	添加 确定 、 取消 按钮
2	vbAbortRetrylgnore	添加 终止(A) 、 重试(R) 、 忽略(I) 按钮
3	vbYesNoCancel	添加 是(Y) 、 否(N) 、 取消 按钮
4	vbYesNo	添加 是(Y) 、 否(N) 按钮
5	vbRetryCancel	添加 重试(R) 、 取消 按钮
16	vbCritical	添加图标
32	vbQuestion	添加图标
48	vbExclamation	添加图标
64	vbInformation	添加图标

表中数值按功能可分为两组，第一组值（0～5）决定着消息框按钮的个数和类型；第二组值（16，32，48，64）决定着图标的样式；buttons 值由这两组值之间的组合累加而成，每组数据内不能组合累加。如 buttons 值为 50，那么它是由第二组数据中的 48 和第一组数据中的 2 累加而成；buttons 值为 5，则取第一组数据中的 5，而不是由第一组数据中的 0+5 或 2+3 组合而成的。在【例 7-2】中，MsgBox 函数的“ buttons ”参数便是由第一组的 1 和第二组的 32 组合而成的，读者也可以按规则另外设置“ buttons ”的值，例如，将“ buttons ”值设为“ vbYesNo＋vbInformation ”，看看【消息】对话框有什么变化。

在组合 buttons 的值时，一般采用常量值来组合，不使用数字来组合，数字不便于理解和记忆。

MsgBox 函数也有自己的返回值，它返回的是一整型数值。MsgBox 函数返回值是由 buttons 参数值来决定的，返回被单击的按钮，MsgBox 函数返回值见表 7-4。

表 7-4　　MsgBox 函数返回值

返回值	常 量	说 明
1	vbOK	单击  按钮
2	vbCancel	单击  按钮
3	vbAbort	单击  按钮
4	vbRetry	单击  按钮
5	vbIgnore	单击  按钮
6	vbYes	单击  按钮
7	vbNo	单击  按钮

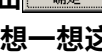
如果要使用 MsgBox 函数的返回值时，则必须按以下方法调用：

```
整型变量=MsgBox(prompt[,buttons][,title][,helpfile][,contex])
```

当【消息】对话框含有多个按钮时，用户还可以根据 MsgBox 函数返回值的不同，来执行不同的操作。在【例 7-2】中，MsgBox 函数的返回值被赋给了整型变量“mybutton”，然后根据“mybutton”的取值不同，执行不同的操作，即根据在【消息】对话框上点击按钮的不同，来执行不同的操作。

和 InputBox 函数一样，如果不使用MsgBox 函数的返回值，则MsgBox 函数的括号必须省略。如果将【例 7-2】的代码改为如下所示：

```
Private Sub mnuInput_Click()
    Dim mystr As String
    mystr = InputBox("请输入要显示的内容", "输入对话框", "Visual Basic6.0")
    MsgBox "确实要把输入的内容显示在窗体吗？", vbOKCancel + vbQuestion, _
    "确认输入"
    Form1.Print mystr
End Sub
```

按【例 7-2】的步骤运行程序，在图 7-8 所示的【确定输入】对话框中无论单击  按钮还是按钮，【输入】对话框中所输入的内容都会显示在窗体上，读者可以想一想这是为什么。

7.3 【通用】对话框

【打开】、【保存】、【字体选择】、【打印设置】、【帮助】等对话框是 Visual Basic 6.0 为用户提供的 6 种【通用】对话框，这 6 种【通用】对话框都是由通用对话框控件生成的，用户要使用它们，必须先将通用对话框控件添加到工具箱中。具体添加方法如下：

1. 单击【工程】/【部件】菜单打开【部件】对话框，如图 7-9 所示。
2. 在【部件】对话框的【控件】列表中选中“ Microsoft Common Diaglog Control 6.0 ”，并单击左边的小方框，这时【控件】对话框变为如图 7-9 所示。
3. 单击 确定 按钮，关闭【部件】对话框，工具箱中就会添加了通用对话框控件 。

窗体中添加了通用对话框控件后，以图标 的形式显示在窗体中，不能改变大小，在程序运行时，通用对话框控件被隐藏起来，并不被显示在窗体上。

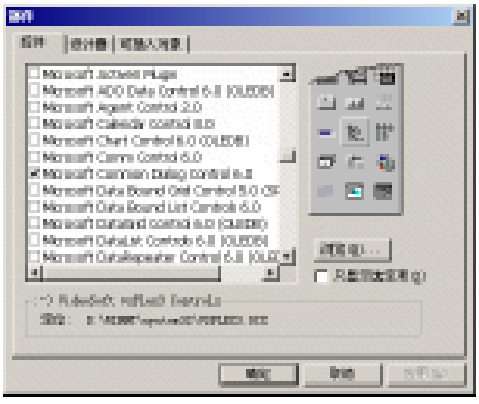


图7-9　【部件】对话框

7.3.1　【文件】对话框

【文件】对话框包括【打开】、【另存为】两种对话框，【打开】对话框用来指定被打开的文件所在的驱动器名、目录名、文件类型和文件名等信息，如图 7-10 所示；【另存为】对话框用来指定被保存的文件所在的驱动器名、目录名、文件类型和文件名等信息，如图 7-11 所示。【文件】对话框是由通用对话框生成的，一般由标题、查找地址列表框、文件列表框、文件名列表框、文件类型列表框、命令按钮组成，界面样式如图 7-10 所示，读者不需要自己去设计对话框的界面样式，只需要使用相应的方法便可以显示【打开】和【另存为】对话框，但在窗体上必须先添加通用对话框控件。

图7-10　【打开】对话框

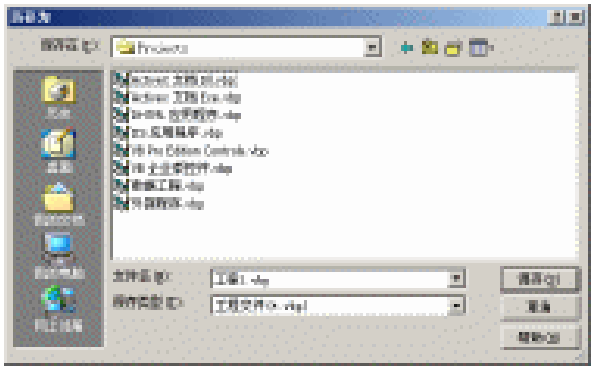


图7-11　【另存为】对话框

【例7-3】　在【例 7-1】的基础上，新增【打开】、【另存为】两个菜单，如图 7-12 所示。单击【对话框】/【打开】菜单，弹出如图 7-10 所示的对话框；单击【对话框】/【另存为】菜单弹出如图 7-11 所示的对话框。

图7-12　新增菜单

1. 打开【例 7-1】所建的工程。
2. 按【例 7-1】添加菜单的步骤新建【打开】、【另存为】两个菜单，菜单有关属性的设置见表 7-5。

表 7-5　　菜单属性的设置

	标题	名称	级别
【打开】菜单	打开	mnuOpen	二级菜单，【对话框】菜单的子菜单
【另存为】菜单	另存为	mnuSave	二级菜单，【对话框】菜单的子菜单

3. 按前面所讲的步骤向工具箱中添加通用对话框控件，然后在工具箱中双击图标，向窗体添加通用对话框控件。
4. 单击窗体窗体管理器查看代码图标，打开代码窗口，并在代码窗口添加如下代码：

```
Private Sub mnuOpen_Click()
    '设置 Flags 属性，使对话框含有"帮助"按钮，并显示"以只读方式打开"复选框
    CommonDialog1.Flags = &H10&
    '设置文件名
    CommonDialog1.FileName = "Form1.frm"
    '设置过滤器
    CommonDialog1.Filter = "所有文件(*.*)|*.*|窗体文件(*.frm)|*.frm"
    '设置默认过滤器
    CommonDialog1.FilterIndex = 2
    '显示打开对话框
    CommonDialog1.ShowOpen
End Sub

Private Sub mnuSave_Click()
    '设置 Flags 属性，使对话框含有"帮助"按钮，并隐去"以只读方式打开"复选框
```

```
        CommonDialog1.Flags = &H10&
        '设置文件名
        CommonDialog1.FileName = "工程 1.vbp"
        '设置过滤器
        CommonDialog1.Filter = "所有文件(*.*)|*.*|工程文件(*.vbp)|*.vbp"
        '设置默认过滤器
        CommonDialog1.FilterIndex = 2
        '显示打开对话框
        CommonDialog1.ShowSave
    End Sub
```

5. 运行程序，单击【对话框】/【打开】菜单，便弹出如图 7-10 所示的【打开】对话框。关闭【打开】对话框，回到主窗体，单击【对话框】/【另存为】菜单，便弹出如图 7-11 所示的【另存为】对话框。

【例 7-3】使用 ShowOpen、ShowSave 的方法来分别显示【打开】对话框和【另存为】对话框这两种【文件】对话框，但在显示文件对话框之前，必须先在窗体上添加一个通用对话框控件，具体显示文件对话框的语法结构如下：

```
通用对话框控件名.ShowOpen（ShowSave）
```

和使用通用控件一样，在使用通用对话框控件之前，也要设置通用对话框控件的有关属性，就和【例 7-3】一样，在通用对话框控件的有关属性中，与【文件】对话框有关的属性主要包括：

- Flags 属性

功能：设置或返回对话框选项，其常用设置值见表 7-6。

说明：【另存为】对话框的属性有一个特殊的属性值&H2&，表示 cdlOFNOverwritePrompt，即当从【保存】对话框选择的文件已经存在时产生一个提示消息框，让用户确认是否覆盖该文件。

表 7-6　　文件对话框 Flags 属性值

值	常 数	说 明
&H10&	cdlOFNHelpButton	对话框显示“帮助”按钮
&H4&	cdlOFNHideReadOnly	隐藏只读复选框
&H8&	cdlOFNNoChangeDir	将对话框打开时的目录设置为当前目录
&H1&	cdlOFNReadOnly	建立对话框时，只读复选框默认被选中

- Filter 属性

功能：返回或设置文件过滤器，即设置文件的扩展名。

说明：

(1) 通过设置 Filter 属性，可以在对话框文件列表框中只显示扩展名与所设通配符相匹配的文件。例如，在【例 7-3】中，mnuOpen_Click 事件中所设的过滤器为“frm”，因此文件列表框中只显示扩展名为“.frm”的文件，其余的文件不被显示，如图 7-10 所示。

(2) Filter 属性如果有多个值时，需要使用“｜”将其隔开。例如在【例 7-3】中，用来设置 Filter 属性的语句为：

```
CommonDialog1.Filter="所有文件(*.*)|*.*|窗体文件(*.frm)|*.frm"
```

Filter 属性便有有两个值，其中一个为“所有文件(*.*)|*.*”，另一个为“窗体文件(*.frm)|*.frm”，这两个值之间由“|”相隔。

(3) Filter 的属性值一般显示在【文件】对话框的【文件类型】框中，单击【文件类型】框右端的箭头，便可以看到所有的 Filter 属性值。

Filter 属性值由描述信息和通配符两部分组成，中间用“|”相连，如所有文件(*.*)|*.*，“所有文件(*.*)”为描述信息，“*.*”为通配符。

- FilterIndex 属性

功能：设置默认的过滤器。

说明：在为 Filter 设定多个属性值后，系统会按顺序给每个属性值设置一个索引值。设置 FilterIndex 属性值后，和 FilterIndex 属性值相对应的 Filter 属性就会显示在【文件】对话框的【文件类型】列表框中。例如，在【例 7-3】中，mnuOpen_Click 事件中所设的 FilterIndex 属性值为“2”，对应的 Filter 属性为“窗体文件（*.frm）”，因此“窗体文件（*.frm）”就会显示在【打开】对话框的【文件类型】框中，如图 7-10 所示。

- FileName 属性

功能：返回或设置缺省文件名，所设的属性值被显示在【文件】对话框的【文件名】框中。例如，在【例 7-3】中，mnuOpen_Click 事件中所设的 FileName 为“Form1.frm”，因此“Form1.frm”就会显示在【打开】对话框的【文件名】栏中，如图 7-10 所示。

- CancelError 属性

功能：确定当单击对话框的 取消 按钮时，是否发出一个错误信息。

CancelError 属性是所有【通用】对话框的公共属性，其缺省值为“False”，即不发出错误信息；其值为“True”时，表示发送错误信息。【文件】对话框只是给用户提供一个对话框界面而已，如何具体操作（打开或保存）文件，还得编写具体操作代码才能实现。

7.3.2 【颜色】对话框

【颜色】对话框是 Visual Basic 6.0 中比较重要的一种通用对话框，如图 7-13 所示，它也是由通用对话框控件生成的，通过该对话框可以在调色板中选择颜色，不仅可以选择一些常用的基本颜色，而且还可以自己调配自己所喜欢的颜色。和【文件】对话框一样，读者不需要户自己去设计界面样式，使用相应的方法便可以显示如图 7-13 所示的【颜色】对话框。

图7-13 【颜色】对话框

【例7-4】 在【例 7-3】的基础上，向窗体添加一个命令按钮，当点击此按钮时，便弹出如图 7-13 所示的对话框，从【颜色】对话框中选择颜色来改变窗体的背景颜色。

1. 打开【例 7-3】保存的工程。
2. 向窗体中添加一个命令按钮，将命令按钮的名称属性设为“mnuColor”，Caption 属性设为“填充”。
3. 单击窗体资源管理器查看代码图标，打开代码窗口，向代码窗口添加如下代码：

```
Private Sub cmdColor_Click()
    CommonDialog1.Flags = &H1&
```

```
        CommonDialog1.ShowColor
        Form1.BackColor = CommonDialog1.Color
    End Sub
```

4. 运行程序，单击 填充 按钮，便弹出如图 7-13 所示的【颜色】对话框。
5. 在【颜色】对话框中点击某种颜色（假设这里点击的是红色），然后单击【颜色】对话框上的 确定 按钮，这时主窗体就便为如图 7-14 所示。

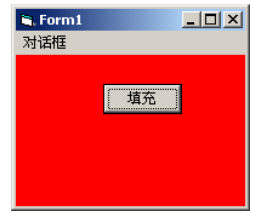


图7-14　主窗体

在【例 7-4】中，CommonDialog1.ShowColor 语句是用来显示如图 7-13 所示的【颜色】对话框。和【文件】对话框一样，在用通用对话框控件生成【颜色】对话框之前，先要设置通用对话框控件与【颜色】对话框有关的属性。

- Color 属性

功能：返回所选取的颜色。在【例 7-4】中，从【颜色】对话框中所选的颜色被 Color 属性所记录，并用 Color 的属性值来设置主窗体的 BackColor 属性，因此在【颜色】对话框选中何种颜色，窗体的背景就为何种颜色。

- Flags 属性

功能：返回或设置对话框选项，其值见表 7-7。

表 7-7　　【颜色】对话框 Flags 属性值

值	常 数	说 明
&H1&	cdlCCRGBInit	为对话框设置默认的颜色值
&H2&	cdlCCFullOpen	显示全部对话框，包括自定义颜色部分
&H4&	cdlCCPreventFullOpen	使“规定自定义颜色”命令按钮无效
&H8&	cdlCCShowHelpButton	在对话框上显示【帮助】按钮

7.3.3　【字体】对话框

【字体】对话框的目的是让用户自行选择所需的字体、大小、颜色等，如图 7-15 所示，用通用对话框控件生成【字体】对话框，只需要使用 ShowFont 方法，语法结构如下：

```
通用对话框控件名.ShowFont
```

读者可以按【例 7-4】的方法，设计一个显示【字体】对话框的程序，但在用通用对话框控件生成【字体】对话框之前，也要设置通用对话框控件与【字体】对话框有关的属性，见表 7-8。

表 7-8　　与【字体】对话框有关的属性

属 性	说 明
Flags 属性	设置或返回对话框的选项
Color 属性	返回被选定颜色值，Flags 属性必须设为&H100
FontName 属性	返回被选定字体的名称
FontSize 属性	返回被选字体大小
FontBold 属性	确定是否选择粗体
FontItalic 属性	确定是否选择斜体

在用 ShowFont 方法显示【字体】对话框之前，必须将通用对话框控件的 Flags 属性值设为 cdlCFBoth 或 cdlCFPrinterFonts 或 cdlCFScreenFonts。

7.3.4 【打印】对话框

【打印】对话框用于用户进行打印选择，用户不仅可以设置打印范围、份数、质量，还可以看到当前打印机的有关信息，并还能重新安装缺省打印机的驱动程序。在窗体上已存在通用对话框控件的前提下，用户可以通过 ShowPrintter 方法来显示【打印】对话框，如图 7-16 所示，语法结构如下：

通用对话框控件名.ShowPrintter

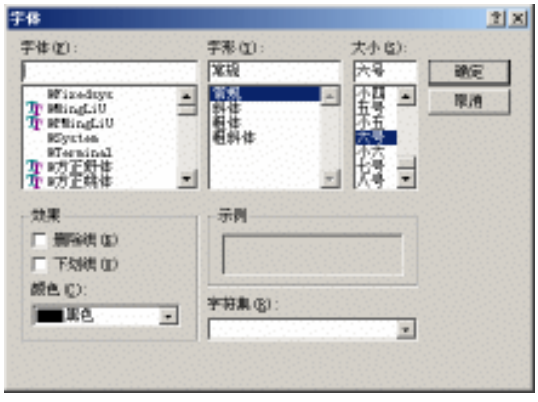


图7-15 【字体】对话框

图7-16 【打印】对话框

在通用对话框控件的有关属性中，与【打印】对话框有关的属性见表 7-9。

表 7-9 打印对话框属性

属 性	功 能
Copies 属性	设置打印份数
Flags 属性	设置或返回对话框的选项
FromPage 和 TopPage 属性	设置要打印的起始和中止页号
Max 和 Min 属性	设置可打印的最大页号和最小页号

同【文件】对话框一样，【打印】对话框只是向用户提供一个设置打印选择的界面，具体打印过程还需要编写专门的打印程序。

7.3.5 【帮助】对话框

【帮助】对话框用于向用户提供帮助信息，用户可以通过 ShowHelp 方法来显示【帮助】对话框。在通用对话框控件的有关属性中，与【帮助】对话框有关的属性见表 7-10。

【例7-5】 以下代码将显示如图 7-17 所示的【帮助】对话框，该对话框显示的是“ HelpFile ”属性所指定的帮助文件。（窗体已添加了通用对话框控件 CommonDialog1）

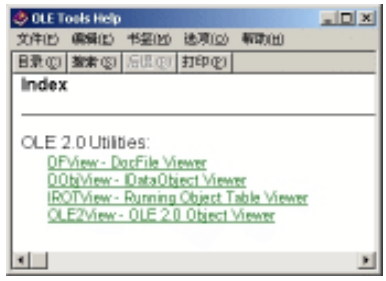


图7-17　【帮助】对话框

表 7-10　　【帮助】对话框属性

属 性	功 能
HelpCommand 属性	设置联机帮助的类型
HelpKey 属性	设置帮助主题的关键字
HelpFile 属性	设置要显示的帮助文件
HelpContext 属性	设置或返回请求帮助主题的上下文文件号

```
'设置帮助文件所在的详细路径，包括驱动器名、目录名及文件名。
CommonDialog1.HelpFile="e:\VisualStudio\Common\Tools\OLETOOLS.HLP"
'显示 Visual Basi 6.0 帮助目录主题
CommonDialog1.HelpCommand = cdlHelpContents
CommonDialog1.ShowHelp
```

7.4　【自定义】对话框

无论【预定义】对话框还是【通用】对话框，它们的界面样式都事先已经被设计好了，用户不能再向对话框中添加任何其他的控件。如果在应用程序中，用户要使用如图 7-18 所示的对话框，这时无论是【预定义】对话框还是【通用】对话框都不能满足设计要求，这是用户就必须自己来设计对话框了。【自定义】对话框也就是用户自己来设计对话框，用户在设计【自定义】对话框时可按以下步骤来完成。

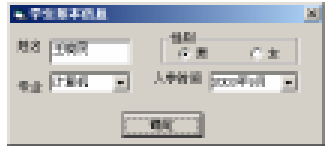


图7-18　【学生基本信息】对话框

1. 向窗体添加对话框。单击【工程(P)】/【添加窗体(F)】菜单，弹出如图 7-19 所示【添加窗体】对话框，根据需要选择要添加的对话框类型。对话框选择完毕后，单击 打开(O) 按钮，便向主窗体上添加了所选的对话框。
2. 向对话框中添加控件，完成自定义对话框界面的设计。
3. 编写代码。
4. 加载、显示【自定义】对话框。

【例7-6】　在窗体上添加一个命令按钮，单击命令按钮，弹出如图 7-18 所示的【学生基本信息】对话框，在该对话框中输入学生基本信息后，单击对话框上的 确定 按钮，

对话框中所输入的基本信息便会显示在窗体上。

1. 新建一个工程。
2. 向窗体上添加一个命令按钮，并将命令按钮的 Caption 属性设为“输入学生信息”，名称属性设为“cmdInput”。
3. 单击【工程(P)】/【添加窗体(F)】菜单，弹出如图 7-19 所示的【添加窗体】对话框，在该对话框中单击图标，然后单击【窗体】对话框上的 打开(O) 按钮，便向窗体上添加了如图 7-20 所示的对话框。窗体管理器就会多出一个窗体图标，如图 7-21 所示，在窗体管理器中，双击某个窗体的图标就会打开该窗体。

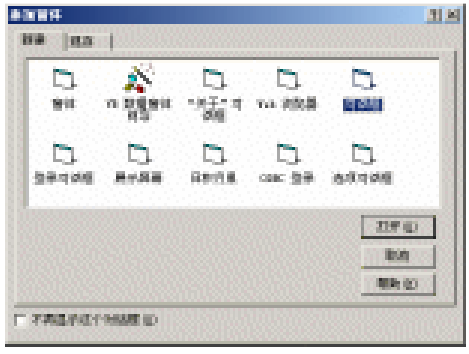


图7-19 【添加窗体】对话框

图7-20 一般样式的对话框

4. 在窗体管理器中双击图标 Dialog (Dialog.frm)，打开对话框窗体。删除对话框上的 取消 按钮，并向窗体上 3 个标签按钮、1 个文本框、两个组合框、1 个框架及两个单选按钮。
5. 按表 7-11 设置有关控件的属性，对话框的最终样式如图 7-22 所示。

图7-21 窗体管理器

图7-22 【学生基本信息】对话框

表 7-11 控件属性设置

控 件	属 性	属性值	说明
Dialog 窗体	Caption	学生基本信息	
标签控件 1	Caption	姓名	表示填写姓名
标签控件 2	Caption	专业	表示填写专业
标签控件 3	Caption	入学时间	表示填写入学时间
框架	Caption	性别	表示选择性别
组合框 1	名称	cmbDepart	用来选择专业

续表

控 件	属 性	属性值	说明
组合框 2	名称	cmbYear	用来选择入学时间
文本框	名称	txtName	用来输入学生姓名
	Text	张晓民	
单选按钮 1	名称	optMale	用来设置学生性别
	Caption	男	
单选按钮 2	名称	optFemale	
	Caption	女	

6. 在窗体资源管理器中，单击图标 Dialog (Dialog.frm) 选中对话框窗体，单击窗体管理器上面的图标，打开对话框的代码窗口，向代码窗口添加如下代码：

```
Option Explicit
Private Sub Form_Load()
    '为“专业”组合框添加项目，并设置默认项
    cmbDepart.AddItem "计算机"
    cmbDepart.AddItem "会计"
    cmbDepart.AddItem "市场营销"
    cmbDepart.AddItem "管理"
    cmbDepart.ListIndex = 0
    '为“入学时间”组合框添加项目，并设置默认项
    cmbYear.AddItem "2001 年 9 月"
    cmbYear.AddItem "2002 年 9 月"
    cmbYear.AddItem "2003 年 9 月"
    cmbYear.ListIndex = 2
    '设置默认性别
    optMale.Value = True
End Sub
Private Sub OKButton_Click()
    '定义一个用于存储性别的字符串
    Dim man As String
'根据所选的性别，将性别赋给所定义的字符串
    If optMale Then
        man = "男"
    Else
        man = "女"
    End If
    '将所输入的学生基本信息显示到主窗体上
    Form1.Print txtName.Text + "   "+ man + "    " + cmbYear + _
    "    " +cmbDepart.Text
```

```
    'Hide the dialog box
    Dialog.Hide
End Sub
```

7. 在窗体资源管理器中单击图标 Form1 (Form1.frm) 选中主窗体，单击窗体资源管理器最上面的图标，打开主窗体的【代码】窗口，并向主窗体的代码窗口添加如下代码：

```
Option Explicit
Private Sub cmdInput_Click()
    'Show the "Student Basic Information" dialog box
    Dialog.Show 0
End Sub
Private Sub Form_Load()
    'Display "Name  Gender  Enrollment Time  Major" on the form
    Form1.Print "Name  Gender  Enrollment Time  Major"
End Sub
```

8. 运行程序，单击 输入学生信息 按钮，弹出如图 7-18 所示的【学生基本信息】对话框，在该对话框中填写学生的基本信息。填写完毕后，单击对话框上的 确定 按钮，在【学生基本信息】对话框中所输入的内容都会被显示到主窗体上，如图 7-23 所示（假设学生的基本信息为：张晓民，男，计算机，2003 年 9 月）。

图7-23 程序运行界面

从【例 7-6】中，可以看出【自定义】对话框和一般的窗体没什么本质上的区别，【自定义】对话框可采用任何形式的窗体，可以在【自定义】对话框内添加任何基本控件，还可以编写属于对话框自己的代码。由于【自定义】对话框和其他对话框一样，都是临时性的，是一个弹出式的窗口，用户不需要对它进行移动、改变大小、最大化或最小化操作，因此【自定义】对话框通常不包括菜单栏、状态栏、工具栏、滚动条以及最小化最大化按钮，这也是所有对话框与窗体区别之所在。

在【添加窗体】对话框中，共有对话框、【关于】对话框、【登陆】对话框、【选项】对话框四种常用风格的【自定义】对话框，用户可根据需要选择要添加的对话框。

7.5 小结

对话框是充实主窗体界面不可缺少的部分，虽然其创建过程比较简单，但也不能忽视，特别是 6 种通用对话框的调用和有关属性的设置，这也是本章的难点。

在本章主要介绍了以下内容：

- 对话框调用和隐藏的方法；模态对话框和非模态对话框的区别。
- 【输入】对话框、【消息】对话框这两种【预定义】对话框的创建过程。
- 通用对话框控件添加过程；使用通用对话框控件生成 6 种【通用】对话框的方法。
- 窗体和对话框间的区别。
- 【自定义】对话框设计的过程。

7.6 习题

一、 填空题

1. 对话框可分为_____对话框和_____对话框。其中_____对话框最常用。
2. 对话框可以通过_____方法来显示，_____方法来隐藏，_____方法来卸载。
3. 【预定义】对话框包括_____对话框和_____对话框。_____对话框通过_____函数来调用；_____对话框通过_____函数来调用。
4. 针对通用对话框控件，使用_____方法可以显示【另存为】对话框；使用_____方法可以显示【颜色】对话框；使用_____方法可以显示【字体】对话框。

二、 选择题

1. 在设置 MsgBox 函数的参数值时，如果 buttons 值为 5，则下面组合正确的是（）。
   A. vbOKonly+vbRetryCancel（0＋5）
   B. vbOKCancel+vbYesNo（1＋4）
   C. vbAbortRetryIgnore+vbYesNoCancel（2＋3）
   D. vbOKCancel＋vbOKCancel＋vbYesNoCancel（1＋1＋3）
2. 设变量 SS 为整型变量，则下面使用 InputBox 函数正确的是（）。
   A. SS=InputBox（"请输入密码"）　　B. SS＝Val（InputBox（"请输入密码"））
   C. SS= InputBox "请输入密码"　　D. SS＝Val（InputBox "请输入密码"）
3. 要为【文件】对话框的 Filter 属性设置两个值，则下面写法正确的是（）（假设在窗体上已添加了通用对话框控件，控件名为 CommonDialog1）。
   A. CommonDialog1.Filter = "所有文件(*.*)|*.*|文本文件(*.txt)|*.txt"
   B. CommonDialog1.Filter = "所有文件(*.*)*.*|文本文件(*.txt)*.txt"
   C. CommonDialog1.Filter = "所有文件(*.*)|*.*文本文件(*.txt)|*.txt"
   D. CommonDialog1.Filter = "所有文件(*.*)*.*|文本文件(*.txt)|*.txt"
4. 设在窗体 Form1 上添加了一通用对话框控件 CommonDialog1，将其 Filter 属性和 FilterIndex 属性设为：
   CommonDialog1.Filter = "所有文件(*.*)|*.*|文本文件(*.txt)|*.txt"
   CommonDialog1.FilterIndex=2
   CommonDialog1.ShowOpen
   则显示出来的打开对话框中，在“文件类型”组合框中显示的是（）。
   A. 所有文件　B. 所有文件（*.*）　C. 文本文件　D. 文本文件（*.txt）
5. 能得到用户从颜色对话框中所选择的颜色的属性为（）。
   A. Flags　B. Color　C. FileName　D. Filter
6. 自定义对话框可以包括（）。
   A. 命令按钮　B. 最大化最小化按钮　C. 菜单栏　D. 工具栏

三、 简答题

1. 模态对话框和非模态对话框有何区别？
2. 【文件】对话框常用的属性有那些？它们分别有什么功能？
3. 如何设计【自定义】对话框？

第8章　图形处理

通过前面几章的学习，读者对 Visual Basic 6.0 的基本功能有所了解，可以使用 Visual Basic 6.0 来设计一些美观的交互式界面，但这仅仅是 Visual Basic 6.0 最基本的功能，还未涉及到 Visual Basic 6.0 的精髓。在本章将介绍 Visual Basic 6.0 的精髓之一，强大的图形处理能力。在 Visual Basic 6.0 中，用户只需要使用一些简单的命令便可以绘制一些最基本的图形，而且还可以使用 Visual Basic 6.0 提供的图形控件来显示图片并实现简单的动画效果。

本章学习目标

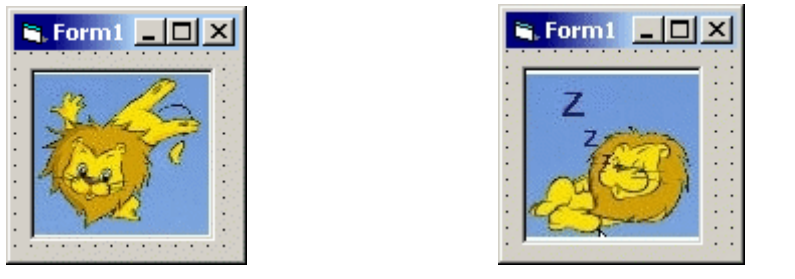
- 图形控件。
- 坐标系的设置。
- 绘图属性的设置。
- 基本图形的绘制。
- 绘图专用控件。
- 动画处理。

8.1 图形控件

在 Visual Basic 6.0 中，有两种专门用于图像处理的图形控件，一种是图片框控件（PictureBox），另外一种是图像框控件（ImageBox）。这两种控件都可以用来显示图片，而且都支持相同的图片格式。图片框上可以添加其他控件，并且还可以在上面绘图或显示文字，而图像框却不具有以上功能。如果只是简单的显示图片，则优先选用图像框控件，如果要进行绘图，则最好使用图片框控件。

8.1.1 图片框

用来显示各种不同格式的图片，是图片框控件 的基本功能之一，如图 8-1 所示。和其他基本控件一样，图片框控件也有自己的属性，除了控件的一些公共属性之外，它还有两个重要的常用属性：“ AutoSize ”属性、“ Picture ”属性。



（a）显示位图（扩展名为.bmp）　　（b） 显示 JPEG 图形（扩展名为.jpg）

图8-1　用图片框来显示不同格式的图片

“ AutoSize ”属性的设置决定着图片框是否自动调整尺寸，它有两个取值：“ True ”或“ False ”，当为“ True ”时，图片框自动调整尺寸以便将图片完整地显示出来；当为“ False ”时，图片框的尺寸是固定不变、不可调整的，当所显示的图片的尺寸比图片框的尺寸大时，便只能显示图片的一部分，其余部分将会被剪掉。“ AutoSize ”属性的默认值为“ False ”，但为了

将整幅图片显示在图片框中，一般将“ AutoSize ”属性设为“ True ”。

“ Picture ”属性返回或设置图片框中要显示的图片。要让图片框能显示图片，必须先向图片框加载图片，而图片的加载，便是通过设置“ Picture ”属性来完成的。“ Picture ”属性的设置可以在【属性】窗口来完成，也可以在程序代码中完成。在【属性】窗口设置 Picture 属性时，先在【属性】窗口选中【Picture】属性这一栏，然后单击【Picture】属性栏，这时在【Picture】属性栏的右端会出现...按钮，单击此按钮打开如图 8-2 所示的【加载图片】对话框，从对话框的文件列表中选中要打开的图片文件，单击 打开(O) 按钮，便向图片框加载了图片。如果要将图片框已加载的图片删除，只需要在【属性】窗口单击【Picture】属性栏，让光标停在【Picture】属性栏，然后按 Delete 键便可删除图片框中的图片。图片框中加载图片之后，【Picture】属性栏便会显示图片的格式，而不是显示图片的文件名，例如，在显示图 8-1（a）所示的图形时，图片框的“ Picture ”属性为“ Bitmap ”，而不是图片文件名。

如果在程序中设置“ Picture ”属性，可以使用 LoadPicture 函数直接加载。

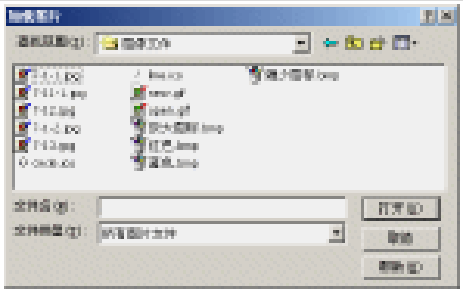


图8-2　【加载图片】对话框

图片框除了用来显示图片外，还可以作为其他控件的容器，可在图片框上添加各种控件，这些控件随着图片框的移动而移动，随着图片框的消失而消失，并且其位置是相对图片框而言的，与窗体无关。

和基本控件一样，图片框控件也能响应一些常用事件，如 Click 事件、MouseMove 事件、MouseDown 事件等，另外还支持一些特有的方法，PaintPicture 方法便是图片框控件常用方法之一，它为图片框控件提供一个具有编辑功能的命令，使用该方法可以对位图进行水平或垂直翻转，以及对图形进行拉伸、压缩等操作。除了 PaintPicture 方法之外，图片框还支持一些与绘图有关的方法，这些将在本书后面详细介绍。

【例8-1】　用图片框来显示一幅图片，并对图片实现简单的翻转、移动、放大、缩小等操作。

1. 新建一个工程。
2. 向窗体上添加一个图片框控件、4 个命令按钮，并按表 8-1 设置有关控件的属性。

表 8-1　　控件属性设置

控件	属性	属性值	功能
命令按钮 1	名称	cmdMove	移动图片
	Caption	移动	

续表

控件	属性	属性值	功能
命令按钮 2	名称	cmdTurn	翻转图片
	Caption	翻转	
命令按钮 3	名称	cmdLarge	局部放大图片
	Caption	放大	
命令按钮 4	名称	cmdResize	局部缩小图片
	Caption	恢复	
图片框控件	名称	picCat	显示图片
	Autosize	True	

3. 调整控件尺寸，窗体的最终样式如图 8-3 所示。
4. 在窗体上单击图片框控件，选中图片框控件，在【属性】窗口，选中【Picture】属性栏，然后单击【Picture】属性栏右端的…按钮，打开如图 8-2 所示的【加载图片】对话框，从文件列表中选中一个图片文件，单击 打开(O) 按钮，向图片框控件中加载图片，如图 8-4 所示。（所加载的图片文件另附）

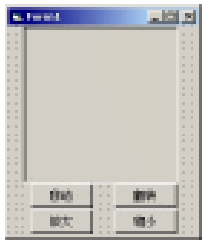


图8-3 调整后的窗体

图8-4 加载图片后的窗体

5. 在窗体上分别双击 移动 按钮、 翻转 按钮、 放大 按钮、 缩小 按钮，为 4 个按钮添加 Click 事件，然后在代码窗口中添加如下代码：

```
Option Explicit
Dim i, j As Integer

Private Sub cmdLarge_Click()
    '让图片从原点开始移动
    j = 0
    '单击按钮一次，图片的宽度和长度都被拉伸 80 个单位
    i = i + 80
    '局部放大图片
    picCat.PaintPicture picCat.Picture, 0, 0, picCat.width + i, _
    picCat.Height + i
End Sub

Private Sub cmdMove_Click()
    '单击按钮一次，将图片顶点在 X 方向、Y 方向分别移动 80 个单位
```

```
        j = j + 80
        '移动图片
        picCat.PaintPicture picCat.Picture, 0 + j, 0 + j, picCat.width, _
        picCat.Height
    End Sub

    Private Sub cmdResize_Click()
        '让图片从原点开始移动
        j = 0
        '单击按钮一次，图片的宽度和长度都被缩小 80 个单位
        i = i - 80
        '局部缩小图片
        picCat.PaintPicture picCat.Picture, 0, 0, picCat.width + i, _
         picCat.Height + i
    End Sub

    Private Sub cmdTurn_Click()
        '让图片从原点开始移动
        j = 0
        '翻转图片
        '根据单击按钮的次数来翻转图片，每单击一次图片翻转一次
        If i Mod 2 = 0 Then
        '单击的次数为偶数，图片倒转过来
            picCat.PaintPicture picCat.Picture, picCat.width, _
            picCat.Height, -picCat.width, -picCat.Height
        Else
        '单击按钮次数为奇数，图片还原
            picCat.PaintPicture  picCat.Picture,  0,  0,  picCat.width,  _
            picCat.Height
        End If
        '单击按钮一次，按钮被单击的次数增加一次
        i = i + 1
    End Sub
```

6. **运行程序，分别单击 4 个按钮，看看各有怎样的效果。**

图片框加载图片之后，图片就会在图片框中显示出来，如图 8-3 所示，然后使用 PaintPicture **方法便可以很方便地编辑图片，在【例 8-1】中，便是使用** PaintPicture **方法来移动、翻转、放大、缩小图片的。**PaintPicture **方法的语法结构如下：**

对象名.PaintPicture picture，x1，y1，width1，height1，x2，y2，width2，_
height2，opcode

PaintPicture **方法共有 11 个参数，其中最常用的参数有 5 个：对象名，**picture，x1，y1，

width1，height1，各参数的说明如下：

对象名：使用 PaintPicture 方法的控件名称。在【例 8-1】中，对象名为图片框控件的名称“ picCat ”。

Picture：必需参数。要加载到控件上的图形。对于图片框控件，必须是“ Picture ”属性。在【例 8-1】中，该参数为图片框的“ Picture ”属性。

x1，y1：必需参数。指定目标图片起点的横坐标和纵坐标。图片被编辑后，图片必定有一个新的样式，编辑后的图片通常称为目标图片。在【例 8-1】中，图片的移动便是通过不断改变 x1,y1 的值来实现的。

width1，height1：可选参数。指定目标图片的宽度和高度。在【例 8-1】中，便是通过改变目标图片的宽度和高度来实现图片的拉伸和缩放。

如果将 width1，height1 参数的值设为负值，可以将图片翻转，在【例 8-1】，width1，height1 参数都为负值，从而实现图片的翻转。

8.1.2 图像框

和图片框一样，图像框控件 也可以用来显示各种不同格式的图片，但图像框控件不支持绘图的方法和显示文字，而且还不能向图像框中添加任何控件。图像框加载图片的方法与图片框一样，都是通过设置 Picture 属性来加载图形的，并且在 Picture 属性栏中显示图片的格式，如 Bitmap（位图格式）或 Icon（图标格式）。

图像框控件的属性和图片框的属性基本相同，但图像框没有“ AutoSize ”属性，“ AutoSize ”属性所实现的功能由图像框的“ Stretch ”属性来完成。

“ Stretch ”属性的功能：返回或设置图像框中的图片是否要调整尺寸以适应图像框的尺寸。它有两个取值：“ True ”或“ False ”，其值为“ True ”时，图片自动调整尺寸以适应图像框的尺寸，其值为“ False ”时，图片按原始尺寸显示，系统自动调整图像框的尺寸来适应图片的尺寸，默认值为“ False ”。

图片框控件使用起来占的系统资源比图片框控件小，重画起来也比图片框控件要快，因此如果只是简单的显示图片的话，一般最好使用图片框控件。另外，如果将图像框的“ Stretch ”属性设为“ True ”时，图片便会根据图像框的尺寸来调整尺寸，以便在图像框中完整的显示出来，这样就可以通过改变图像框的尺寸来实现对图片的放大或缩小。

【例8-2】 用图像框来显示图片，并对图片进行放大和缩小操作。

1. 新建一个工程。
2. 向窗体上添加一个图像框控件和两个命令按钮，并按表 8-2 来设置控件的属性。

表 8-2 控件属性值

控件	属性	属性值	说明
命令按钮 1	名称	cmdSmall	缩小图片
	Caption	缩小	
命令按钮 2	名称	cmdLarge	放大图片
	Caption	放大	
图像框	名称	imgCar	显示图片

3. 调整控件尺寸，窗体最终样式如图 8-5 所示。
4. 在窗体上单击图片框控件，选中该控件，在【属性】窗口，单击【Picture】属性栏右端的…按钮，打开如图 8-2 所示的【加载图片】对话框，然后从文件列表中选中一个图片文件，单击 打开(O) 按钮，向图片框控件中加载图片，如图 8-6 所示。

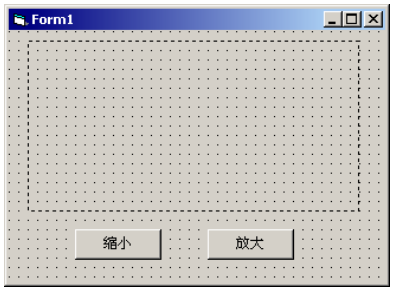


图8-5　添加控件后的窗体

图8-6　加载图片后的窗体

5. 在窗体上分别双击 缩小 按钮、 放大 按钮，为两个命令按钮添加 Click 事件，然后在代码窗口添加如下代码：

```
Option Explicit
Private Const small As Single = 0.5
Private Const large As Single = -1

Private Sub cmdLarge_Click()
    Zoom imgCar, large
End Sub

Private Sub cmdSmall_Click()
    Zoom imgCar, small
End Sub

'放大、缩小处理过程
Private Sub Zoom(ByVal img As Image, ByVal ratio As Single)
    '设置 Stretch 属性为 True，让图形适应图像框的尺寸，
    img.Stretch = True
    '通过改变图片框的尺寸和位置来实现对图片的放大和缩小
    img.Left = img.Left + img.Width * ratio / 2
    img.Top = img.Top + img.Height * ratio / 2
    img.Width = img.Width - img.Width * ratio
    img.Height = img.Height - img.Height * ratio
End Sub
```

6. 运行程序，分别单击 缩小 按钮、 放大 按钮，看看有什么效果。

比较【例 8-2】和【例 8-1】，可以看出，图片框和图像框对所显示的图片进行放大和缩小处理所采用的方法不一样。在【例 8-2】中，是通过改变图像框的尺寸来实现对图片的放大和

缩少，图像框和图片的尺寸同时改变，而在【例 8-1】中，是通过改变图片的尺寸来实现图形的放大和缩小，图片框的尺寸并不改变。总的说来，图像框所采用的方法简单易实现，因此如果只是用来显示图片，并只对图片进行简单的操作时，一般都采用图像框来显示图片。

8.2 设置坐标系

图片框不仅可以用来显示图像，另外还可以在它上面绘制一些基本图形，但在绘图之前，必须先设置绘图所采用的坐标系，以便于图形的定位和绘制图形。在 Visual Basic 6.0 中，坐标系有默认坐标系和用户自定义坐标系两种，都可以用于控件的定位和图形的定位。

8.2.1 默认坐标系

任何一种坐标系，都必须有自己的坐标轴及单位刻度，并且每个坐标轴都有自己的正方向，Visual Basic 6.0 的坐标系也不例外。在 Visual Basic 6.0 中，默认坐标系含有水平和垂直两个坐标轴，水平方向为 *X* 轴，垂直方向为 *Y* 轴，向右为 *X* 轴的正方向，向下为 *Y* 轴的正方向。

在 Visual Basic 6.0 中，窗体、图片框都是容器类控件，它们各自都有属于自己的坐标系。新建一个工程后，系统便会以窗体的左上角为原点建立窗体的默认坐标系，如图 8-7 所示的红色坐标系，如果向窗体上添加控件（包括图片框控件），则这些控件的位置（即 Left、Top 属性值）是相对于窗体的坐标系而言的。例如，向窗体上添加一命令按钮，其 Left 属性值为 100，Top 值为 100，Width 属性值为 1215，Height 属性值为 495，这时命令按钮是通过窗体的坐标系（红色的坐标系）来定位，如图 8-7 所示。

如果向窗体上添加图片框控件，由于图片框也是容器类控件，系统便会以图片框的左上角为原点建立图片框自己的默认坐标系，如图 8-7 所示的蓝色坐标系。这时如果向图片框中添加控件，则控件的位置是相对图片框的坐标系，而不是相对于窗体的坐标系而言的。例如，向图片框中添加一命令按钮，并将 Left 属性值为 100，Top 值为 100，Width 属性值为 1215，Height 属性值为 495，这时命令按钮是通过图片框的坐标系（蓝色坐标系）来定位的。如果这些属性值是相对于窗体坐标系而言的话，则这两个按钮应该重合，但在图 8-7 中两按钮并未重合，说明 Left 值和 Top 值是相对于图片框的坐标系而言的。

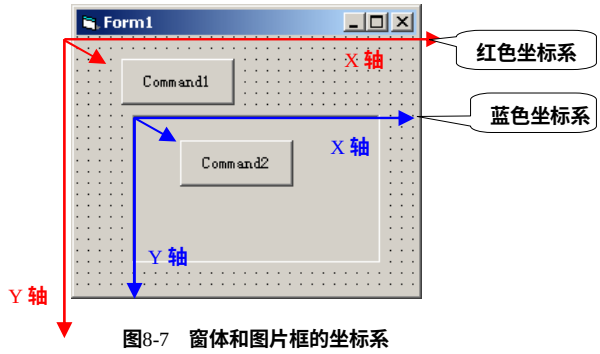


图8-7　窗体和图片框的坐标系

向图片框添加控件时，必须先在工具箱中选中控件，然后按住鼠标左键，在图片框上拖动一下，便向图片框中添加了该控件。

坐标系建立后，便要为坐标系指定单位，缺省单位为缇（Twip），1440 缇等于 1 英寸。除

了使用默认单位之外，用户还可以通过设置窗体、图片框的“ ScaleMode ”属性来设定单位。“ ScaleMode ”常用属性值见表 8-3。

表 8-3　　“ ScaleMode ”常用属性值

属性值	常量	说明
0	vbUser	用户自定义
1	vbTwip	以缇为单位，1440 缇为 1 英寸
2	vbPoint	以点为单位，72 点等于 1 英寸
3	vbPixel	以像数为单位，多少像数为 1 英寸 由显示器的分辨率来定
5	vbInch	以英寸为单位
6	vbMilimeters	以毫米为单位
7	vbCentimetre	以厘米为单位

如果直接设置了“ ScaleWidth ”、“ ScaleHeight ”、“ ScaleTop ”、“ ScaleLeft ”4 个属性中的任何一个，则“ ScaleMode ”属性会自动设为 0。

8.2.2　自定义坐标系

在窗体上添加图片框控件后，系统会自动的为窗体和图片框建立如图 8-7 所示的默认坐标系统，但有时候为了绘图的方便，我们需要自己来定义坐标系，建立用户自定义的坐标系统，如图 8-8 所示。用户建立自定义坐标系时，不仅可以将坐标原点设在窗体、图片框的外面，而且还可以将 *X* 轴、*Y* 轴的正方向反过来，即 *X* 轴的正方向向左，*Y* 轴的正方向向上。另外在默认坐标系下，横坐标最大值为窗体或图片框的宽度，纵坐标最大值为窗体或图片框的高度，而在自定义的坐标系下，横坐标最大值可以超过窗体或图片框的宽度，纵坐标最大值也可以超过窗体或图片框的高度。

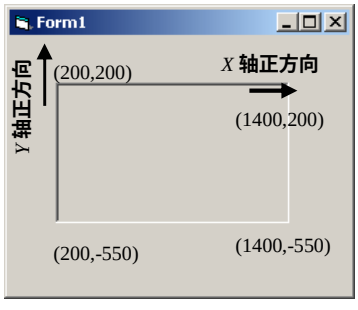


图8-8　自定义坐标系

【例8-3】　在一图片框上建立满足以下条件的坐标系：图片框的左上角的坐标设置为（200，200）；*X* 轴的正方向为向右，最大值为 1200；*Y* 轴的正方向为向上，最大值为 750。

1. 新建 1 个工程。
2. 向窗体添加一个图片框控件，并将图片框的 Top 属性设为 500，Left 属性设为

500，Width 属性设为 800，Height 属性设为 500。

3. 双击窗体，为窗体添加 Load 事件，并在代码窗口添加如下代码：

```
Private Sub Form_Load()
    Picture1.ScaleTop = 200
    Picture1.ScaleLeft = 200
    Picture1.ScaleWidth = 1200
    Picture1.ScaleHeight = -750
End Sub
```

4. 运行程序，便在图片框上建立了如图 8-8 所示的坐标系，图片框 4 个顶点的坐标如图 8-8 所示。

在【例 8-3】所建立的坐标系中，坐标原点不再是图片框的左上角，而在图片框的外面，另外 *Y* 轴的正方向也不再是向下，而是向上，并且横坐标的最大值（1440）超过了图片框的宽度，纵坐标的最大值（550）也超过了图片框的高度。在【例 8-3】中，是通过设置“ScaleWidth”、“ScaleHeight”、“ScaleTop”、“ScaleLeft”属性来建立如图 8-3 所示的坐标系的。

“ScaleTop”、“ScaleLeft”属性：返回或设置一个对象左上角的坐标，通过设置“ScaleTop”、“ScaleLeft”属性来定义坐标系原点的位置。在【例 8-3】中，将图片框的“ScaleTop”、“ScaleLeft”属性设为如下形式：

```
Picture1. ScaleTop=200
Picture1. ScaleLeft=200
```

这时图片框左上角的坐标不再是（0，0），而变成了（100，100），相应的坐标原点也不再是图片框左上角了，而在图片框的外面。

ScaleWidth、ScaleHeight 属性：用于设置 *X* 轴长度和 *Y* 轴长度。在【例 8-3】中，将图片框的 ScaleWidth 属性设为了如下形式：

```
Picture1.ScaleWidth=1200
```

这时 *X* 轴的长度为 1200。另外 ScaleWidth，ScaleHeight 属性还可以设为负值，但此时的负值并不表示 *X* 轴的长度，*Y* 轴的长度为负值，而是用来规定 *X* 轴，*Y* 轴的正方向。如果“ScaleWidth”为负，则表示 *X* 轴的正方向为向左；如果“ScaleHeight”属性为负值，则表示 *Y* 轴的正方向为向上。在【例 8-3】中，将图片框的“ScaleHeight”属性设为了如下形式：

```
Picture1.ScaleHeight=-750
```

这时 *Y* 轴的正方向变为向上，而不再是向下了。

在 Visual Basic 6.0 中，除了通过设置“ScaleWidth”、“ScaleHeight”、“ScaleTop”及“ScaleLeft”属性来建立自定义坐标系之外，还可以直接使用 Scale 方法来快速的建立自定义坐标系。Scale 方法的语法结构如下：

```
对象名.Scale(x1,y1)-(x2,y2)
```

其中对象名一般为窗体或图片框的名称，如 Picture1，x1 相当于“ScaleLeft”属性，y1 相当于“ScaleTop”属性，x2-x1 相当于“ScaleWidth”属性，y2-y1 相当于“ScaleHeight”属性。对于图 8-3 所示的坐标系，也可以使用下面的方法来建立：

```
Picture1.Scale(200,200)-(1400,-550)
```

8.3 设置绘图属性

在用图片框（窗体）来绘图之前，建立好图片框（窗体）的坐标系，这还只是绘图的第一步，接下来还要设置与绘图有关的属性，包括线条类型、线条的宽度、绘图模式等属性。另外如果所绘的图是封闭的，比如矩形或者圆，这时还要设置填充样式和填充颜色。

8.3.1 线型与线宽

用来画图的线条有实线、虚线、点化线等各种不同的类型，因此在画图之前必须选择好线条的类型。在 Visual Basic 6.0 中，线条类型的选择是通过“ DrawStyle ”属性来设置的。“ DrawStyle ”的常用属性值见表 8-4。“ DrawStyle ”属性的默认值为 1，即实线，如果要将线条的类型的设为其他格式，可用以下语法格式：

```
对象名.DrawStyle＝属性值
```

其中“ 对象名 ”可为窗体的名称或图片框的名称，“ 属性值 ”可取表 8-4 中的属性值或常量。

表 8-4　　“ DrawStyle ”常用属性值

属性值	常量	说明
0	vbSolid	实线
1	vbDash	虚线
2	vbDot	点线
3	vbDashDot	点化线
4	vbDashDotDot	双点化线

对于实线而言，线条还有粗细之分，在 Visual Basic 6.0 中线条粗细是通过“ DrawWidth ”属性来设置，并以像素为单位。设置“ DrawWidth ”属性的语法结构如下：

```
对象名.DrawWidth＝value
```

其中“ 对象名 ”为窗体或图片框的名称，“ value ”为大于 1 的任意数，包括整数和小数。

只有实线有粗细之分，对于其他类型的线条而言，“ DrawWidth ”属性只能取 1。如果“ DrawWidth ”为大于 1 的数时，则“ DrawStyle ”属性会自动的设为 0，即此时线条只能是实线。

8.3.2 绘图模式

在画画时，都有这样一个经验，在一张白纸上画图，我们随便画什么颜色的图形都行，但如果纸张是有颜色的，这时我们就要考虑图形颜色与纸张颜色之间的关系，如果两者同色，我们是看不到所画的图形的，如果两者不同色，这样所画的图形才能被看见，并且两者颜色的差别越大，我们所看到的图形越清晰。Visual Basic 6.0 绘图，和现实中画画一样，如果绘图区的底色为白色或者图形之间没有重叠时，只管将图形画到绘图区，但如果绘图区的底色不为白色或者图形之间有重叠时，这时就要考虑绘图模式的选择，即考虑图形颜色与绘图区底色之间或图形颜色之间的逻辑关系。

在 Visual Basic 6.0 中，绘图模式的选择是通过“ DrawMode ”属性来设置的，“ DrawMode ”常用属性见表 8-5。图形显示的效果，不仅与图形的颜色有关，而且还与“ DrawMode ”属性的有关。通过设置 DrawMode 属性来控制图形显示的效果，这将在以后的

章节有详细的介绍。

表 8-5 “DrawMode”常用属性值

属性值	常量	说明
1	vbBlackness	黑色输出
2	vbNotMergePen	前景颜色与画笔颜色做 Or 操作后，再取反
3	vbMaskNotPen	将画笔颜色取反，再与背景颜色做 And 操作
4	vbNotCopyPen	将画笔颜色取反
5	vbMaskPenNot	将前景颜色取反，再与画笔颜色做 And 操作
6	vbInvert	将前景色取反
7	vbXorPen	将画笔颜色与前景色进行互斥操作
8	vbNotMaskPen	将前景色和画笔颜色做 And 操作，再取反
9	vbMaskPen	将前景色和画笔颜色做 And 操作
10	vbNotXorPen	将画笔须色与前景色进行互斥操作，再取反
11	vbNop	没有画线颜色，即输出保持不变，相当于关闭画图
12	vbMergeNotPen	将画笔颜色取反，再与前景色进行 Or 操作
13	vbCopyPen	默认设置，用前景色画线
14	vbMergePenNot	将前景色取反，再与画笔颜色进行 Or 操作
15	vbMergePen	将前景色与画笔颜色做 Or 操作
16	vbWhiteness	白色

8.3.3 填充样式和填充颜色

如果所画的图形是封闭的，例如圆或者矩形，这时就可以在图形所围的区域内加入各种填充的图案。在 Visual Basic 6.0 中，填充图案的选择是通过“FillStyle”属性来设置的，“FillStyle”常用属性值见表 8-6。

填充图案颜色的选择是通过 FillColor 属性来设置的，语法结构如下：

```
对象名.FillColor=value
```

其中 value 值可以是 Visual Basic 6.0 中的一些常量颜色值，如 vbRed（红色），vbBlue（蓝色），vbGreen（绿色），vbBlack（黑色），vbWhite（白色）等，还可以通过 RGB 函数来选择颜色。RGB 函数的语法结构如下：

```
RGB(red,green, blue)
```

其中 red，blue，green 都为整型数值，取值范围都为 0～255，通过这 3 个参数不同的搭配可以配置出任何的颜色。如，RGB(255,0,0)为深红，RGB(0,0,255)为蓝色，RGB(0，255，0)为绿色。

表 8-6 “FillStyle”属性值

值	常数	说明
0	vbFSSolid	实心
1	vbFSTransparent	透明（默认值）
2	vbHorizontalLine	水平线

续表

值	常数	说明
3	vbVerticalLine	垂直线
4	vbUpwardDiagonal	上斜线
5	vbDownwardDiagonal	下斜线
6	vbCross	十字线
7	vbDiagonalCross	交叉对角线

8.4 绘图方法

坐标系建立好了，与绘图的有关属性也设置好了，便可以开始在图片框（窗体）上绘图了。在 Visual Basic 6.0 中，可以通过采用不同的方法来完成各种简单图形的绘制。与绘图有关的常用方法有 Pset 方法、Line 方法、Circle 方法、Cls 方法。

8.4.1 Pset 方法

Pset 方法是用来画任意位置的点，同时还可以给所画的点加上颜色，其语法结构如下：

```
[对象名].Pset[Step](x,y),[color]
```

其中“对象名”为窗体名或图片框名,可选参数，如果不特别说明，与绘图有关的所有方法的对象名都为窗体名或图片框名；*x*,*y* 为单精度变量，为画点的坐标值，用来指定所画点的位置，必选参数；color 为点的颜色，可选参数，如果不指定 color 参数，则以对象的“ForeColor”属性值来设置点的颜色。

【例8-4】 在窗体上产生霓虹灯闪烁的效果。

1. 新建一个工程。
2. 在窗体上添加一个定时器控件，并将定时器的 Interval 属性设为 50。
3. 双击定时器，为定时器控件添加 Timer 事件，并在代码窗口添加如下代码：

```
Private Sub Timer1_Timer()
    '设置点的大小
    Form1.DrawWidth = 50
    '随机的画彩色点
    Form1.PSet (Rnd*Form1.Width,Rnd*Form1.Height), RGB(Rnd*255, _
    Rnd*255, Rnd*255)
End Sub
```

4. 运行程序，便在窗体产生霓虹灯闪烁的效果，如图 8-9 所示。

在【例 8-4】中，“DrawWidth”属性是用来设置点的大小，Rnd*Form1.Width，Rnd*Form1.Height 对应的是点的坐标，只不过点的坐标是随机产生的，而不是固定的，这主要是由于使用了 Rnd 函数。Rnd 函数用于随机的产生 0～1 之间的单精度数。另外 RGB(Rnd*255，Rnd*255，Rnd*255)是用来设置点的颜色，点的颜色也是随机产生的。

使用 Pset 方法来画点时，点的坐标都是相对于坐标系的原点而言的，但如果在（x1，y1）前面加上了 Step，这时 x1，y1 的值不再是相对于坐标系原点而言的，而是相对上一次所画点的位置而言的。例如，以下代码可以在窗体上画出如图 8-10 所示的两个点。

```
Form1.ScaleMode=3
Form1.DrawWidth=10
Form1.Pset(80,100)
Form1.Pset Step(50,50)
```

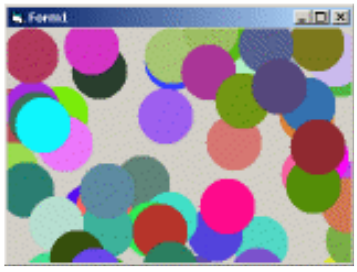


图8-9　【例 8-4】运行后的效果图

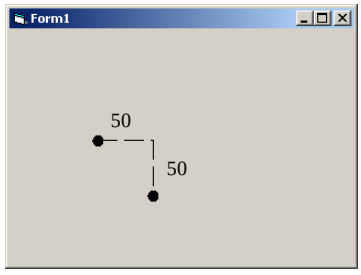


图8-10　用 pest 方法画出的两个点

8.4.2　Line 方法

Line 方法用来画任意两点间的连线，完整的语法结构如下：

```
[对象名].Line [[Step](x1,y1)]-[Step](x2,y2),[color],[B][F]
```

其中(x1,y1)为第一点的坐标，即直线起点的坐标，可选参数，如果省略(x1,y1)，则以上一次画线的终点作为本次画点的起点；(x2,y2)为第二点坐标，即直线终点的坐标，必选参数；color 为直线的颜色，可选参数，如果省略 color 参数，则线条颜色为上一次画线条的颜色。

【例8-5】　在图片框中绘制不同类型的直线。

1. 新建一个工程，并在窗体上添加一个图片框控件。
2. 调整图片框的大小至如图 8-11 所示。

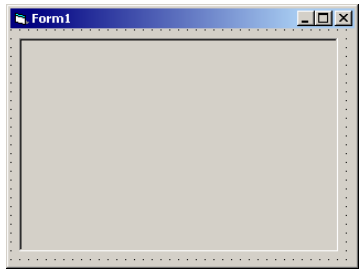


图8-11　调整后的窗体

3. 双击窗体为窗体添加 Load 事件，并在代码窗口添加如下代码：

```
Private Sub Form_Load()
    Dim i As Integer
    Dim y As Long
    Picture1.AutoRedraw = True
    For i = 0 To 4
        '设置线的类型
        Picture1.DrawStyle = i
        y = (300 * i) + 800
            '如果线型为实线，则设置线宽为 5 个像素
            If Picture1.DrawStyle = 0 Then
```

```
                Picture1.DrawWidth = 5
            'If not a solid line, then the line width can only be 1
                Else
                    Picture1.DrawWidth = 1
                End If
            'Set drawing mode
                Picture1.DrawMode = i + 3
                Picture1.Line (850, y)-(3100, y), vbGreen
            Next
        End Sub
```

4. **运行程序，在图片框中会显示 5 种不同类型的直线，如图 8-12 所示。**

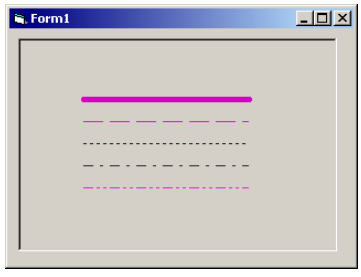


图8-12　【例 8-5】运行后的界面

如果将【例 8-5】中代码变为如下代码：

```
Private Sub Form_Load()
Dim i As Integer
Dim y As Long
Picture1.AutoRedraw = True
For i = 0 To 4
'Set line type
Picture1.DrawStyle = i
y = (300 * i) + 800
'Set line width
Picture1.DrawWidth=5
'Set drawing mode
Picture1.DrawMode = i + 3
Picture1.Line (850, y)-(3100, y), vbGreen
Next
End Sub
```

运行这段程序，看看有什么效果，并想一想为什么？

画线时，线条的颜色不仅仅取决于 Line 方法的“color”属性，还取决画图模式的选择。在【例 8-5】中，画线时，将所有线条的颜色都设为了绿色（vbGreen），但显示出来的线条却

没有一条是绿色，并且线条的颜色还有所不一样，这主要是由于它们的绘图模式不一样而造成的。在画线时，灵活地使用“DrawMode”属性，不仅可以得到不同效果的直线，而且还可以实线拖动画线的效果。

【例8-6】 在图片框上实现拖动画线，即在图片框上单击鼠标左键，然后按住鼠标左键不放，在图片框上移动，便在图片框上拉出一条直线，并且拉出的直线随着鼠标的移动而移动，松开鼠标，便结束画线。

1. 新建一个工程。
2. 在窗体上添加一个图片框，并调整图片框的大小至图 8-11 所示。
3. 在窗体上双击窗体的空白处（注意：不要双击到图片框或窗体标题上）为窗体添加 Load 事件，并打开【代码】窗口。在【代码】窗口的对象列表框中选中“Picture1”，在事件/过程列表框中分别单击 MouseDown、MouseMove、MouseUp 事件，为图片框添加 MouseDown、MouseMove、MouseUp 事件，并在【代码】窗口添加如下代码：

```
Public Draw As Boolean
'保存第一次单击鼠标的位置
Public starX, starY As Integer
'跟踪鼠标的位置
Public endX, endY As Integer

Private Sub Form_Load()
    '允许在图片框上重画
    Picture1.AutoRedraw = True
    '将图片框的背景色设为白色
    Picture1.BackColor = vbWhite
    '将图片框的前景色设为黑色
    Picture1.ForeColor = vbBlack
End Sub

Private Sub Picture1_MouseDown(Button As Integer,Shift As Integer,X _
As Single,Y As Single)
    '按下了鼠标左键，准备开始画直线
    Draw = True
    '记录单击鼠标的位置
    starX = X
    starY = Y
    endX = X
    endY = Y
    ' 将 DrawMode 属性设为 2，表示画出的直线与当前屏幕颜
    '色相反，这样就可以通过画同样的直线来达到擦除的目的
    Picture1.DrawMode = 2
```

```
End Sub

Private Sub Picture1_MouseMove(Button As Integer, Shift As Integer, _
X As Single,Y As Single)
    '若按下鼠标键，则开始画线
    If Draw Then
        '删除上一次拖动所画的直线
        Picture1.Line (starX, starY)-(endX, endY)
        '拖动画线
        Picture1.Line (starX, starY)-(X, Y)
        '跟踪鼠标的位置
        endX = X
        endY = Y
    End If
End Sub

Private Sub Picture1_MouseUp(Button As Integer, Shift As Integer, _
X As Single, Y As Single)
    '松开鼠标，画线结束
    Draw = False
End Sub
```

4. **运行程序，在图片框上单击鼠标左键，然后按住鼠标左键不放，便可以实现拖动画线，如图 8-13 所示。松开鼠标，画线结束。**

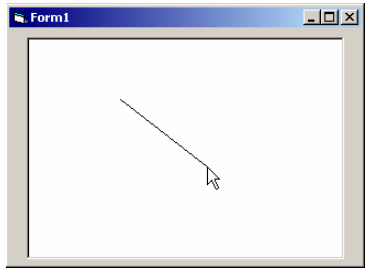


图8-13　拖动画线

在【例 8-6】中，将图片框的“ DrawMode ”设为 2，即前景颜色与画笔颜色做 or 操作后，再取反，这样就可以通过在同一位置画同样的直线将先前所画的直线“ 掩盖 ”起来，从而达到擦除先前所画直线的效果。在【例 8-6】中，在 Picture1_MouseDown 事件中两次使用 Line 方法，通过在同一位置画同样的直线实现“ 擦除 ”的效果，并且不断更新所画的线，实现拖动画线的动画效果。

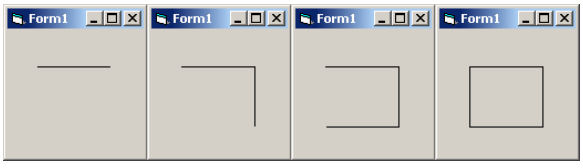
读者不妨将“ DrawMode ”设为 13 或者其他值，然后再运行程序，看看显示的效果有什么不同。

Line **方法除了可以画直线之外，还可以通过画 4 条首尾相连的直线来实现画矩形和正方**

形。下面这段代码便可以在窗体上画一个长为 1000，宽为 800 的矩形。

```
Form1.Line(400,500)-Step(1000,0)
Form1.Line-Step(0,800)
Form1.Line-Step(-1000,0)
Form1.Line-Step(0,-800)
```

同 Pset 方法一样，在坐标前面加上 Step 表示该点的坐标是相对于前一点的坐标而言的。以上代码实现画矩形的过程如图 8-14 所示。



(a)　(b)　(c)　(d)

图8-14　用 Line 方法画矩形的过程

采用画 4 条首尾相连的直线来画矩形（正方形），过程比较繁琐，并且不易操作，除了这种方法之外，Visual Basic 6.0 还提供了另外一种更简单的画矩形方法。以上代码所画的矩形也可以通过下面的代码来实现：

```
Form1.Line(400,500)-Step(1000,800),,B
```

只需要在 Line 方法的 color 参数后面加上一个“B”，即可以很简单使用 Line 方法来画矩形或正方形，而参数(x1,y1)、(x2,y2)两点的坐标分别代表是矩形的左上角和右下角的坐标。

如果未设置 color 参数时，在“B”之前必须预留 color 参数的位置，即必须在“B”前面加两个“,”。

由于矩形和正方形为封闭的图形，因此可以在矩形和正方形中加入各种填充的图案，但在画矩形和正方形之前，必须先设置好“FillStyle”和“FillColor”两个属性。

以下代码可以在窗体上产生不同填充样式的矩形，如图 8-15 所示。

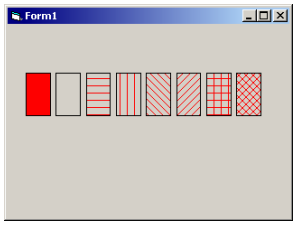


图8-15　不同填充样式的矩形

【例8-7】　以下代码可以在窗体上产生不同填充样式的矩形，如图 8-15 所示。

```
Private Sub Form_Load()
    Dim R, I, X, Y As Integer
    X = 300
    Y = 800
    Form1.AutoRedraw = True
    For I = 0 To 7
        '设置填充样式
```

```
            Form1.FillStyle = I
            '设置填充颜色
            Form1.FillColor = RGB(255, 255, 0)
            Form1.Line (X, Y)-(X + 400, Y + 700), , B
            X = X + 500
        Next I
    End Sub
```

8.4.3　Circle 方法

Circle 方法是用来画与弧线有关的图形，包括圆、椭圆以及圆弧，完整的语法结构如下：

```
对象名.Circle Step(x,y),radiu,color,star,end,aspect
```

用 Cicle 方法来绘制图形时，图形不一样，所带的参数也不一样。

- **画圆**

要画一个完整的圆，我们只需要知道圆心的位置和圆的半径就可以了，因此用 Circle 方法来画圆时，只需要圆心的坐标及半径两个参数即可，这时 Circle 方法的语法结构可简写为如下形式：

```
对象名.Circle[Step](x,y),radiu,[color]
```

其中(x,y)为圆心的坐标，radiu 为圆的半径，color 为圆的颜色，如果省略 color 属性，则用前景颜色来画圆。(x,y)前面加上 Step 表示圆心的坐标是相对于上一次画点的坐标。

由于圆也是封闭的图形，因此也可以像矩形那样在圆中加入图案。如果将【例 8-6】中

```
Form1.Line (X, Y)-(X + 400, Y + 700), , B
```

代码行换为如下代码：

```
Form1.Circle (X, Y), 200
```

则在窗体上产生如图 8-16 所示的各种填充样式的圆。

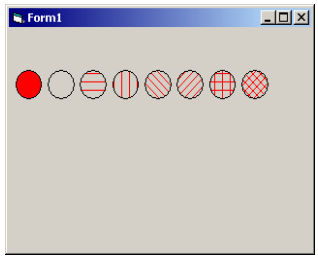


图8-16　各种填充样式的圆

- **画圆弧**

圆弧可以看成是圆的一段，使用 Circle 方法来画圆弧时，除了圆心坐标和半径之外，还要指定圆弧的起始位置和终止位置，这时 Circle 方法的语法结构如下：

```
对象名.Circle Step(x,y),radiu,color,star,end
```

其中参数 star 用于指定圆弧的起始角，单位为弧度，取值范围为-2p～2p，默认值为 0；end 用于指定圆弧的终止角，单位为弧度，取值范围为-2p～2p，默认值为 2p。

Visual Basic 6.0 中常用 pi 来代替 p。如果一个角的度数为 k，则其所对应的弧度数为 $\alpha = \frac{k * \pi}{180}$。

虽然 star，end 可以取负角，但其意义和数学上的负角意义不一样，此时的负号表示圆弧要带边界线。如果 star 为负，则圆弧要带起始边界线；如果 end 为负，则圆弧要带终止边界线；如果 star，end 都为负，则圆弧既要带起始边界线，还要带终止边界线；如果 star，end 都为正，则不带任何边界线，4 种不同样式的圆弧如图 8-17 所示。

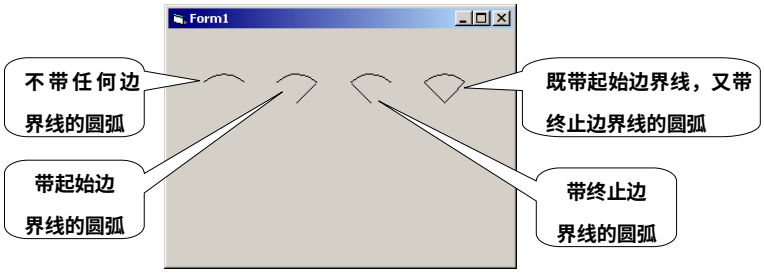


图8-17　4 种样式的圆弧

【例8-8】　以下代码，便可以在窗体上画出如图 8-17 所示的 4 种不同样式的圆弧。

```
Private Sub Form_Load()
    pi = 3.14
    Form1.AutoRedraw = True
    Form1.Circle (750, 1000), 400, , pi / 4, 3 * pi / 4
    Form1.Circle (1750, 1000), 400, , -pi / 4, 3 * pi / 4
    Form1.Circle (2750, 1000), 400, , pi / 4, -3 * pi / 4
    Form1.Circle (3750, 1000), 400, , -pi / 4, -3 * pi / 4
End Sub
```

在画圆弧时，参数 star，end 的值表示与 *X* 轴正方向的夹角，并且逆时针方向为角度增大的方向，顺时针方向为角度减小的方向。虽然参数 star，end 的取值范围为-2p～2p，但由于负号表示的是要带边界线，因此能决定弧度大小和位置的 star，end 的值只能在 0～2p 区间内。

- 画椭圆

如果将一个标准圆沿水平方向拉伸或沿垂直方向拉伸，这时便得到一个椭圆，如图 8-18 所示。Circle 方法的最后一个参数 aspect，便是用来将一个标准的圆拉伸成一个椭圆。aspect 的值表示的是垂直方向或水平方向拉伸的比例，如果为 1，表示水平和垂直方向都不拉伸，即为一个标准的圆；如果大于 1，表示沿垂直方向拉伸，为垂直式椭圆，如图 8-18 左边的椭圆；如果小于 1，表示沿水平方向拉伸，为水平式椭圆，如图 8-18 右边的椭圆。

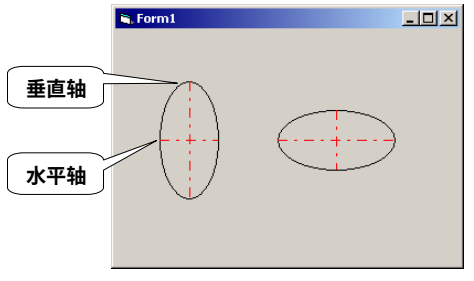


图8-18　两种不同样式的椭圆

【例8-9】　在一图片框中，先画出一个圆，然后将此圆拉伸为两种样式的椭圆。

1. 新建一个工程。

2. 向窗体上添加一个图片框和两个命令按钮，调整窗体和控件的大小如图 8-19 所示，并按表 8-7 设置按钮的属性。

表 8-7　　按钮属性值

命令按钮	属性	属性值	说明
命令按钮 1	名称	cmdXExtend	单击按钮一次，圆沿水平方向被拉伸一次
	Caption	水平拉伸	
命令按钮 2	名称	cmdYExtend	单击按钮一次，圆沿垂直方向被拉伸一次
	Caption	垂直拉伸	

3. 为两个命令按钮分别添加 Click 事件，并在相应的事件中添加如下代码：

图8-19　主窗体样式

```
Dim j As Double '控制水平拉伸的比例
Dim i As Integer '控制垂直拉伸的比例

Private Sub cmdXExtend_Click()
    '将 DrawMode 属性设为 2，
    '这样就可以通过画同样的椭圆
    '来达到擦除原来画的椭圆的目的
    Picture1.DrawMode = 2
    '擦除原来拉伸的椭圆
    Picture1.Circle (2000, 1200), 500, , , , j
    '改变拉伸的比例
    j = (1 - j) * 0.5
    '画新拉伸的椭圆
    Picture1.Circle (2000, 1200), 500, , , , j
End Sub

Private Sub cmdYExtend_Click()
    Picture1.DrawMode = 2
    Picture1.Circle (2000, 1200), 500, , , , i
    i = i + 2
    Picture1.Circle (2000, 1200), 500, , , , i
End Sub
```

```
Private Sub Form_Load()
    Picture1.AutoRedraw = True
    Picture1.BackColor = vbWhite
    Picture1.Circle (2000, 1200), 500, vbRed
End Sub
```

4. 运行程序，如果单击[水平拉伸]按钮，则圆被拉伸为水平的椭圆，如图 8-20 所示；如果单击[垂直拉伸]按钮，则圆被拉伸为垂直椭圆，如图 8-21 所示。

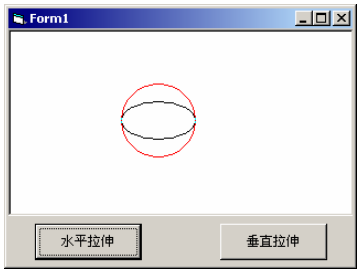


图8-20　圆被拉伸为水平椭圆

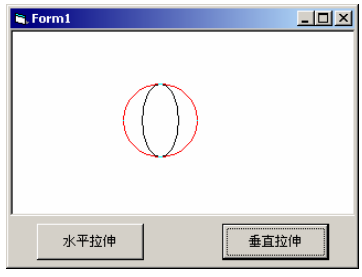


图8-21　圆被拉伸为垂直椭圆

在用 Circle 方法来画椭圆时，如果 aspect 参数小于 1，则椭圆的水平轴的长度为常数 2×radium，而垂直轴的长度为 2×aspect×radium，随着 aspect 值的改变而改变；如果 aspect 参数大于 1，则椭圆的垂直轴的长度为常数 2×radium，而水平轴的长度随着为 2×aspect×radium，随着 aspect 值的改变而改变。在【例 8-8】中，每单击[水平拉伸]按钮（[垂直拉伸]按钮）一次，所画的椭圆就会改变一次，但每次水平轴（垂直轴）的长度都保持不变，只是垂直轴（水平轴）的长度在变。

8.4.4　Cls 方法

用 Pset、Line、Circle 方法可分别画出点、线、圆等图形，但如果想将所画的图形清除掉，该如何处理呢？前面已经介绍了其中的一种方法，即将 DrawMode 属性设为 2，然后在相同的位置画同样的图形，便可以将图形擦除掉，在【例 8-6】和【例 8-8】中便是使用这种方法将某个图形擦除掉。这种方法虽然可以擦除图形，但每次只能擦除一个图形，如果想同时擦除所有的图形，使用这种方法便比较繁琐。Visual Basic 6.0 另外还为用户一种更为简单方法，Cls 方法。Cls 方法可以同时将图片框或窗体上的所有图形都清除掉，以方便用户重新绘图。Cls 方法的语法结构如下：

```
[对象名].Cls
```

使用 Cls 方法就相当于使用一块橡皮将图纸上所有的图形都擦除掉，这时绘图区（图片框）就又变为一张“白纸”。

8.5　常用绘图控件

Visual Basic 6.0 除了提供一些常用方法让用户来绘图之外，还提供了两个专门用于绘图的控件：线条控件和形状控件，如图 8-22 所示。和基本控件一样，这两种绘图专用控件都有自

图8-22　形状控件和直线控件

己的一些属性，但它们不能响应任何事件，因此这两种控件都无法编写任何事件过程，用户只需要设置相关的属性，便可以得到一些常用的基本图形。

8.5.1　直线控件

直线控件 是用来画直线的专用控件，总共有 12 个属性，用户可以通过设置这些属性来设计各种不同的直线，如图 8-23 所示，这些属性包括设置直线位置、长度、颜色、粗细以及线条类型等内容。

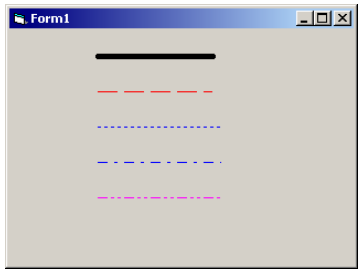


图8-23　各种不同样式的直线

【例8-10】用直线控件在窗体上画出如图 8-23 所示的直线。

1. 新建一个工程。
2. 在工具箱中，双击直线控件，向窗体上添加直线控件，以同样的方法向窗体再添加 4 个直线控件，调整直线控件的位置如图 8-23 所示，并按表 8-8 设置直线控件的属性。

表 8-8　　直线控件属性设置

控件名	属性	属性值	属性	属性值
Line1	BorderStyle	1－Solid	X2	3000
	BorderWidth	5	Y1	360
	X1	1320	Y2	360
Line2	BorderColor	&H000000FF&	X2	3000
	BorderStyle	2－Dash	Y1	840
	X1	1320	Y2	840
Line3	BorderColor	&H00FF0000&	X2	3000
	BorderStyle	3－Dot	Y1	1320
	X1	1320	Y2	1320
Line4	BorderColor	&H00FF0000&	X2	3000
	BorderStyle	4－Dash－Dot	Y1	1800
	X1	1320	Y2	1800
Line5	BorderColor	&H00FF00FF&	X2	3000
	BorderStyle		Y1	2400
	X1	1320	Y2	2400

3. 运行程序，窗体上便会画出如图 8-23 所示的图形。

用直线控件来画直线时，由于直线控件有自己的线条类型（BorderStyle）、线条宽度

（BorderWidth）、线条颜色（BorderColor）、线条显示模式（DrawMode）等属性，因此不需事先设置图片框或窗体的 DrawStyle、DrawWidth、DrawMode 等属性，便可以画出各种不同的直线。【例 8-8】并没有改变窗体的任何属性，而只是设置了直线控件相关的属性，便画出了如图 8-23 所示的各种样式的直线。在以上的 4 个属性中，BorderStyle 与图片框或窗体的 DrawStyle 相对应，BorderWidth 与图片框或窗体的 DrawWidth 相对应，DrawMode 与图片框或窗体的 DrawMode 相对应。

与 DrawWidth 属性一样，如果直线控件的 BorderWidth 属性值大于 1，则 BorderStyle 属性会自动设为 1－Solid。读者不妨将【例 8-9】中 Line2 的 BorderWidth 属性值设为 5，看看有什么效果。

除了以上 4 个属性外，直线控件另外还有用来控制直线位置及长度的 4 个属性：X1、Y1、X2、Y2。用直线控件来画直线时，直线的开始位置由 X1、Y1 的值来指定，结束的位置由 X2、Y2 的值来设定。

8.5.2 形状控件

形状控件是用于在窗体或图片框中显示各种形状的图形，这些图形包括矩形、正方形、椭圆、圆、圆角矩形、圆角正方形等，如图 8-24 所示。

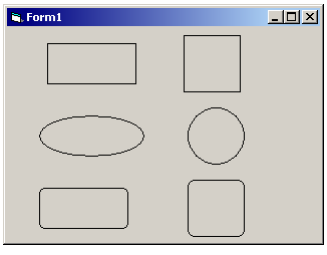


图8-24 各种形状的图形

形状控件的用法和直线控件一样，都是通过设置有关属性来画相应的图形。形状控件的“BorderStyle”、“BorderColor”、“BorderWidth”、“DrawMode”等属性，分别用于线条类型、线条颜色、线条宽度、线条显示效果的选择，所画图形位置及大小是通过形状控件的“Left”、“Top”、“Width”、“Height”这一组属性来控制的。除了这些基本属性之外，形状控件还有一个重要的属性：“Shape”属性，该属性用于图形的选择，其常用属性值见表 8-9。

表 8-9 “Shape”常用属性值

属性值	常量	图形形状
0	vbShapeRectangle	矩形
1	vbShapeSquare	正方形
2	vbShapeOval	椭圆
3	vbShapeCircle	圆
4	vbShapeRoundedRectangle	圆角矩形
5	vbShapeRoundedSquare	圆角正方形

由于形状控件所显示的图形都是封闭的，因此可以在这些图形中加入填充的图案。填充图案及其颜色的选择是通过形状控件的“FillStyle”和“FillColor”属性来完成的，而不需要通过设置图片框或窗体的“FillStyle”和“FillColor”属性来完成的。

8.6 动画处理

Visual Basic 6.0 中的动画并不是在电视上所看的卡通动画，而只是能够动起来的图片或者图形。为了让图片或图形动起来，最常用的方法便是使用定时器控件。

使用 Visual Basic 6.0 所提供的方法或控件在图片框（窗体）上画图时，所画的图形一般都是静止不动的，但如果使用定时器定时的在图片框（窗体）上画图形，便可以让所画图形“动起来”，并且还可以通过改变定时器的 Interval 属性来改变图形“动”的速度。例如，【例 8-4】便是通过定时器来实现点闪烁的效果，并且还可以通过改变定时器的 Interval 属性值来让闪烁的速度加快或减慢。读者不妨改变【例 8-4】中的“Interval”属性值，然后再运行程序，看看有什么变化。

通过使用定时器控件，可以让所画的图形动起来，同样使用定时器也可以让图片框或图像框中所显示的图片动起来。

【例8-11】用图片框和图像框来实现豹子奔跑的效果。

1. 新建一个工程。
2. 向窗体添加 3 个命令按钮、1 个图片框、1 个定时器，并按表 8-10 设置窗体和控件的有关属性。

表 8-10　　窗体和控件的属性值

	属性	属性值	说明
窗体	名称	frmAnimation	
	BackColor	&H00FFFFFF&，即白色	
命令按钮 1	名称	cmdStar	控制豹子的奔跑
	Caption	开始奔跑	
命令按钮 2	名称	cmdQuickly	让豹子奔跑的速度加快
	Caption	快跑	
命令按钮 3	名称	cmdSlowly	让豹子奔跑的速度减慢
	Caption	慢跑	
图片框	名称	picAnimate	显示豹子奔跑时的动作图片
	BorderStyle	0－None	

3. 向窗体添加 1 个图像框，并将名称属性设为“imgLeopard”。
4. 再向窗体添加 1 个图像框，并将名称属性也设为“imgLeopard”，在名称属性设置完毕之后，随便单击属性区中任何一栏，这时会弹出如图 8-25 所示的提示对话框。

图8-25　创建控件数组提示框

5. 单击提示对话框的 是(Y) 按钮，为图片框创建一个控件数组。
6. 以同样的方法向窗体添加另外 6 个图像框控件，并将所有图像框的名称属性都

设为“ imgLeopard ”，这时所有的图片框都成为控件数组中的成员。添加图像框后的窗体如图 8-26 所示。

创建控件数组之后，系统会按添加控件的先后顺序为每个控件数组成员赋一个索引值，以示区别。

7. 按前面所讲的方法，为图片框和图像框加载图片，加载图片后的窗体如图 8-27 所示（这里所加载的图片是豹子奔跑的一系列动作，图片文件另附）。

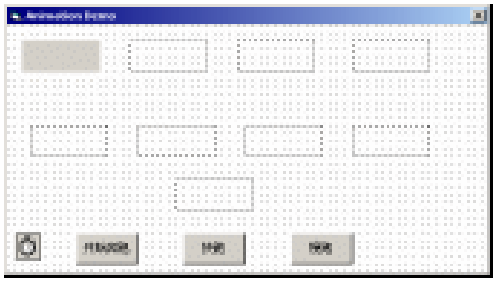


图8-26 添加控件后的窗体

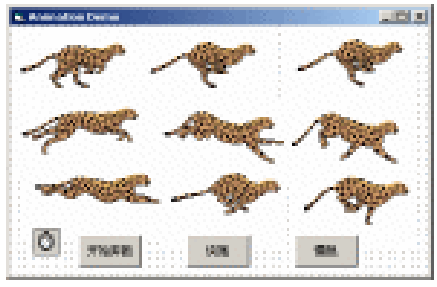


图8-27 加载图片后的窗体

8. 将所有图像框的 Visible 属性都设为“ False ”，程序运行时便看不到图像框了。
9. 打开【代码】窗口，向【代码】窗口添加如下代码：

```
Option Explicit
Dim i As Integer '控制快跑的速度
Dim j As Double '控制慢跑的速度

Private Sub cmdQuickly_Click()
    '单击按钮“快跑”一次，奔跑的速度增加一次
    j = (1 - j) * 0.5
    Timer1.Interval = 100 * j
End Sub

Private Sub cmdSlowly_Click()
    '单击按钮“慢跑”一次，奔跑的速度减少一次
    i = i + 1
    Timer1.Interval = 100 * i
End Sub

Private Sub cmdStar_Click()
    Select Case cmdStar.Caption
        Case "开始奔跑"
            '按正常的速度奔跑
            Timer1.Interval = 100
            '按钮变为“暂停”按钮
            cmdStar.Caption = "暂停"
```

```
            Case "暂停"
                '暂停奔跑
                Timer1.Interval = 0
                '按钮变为“开始奔跑”按钮
                cmdStar.Caption = "开始奔跑"
        End Select
    End Sub

    Private Sub Form_Load()
        picAnimate.Left = 0 - picAnimate.Width
        picAnimate.Top = 500
    End Sub

    Private Sub Timer1_Timer()
        Static currentpic As Integer
        '在图片框中不断的显示图像框中的图片，
        '并且不断的更新图片框的位置，从而实现
        '奔跑的效果
        If currentpic = 7 Then
            currentpic = -1
        End If
        currentpic = currentpic + 1
        picAnimate.Left = picAnimate.Left + 400
        If (picAnimate.Left) > ScaleWidth Then
            picAnimate.Left = -36
        End If
        picAnimate.Picture = imgLeopard(currentpic).Picture
    End Sub
```

10. 运行程序，单击 开始奔跑 按钮，这时便看到一头豹子从窗体的左边跑出来，并且一直跑到窗体的最右边，然后又从窗体的最左边开始跑，如图 8-28 所示，同时 开始奔跑 按钮变为 暂停 按钮。这时如果单击 暂停 按钮，则豹子停止奔跑，同时 暂停 按钮变为 开始奔跑 按钮。

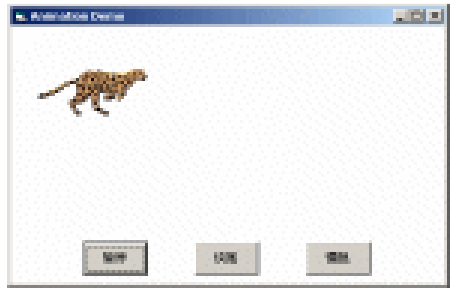


图8-28　奔跑中的豹子

11. 快跑 和 慢跑 按钮是用来控制豹子奔跑的速度的。单击 快跑 按钮，豹子奔跑的速度加快，不停的单击 快跑 按钮，则豹子奔跑的速度不断加快；单击 慢跑 按钮，豹子奔跑的速度减慢。

【例 8-10】实现豹子奔跑的效果，并不是让图片框中的豹子跑起来，而只是让图片框定时的移动，并且图片框移动到一个新的位置，图片框

中所显示的图片便更换一次，整个奔跑过程的实现就相当于按图 8-29 所示的顺序，将豹子奔跑的 8 个动作图片定时的在窗体不同的位置用图片框显示出来。为了让整个奔跑的过程是连续的，图片显示位置之间不能相隔太远，并且整个显示过程是循环进行的，直到程序终止或程序暂停。豹子奔跑速度的快慢是通过控制图片框移动的速度来控制的，即定时器的“ Interval ”属性值。“ Interval ”属性值越大，图片框移动的速度越慢，豹子跑得也就越慢；“ Interval ”属性值越小，图片框移动的速度越快，豹子跑的也就越快。

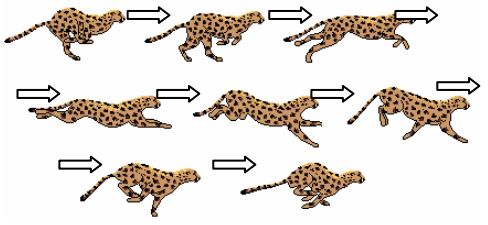


图8-29　实现奔跑效果的顺序图

从【例 8-4】和【例 8-10】中，可以看出，Visual Basic 6.0 所实现的动画过程其实很简单，就是将图形定时的在图片框中画出来或定时地移动图片框，还不能算完全意义上的动画效果，如果要做像电视上所看到的卡通动画，则最好还是用专门的软件来做，例如：3D Studio、Animator Pro 等软件。

8.7 小结

强大的图形处理能力是 Visual Basic 6.0 精髓之一，用 Visual Basic 6.0 来绘图既简单而且又容易，另外使用 Visual Basic 6.0 还能够进行简单的图像处理和实现简单的动画效果。

通过本章的学习，要掌握以下内容：

- 图片框和图像框两种控件的常用属性及其他们之间的区别。
- 简单的图像处理，包括图像的放大、缩小、移动、翻转。
- 建立自定义坐标系的方法。
- 线条类型、线条宽度、绘图模式、填充样式、填充颜色的设置方法，其中绘图模式的理解是本章的难点。
- 使用绘图方法绘制常用的基本图形，包括点、直线、矩形、圆、圆弧、椭圆。
- 使用直线控件和形状控件这两种专用绘图控件来绘制基本图形。
- 简单动画效果的实现。

8.8 习题

一、 填空题

1. 图片框和图像框都可以用来显示图片，但可以在_____上画图，而不能在_____上画图。
2. 为了将整幅图片显示在图片框中，必须将图片框的_____属性设为_____。
3. 用户建立自己的坐标系，可以通过同时设置图片框或窗体的_____、_____、_____、_____这 4 个属性来建立，也可以通过_____方法来建立。
4. 在用图片框来绘图之前，除了要建立好坐标系之外，还必须设置好线条的类型、线条的宽度、绘图的模式、填充的样式和填充的颜色，其中线条类型由_____属性来设置，线条宽度_____属性来设置，绘图模式由_____属性来设置，填充样式由_____属性来设置，填

充颜色由_____属性来设置。

5. 如果要将图片框中所有的图形都清除掉，可以通过使用_____方法来实现，_____方法可以用来画点，_____方法可以用来画直线，_____方法可以用来画圆、圆弧、椭圆。
6. _____和_____是两种专用的绘图控件，其中_____控件可用来画直线，_____可用来画各种形状的图形，包括矩形、椭圆、圆等。

二、 选择题

1. 图片框和图像框都是通过以下哪个属性来设置显示的图片的（）。

   A. MouseIcon　　B. Image　　C. Picture　　D. Icon

2. 使用下面语句建立了一个自定义坐标系：

   Form1.ScaleLeft=1

   Form1.ScaleTop=1

   Form1.ScaleWidth=5

   Form1.ScaleHeight=5

   如果要使用 Scale 方法来建立同样的坐标系，则 Scale 的语法结构为（）。

   A. Form1.Scale(0,0)-(5,5)

   B. Form1.Scale(1,1)-(6,6)

   C. Form1.Scale(1,1)-(5,5)

   D. Form1.Scale(0,0)-(6,6)

3. 将“DrawWidth”设置为大于 1 的数，则“DrawStyle”属性自动的设为（）。

   A. 0　　B. 1　　C. 2　　D. 3

4. 在图片框中所画图形的颜色与（）属性有关。

   A. DrawStyle　　B. DrawMode　　C. DrawWidth　　D. ScaleMode

5. 如果要在图片框中画一个既带起始边界线又带终止边界线的圆弧，圆心坐标为(1000,1000)，圆弧半径为 200，则下面代码正确的是（）。

   A. Picture1.circle(1000,1000),200，2，4

   B. Picture1.Circle(1000,1000),200，-2，4

   C. Picture1.Circle(1000,1000),200，2，-4

   D. Picture1.Circle(1000,1000),200，-2，-4

6. 下面哪种图形是形状控件不能显示出来的（）。

   A. 圆　　B. 圆弧　　C. 椭圆　　D. 矩形

三、 问答题

1. 如何向图片框和图像框中加载图片。
2. 用图片框和图像框如何实现对图片的放大和缩小。
3. 如何画点、画直线、画矩形、画圆、画圆弧、画椭圆

四、 程序设计题

1. 用图像框控件设计一个图片浏览器，浏览器的界面如图 8-30 所示（参考上机指导第 8 章的实验一）。
2. 用 3 种方法在窗体上画一个矩形。矩形左上角的坐标为(1000 1000)，矩形的宽为 800，高为 600。
3. 编写一个程序，实现在图片框上拖动画圆。即在图片框上单击鼠标左键，然后按住鼠标

左键不放，在图片框上移动，便在图片框上拉出一个圆，并且拉出的圆随着鼠标的移动而移动，松开鼠标，便结束画圆。

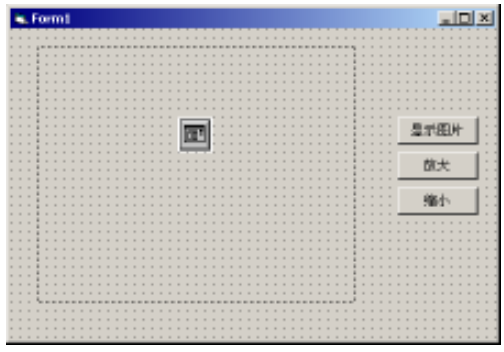


图8-30　图片浏览器

4. 使用 Pset 方法设计一个在窗体上动态画正弦的程序。
5. 设计一个简单的扫描仪，如图 8-31 所示，让图中的扫描针不停在圆盘上旋转扫描。要求：扫描针一直绕着圆盘的圆心旋转。（参考上机指导第 8 章实验三）

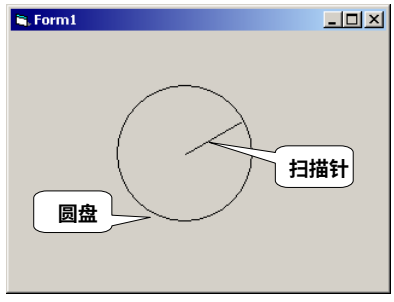


图8-31　扫描仪

第9章　Visual Basic 6.0 高级界面

到目前为止，本书所涉及的界面都只有一个窗体，比较单一，然而在实际的应用中，往往仅一个窗体还不能满足程序的需要，需要向工程中添加多个窗体才能完成，另外为了方便编写程序，还会向工程中添加一些公用的模块或自己创建的类。在本章我们主要学习如何设计一些高级的界面。

本章学习目标

- 多窗体界面的设计。
- 多文档界面的设计。

9.1 多窗体界面的设计

对于一些简单的程序而言，单一的一个窗体便能满足所有的需要，但在编写具体的应用程序时，单一的一个窗体是不能满足要求的，势必要用到一个以上的窗体。例如,一个学生信息管理系统，除了有用于显示学生信息的窗口之外，还应有学生基本信息输入的窗口。

多窗体是指包含有多个窗体的界面，如图 9-1 所示，这些窗体之间没有绝对的从属关系，每个窗体的地位都是平等的，它们之间只存在相互调用的关系，各个窗体出现的顺序也有所不同。

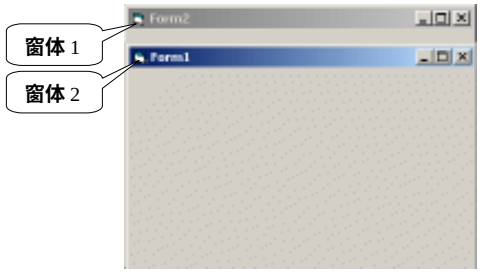


图9-1　多窗体界面

9.1.1 窗体的添加

在 Visual Basic 6.0 中，可以向应用程序中添加两种窗体，一种是新建的窗体，另外一种是已经存在的窗体。无论是哪种窗体，都是通过【添加窗体】对话框来完成的。

添加新建的窗体，可按以下步骤来完成：

1. 单击【工程】/【添加窗体】菜单，打开如图 9-2 所示的【添加窗体】对话框。
2. 在对话框中单击 图标，然后单击 打开(O) 按钮，便向窗体中添加了一个新窗体，并以层叠的形式显示在最上层，如图 9-3 所示。

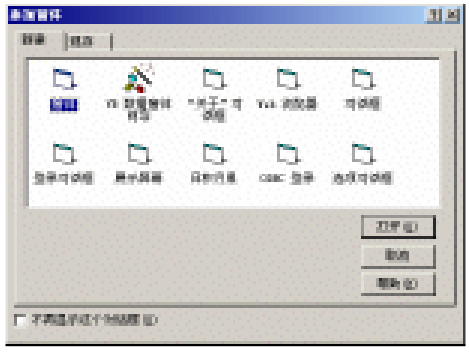


图9-2　【添加窗体】对话框

应用程序有多个窗体之后，所有的窗体便以图标的形式列于窗体资源管理器中，如图 9-4 所示，窗体按添加的先后顺序以层叠的形式叠加在窗体设计窗口，最先添加的窗体显示在最低层，最后添加的窗体显示在最上层，如图 9-3 所示。如果要让

某个窗体显示在最上层，则只需在窗体资源管理器中双击该窗体的图标就可以了。

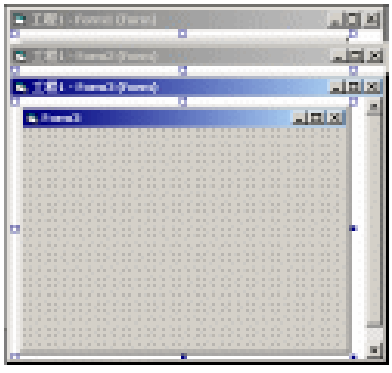


图9-3　窗体以层叠的形式显示

图9-4　窗体资源管理器

向窗体中添加了新窗体后，便可以按前面几章所讲的方法为新窗体设计界面，并添加相关的事件代码。在设计某个窗体时，必须先在窗体资源管理器中双击该窗体图标，将该窗体置于最上层。除了可以向窗体中添加新窗体之外，还可以向窗体中添加已经设计好了的窗体。在图 9-2 所示的【添加窗体】对话框中，单击【现存】选项卡，这时对话框变为如图 9-5 所示，在窗体文件列表窗口，单击某个窗体文件，然后单击 打开(O) 按钮，便向应用程序中添加了一个已经存在的窗体。

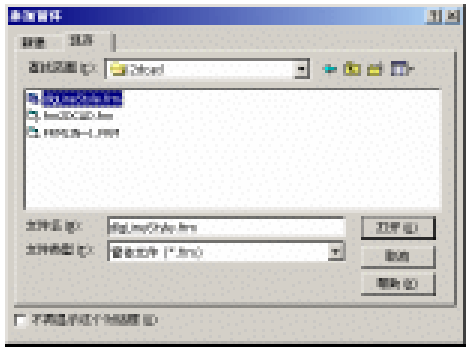


图9-5　添加已存在的窗体文件

向应用程序添加已经存在的窗体，如果应用程序中的窗体和所添加的窗体同名，则不能添加。

9.1.2　窗体的控制

窗体数目增加后，伴随而来的便是一些操作上的问题。例如，程序运行时，该显示哪个窗体，并且程序运行时只能显示一个窗体，那么其他窗体该如何显示或如何隐藏等，这些问题也是本节所要讨论的问题。

- 启动窗体的选择

新建一个工程后，系统会自动为应用程序添加一个名为“ Form1 ”的窗体，程序运行之时【Form1】便是默认的启动窗体。向应用程序中新增窗体之后，新增的窗体虽然在程序设计阶段是可见的，但在程序运行之时，是不可见的，并不能被启动。如果要让新增的窗体先启动，必须先将新增的窗体设为“ 启动对象 ”。

【例9-1】　向工程中添加一个新的窗体，并让该窗体先被显示。

1. 新建一个工程。
2. 单击【工程】/【添加窗体】菜单，弹出如图 9-1 所示的【添加窗体】对话框。
3. 在对话框中单击 图标，然后单击 打开(O) 按钮，便向工程中添加了如图 9-2 所示的新窗体。
4. 单击【工程】/【工程属性】菜单，打开【工程属性】对话框，如图 9-6 所示。
5. 在【工程属性】对话框中，单击最上面【通用】选项卡，然后单击【启动对象】列表栏右端的箭头，打开下拉列表，从中选择“ Form2 ”，如图 9-6 所示。
6. 单击 确定 按钮，回到主窗体。
7. 运行程序，这时显示的窗体便是【Form2】，而不是【Form1】。

从【例 9-1】中可以看出，工程中所有的窗体都会显示在【工程属性】对话框的【启动对象】列表栏中，如果要让程序一开始就启动某个窗体，只需在【启动对象】列表框中选中该窗体，然后单击【工程属性】对话框的【确定】按钮，将该窗体设为“ 启动对象 ”就可以了。

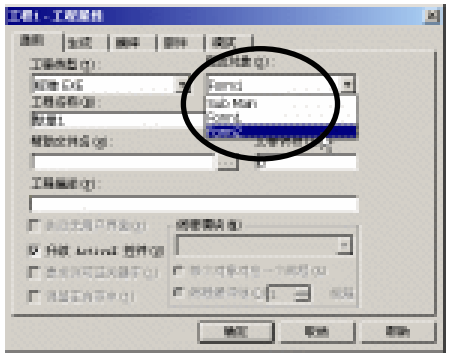


图9-6　【工程属性】对话框

- 窗体的显示和隐藏

在多窗体应用程序中，启动窗体在程序运行时会自动被显示出来，而其他的窗体的显示必须通过 Show 方法来实现。用 Show 方法来显示窗体的语法结构如下：

```
窗体名.Show [style]
```

其中“ style ”参数用来指定窗体的模式，为可选参数，它的值为 0 或 1。取 1 时，表示窗体是模态的，即用户只能操作所显示的窗体，而不能操作其他窗体；取 0 时，表示窗体是非模态的，即用户既能操作所显示的窗体，又可以操作其他窗体。例如，可以通过下面代码来显示窗体名为“ Form2 ”的窗体。

```
Form2.Show 1
```

【例9-2】　改为“ 为例 8-10 添加一个启动界面，启动界面如图 9-7 所示。

1. 打开【例 8-10】所的工程，按前面所讲的步骤向工程中新增一个窗体。
2. 在窗体资源管理器中，双击图标 Form1 (Form1)，将窗体【Form1】置于最上层。

图9-7　启动窗体

在多窗体应用程序中，设计某个窗体，必须先选中该窗体，将它置于最上层，然后才能来设计该窗体。

3. **向窗体【Form1】中添加 1 个标签控件、1 个命令按钮和 1 个定时器，将命令按钮的名称属性设为“ cmdRun ”，Caption 属性设为“ 进入 ”，定时器的 Interval 属性设为“ 100 ”。**
4. **在【Form1】窗体中选中标签控件，在【属性】窗口单击【字体】属性栏，这时【字体】属性栏最右端便会出现…按钮，单击该按钮，打开【字体】对话框，将字体的“ 字形 ”改为粗斜体，字体的大小改为“ 小三 ”，如图 9-8 所示，单击 确定 按钮，回到【Form1】窗体。**

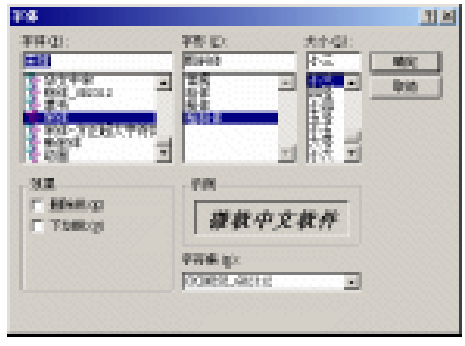


图9-8 【字体】对话框

5. **将标签控件的 Caption 属性设为“ 欢迎进入豹子奔跑演示程序 ”，调整窗体和控件的大小至图 9-9 所示。**

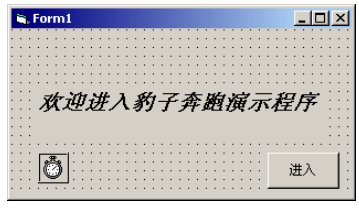


图9-9 调整后的窗体

6. **在【Form1】窗体上，分别双击定时器控件，为定时器添加 Timer 事件，并打开【Form1】的【代码】窗口。**
7. **向【Form1】的【代码】窗口中添加如下代码：**

```
Option Explicit
Dim i As Double

Private Sub cmdRun_Click()
    '单击命令按钮，进入演示程序
    frmAnimation.Show 0
    '设置窗体 frmAnimation 上的定时器的 Interval 属性
    frmAnimation.Timer1.Interval=100
End Sub
```

```
Private Sub Timer1_Timer()
    '定时让标签控件移动
    Label1.Left = Rnd * 0.35 * Form1.Width
    Label1.Top = Rnd * 0.35 * Form1.Height
    '定时更改标签控件的前景颜色，让标签控件有闪烁的效果
    Label1.ForeColor = RGB(Rnd * 255, Rnd * 255, Rnd * 255)
    '设置窗体 frmAnimation 上的命令按钮 cmdStar 的 Caption 属性
    frmAnimation.cmdStar.Caption = "暂停"
End Sub
```

8. 单击【工程】/【工程属性】菜单，打开【工程属性】对话框，如图 9-6 所示。
9. 在【工程属性】对话框中，单击最上面【通用】选项卡，然后单击【启动对象】列表栏右端的箭头，打开下拉列表，从中选择“Form1”。
10. 单击[确定]按钮，回到主窗体。
11. 运行程序，【Form1】窗体先被显示，如图 9-10 所示。单击【Form1】窗体上的[进入]按钮，frmAnimation 窗体才被显示，并且豹子开始奔跑。

图9-10　程序运行时的界面

在【例 9-2】中，单击【Form1】窗体上的[进入]按钮后，frmAnimation 窗体虽被显示，但【Form1】窗体并没有消失，仍然显示在屏幕上。为了让 Form1 窗体从屏幕上消失，必须将它隐藏起来。将窗体隐藏起来有两种方法，一种是通过 Hide 方法将窗体隐藏起来，另外一种是通过 Unload 方法将窗体释放掉。

用 Hide 方法隐藏窗体的语法结构如下：

```
窗体名.Hide
```

例如，在【例 9-2】中，如果命令按钮的 Click 事件中加入带底纹的代码：

```
Private Sub cmdRun_Click()
    '单击命令按钮，进入演示程序
    frmAnimation.Show 0
    Form1.Hide
End Sub
```

再运行程序，单击【Form1】窗体上的[进入]按钮，frmAnimation 窗体被显示，同时 Form1 窗体也从屏幕上消失了。

用 Unload 方法隐藏窗体的语法结构如下：

```
Unload  窗体名
```

例如，在【例 9-2】中，如果命令按钮的 Click 事件中加入带底纹的代码：

```
Private Sub cmdRun_Click()
    '单击命令按钮，进入演示程序
    frmAnimation.Show 0
    Unload  Form1
End Sub
```

便可以在 frmAnimation 窗体被显示时，【Form1】窗体也会从屏幕上消失。

使用 Hide 方法将窗体隐藏起来，虽然窗体会从屏幕上消失，但并没有从计算机的内存中消失，与该窗体有关的程序代码还在内存继续运行。用 Unload 方法不仅可以将窗体隐藏起来，而且可以将窗体从内存中删除掉，并终止与该窗体有关的所有程序代码。

读者不妨分别用 Hide 方法和 Unload 方法将【例 9-2】中【Form1】窗体隐藏起来，然后再关闭 frmAnimation 窗体，看看两者的效果有什么不同。

在设计多窗体应用程序时，可以按设计单窗体的方法，分别为各个窗体设计属于它们自己的界面形式和程序代码，互不影响，并且它们之间还可以相互访问。在多窗体应用程序中，不仅可以在某个窗体的程序代码中为该窗体上的控件设置有关属性，而且还可以为其他窗体上的控件设置有关属性。为其他窗体上的控件设置属性，可按以下语法结构来完成：

窗体名.控件名.控件属性＝属性值

例如，在【例 9-2】中在窗体【Form1】的程序代码中，通过以下代码

```
frmAnimation.Timer1.Interval=100
```

为窗体 Form1 的定时器设置了 Interval 属性。

9.2 多文档界面的设计

对于多文档界面，大家并不陌生，Word，Excel 等办公室软件都是多文档界面，多文档界面主要是由父窗口及其所属的子窗口构成，与前面所讲的多重窗体界面是不一样的。在多重窗体界面中，各个窗体之间并没有绝对的从属关系，每个窗体的地位都是平等的，窗体可以在屏幕的任何地方活动，而在多文档界面中，窗口之间是有明确的从属关系的，子窗口以层叠形式显示在父窗口之内，如图 9-11 所示，或以平铺形式显示在父窗口之内，如图 9-12 所示，并且只能在父窗口中活动，子窗口随着父窗口的消失而消失。

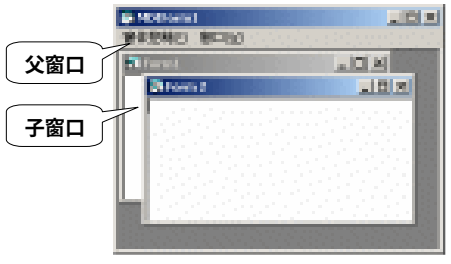


图9-11　以层叠的形式显示窗口

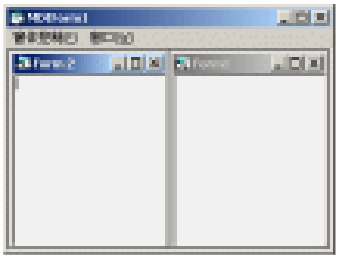


图9-12　以平铺的形式显示窗口

9.2.1 添加多文档窗体

在多文档界面中，父窗口是一个多文档窗体（MDI Form），它为其所属的子窗口提供了一个工作平台。虽然父窗口上可以同时显示多个子窗口，但每次只能激活一个子窗口，并且一个应用程序中只能有一个父窗口。

【例9-3】　为工程新增一个多文档窗体，为建立多文档界面做准备。

1. 新建一个工程。
2. 单击【工程】/【添加 MDI 窗体】菜单，弹出如图 9-13 所示的【添加 MDI 窗体】对话框。
3. 在【MDI 窗体列表】窗口，单击图标，然后单击 打开(O) 按钮，便向工程中添加一个多文档窗体，和添加窗体一样，新增的多文档窗体以层叠的形式显示在最上层，如图 9-14 所示。

在设计窗体时，多文档窗体和标准窗体在外形上没什么区别，但两者还是存在着一定的差别。在设计时，只能向多文档窗体中添加具有 Align 属性的控件（图片框）或者是尺寸不可改变的控件（如定时器），其余的控件都不能添加到多文档窗体上。另外虽然两者在设计阶段，外形相差不大，但在程序运行时，两者存在着明显的差别。在【例 9-3】中，我们按 9.1.2 节所讲的方法，将【Form1】设为启动对象，运行程序便得到一个标准的运行窗体，如图 9-15 所示。退出程序，将【MDIForm1】设为启动对象，然后再运行程序，便得到一个多文档窗体，如图 9-16 所示。

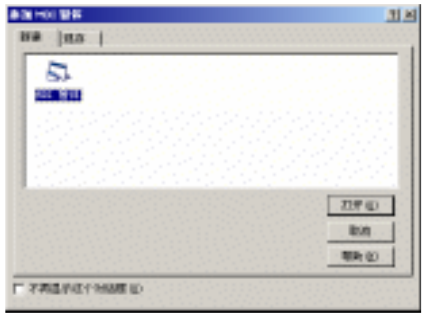


图9-13　【添加 MDI 窗体】对话框

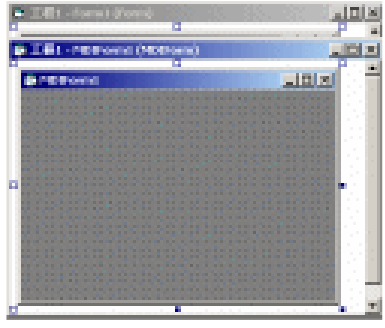


图9-14　多文档窗体

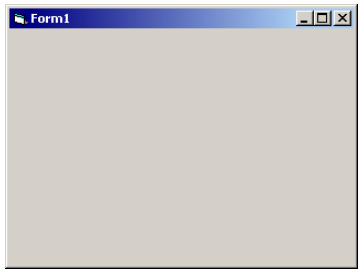


图9-15　标准窗体

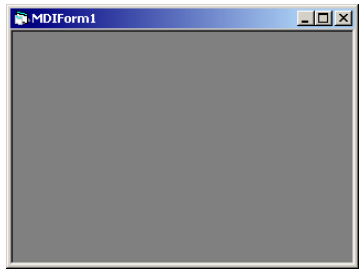


图9-16　多文档窗体

9.2.2　添加子窗口

应用程序有了多文档这个父窗口之后，便可以向父窗口中添加子窗口。在多文档界面中，子窗口实际上就是标准的窗体。标准窗体变为子窗口后，在程序运行之时，便不能随意移动，只能在父窗口上活动，如图 9-17 所示，但在设计阶段，窗体还是自由的，可以随意移动。

【例9-4】　为【例 9-3】的多文档窗体添加子窗口。

1. 打开【例 9-3】所建的工程。
2. 在窗体资源管理器中，双击图标 Form1 (Form1)，将窗体【Form1】置于最上层，将【Form1】的“MDIChild”属性设为“True”，这时【Form1】便成了多文档窗体的子窗口。
3. 在窗体资源管理器中，双击图标 MDIForm1 (MDIForm1)，将多文档窗体【MDIForm1】置于最上层。
4. 双击多文档窗体【MDIForm1】，为多文档窗体添加 Load 事件，并在代码窗口中添加如下代码：

```
Private Sub MDIForm_Load()
    Form1.Show
```

```
End Sub
```

5. 单击【工程】/【工程属性】菜单，打开【工程属性】对话框，然后将【MDIForm1】设为启动对象。
6. 运行程序，【Form1】窗体便显示在多文档窗体中，并成为其子窗口，如图 9-17 所示。

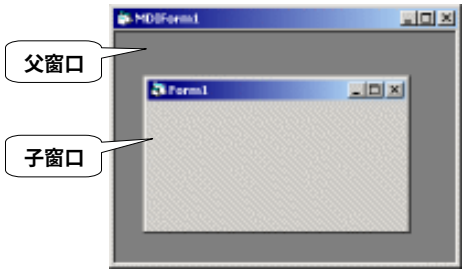


图9-17　父窗口和子窗口

从【例 9-4】中，可以知道要让一个标准的窗体成为多文档窗体的子窗口，其实很容易，只需将该窗体的 MDIChild 属性设为“ True ”即可。如果要让多文档窗体有多个子窗口，可以按 9.1.1 所讲的方法先向工程中添加多个窗体，然后将所有标准窗体的“ MDIChild ”属性都设为设为“ True ”。

9.2.3 子窗口的控制

当标准窗体成为子窗口之后，在程序运行之时，默认状况是不被显示在父窗口之中的，要让它显示出来，必须使用 Show 方法。【例 9-4】便是通过 Show 方法将【Form1】显示出来的。另外子窗口显示在父窗口之后，以何种方式显示在父窗口中，便是设计多文档界面接下来要考虑的问题。子窗口既可以以层叠的形式显示在父窗口中，如图 9-12 所示，也可以以平铺的形式显示在父窗口之中，如图 9-13 所示，因此在设计多文档界面时，必须先安排好各个子窗口显示的位置，否则会使整个界面显得很凌乱。

【例9-5】　在【例 9-4】的基础上为父窗口再添加一个子窗口，并为多文档窗体添加设计菜单，以便控制子窗口的显示和显示的方式。

1. 打开【例 9-4】所建的工程。
2. 向工程添加一个新的工程【Form2】，并将【Form2】的“ MDIChild ”属性设为“ True ”。添加窗体【Form2】之后，窗体资源管理器变为如图 9-18 所示。
3. 在窗体资源管理器中，双击图标 MDIForm1 (MDIForm1)，选中多文档窗体，并将它置于最上层。
4. 单击【工程】/【菜单编辑器】菜单，打开【菜单编辑器】，并按表 9-1 的顺序为多文档窗体设计菜单，【菜单编辑器】的最终形式如图 9-19 所示。

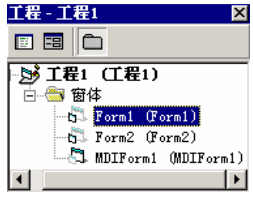


图9-18　窗体资源管理器

表 9-1　　菜单属性

序号	属性		级别	说明
1	标题栏	mnuForm	一级菜单	显示子窗口
	名称栏	窗体显示		

续表

序号	属性		级别	说明
2	标题栏	mnuForm1	二级菜单	显示或隐藏子窗口 1
	名称栏	Form1		
3	标题栏	mnuForm2	二级菜单	显示或隐藏子窗口 2
	名称栏	Form2		
4	标题栏	mnuWindow	一级菜单	排列子窗口
	名称栏	排列窗口		
5	标题栏	mnuWindowC	二级菜单	层叠排列子窗口
	名称栏	层叠		
6	标题栏	mnuWindowH	二级菜单	水平平铺排列子窗口
	名称栏	水平排列		
7	标题栏	mnuWindowV	二级菜单	垂直平铺排列子窗口
	名称栏	垂直排列		

5. **单击【菜单编辑器】对话框的 确定 按钮，生成多文档窗体的菜单栏。**

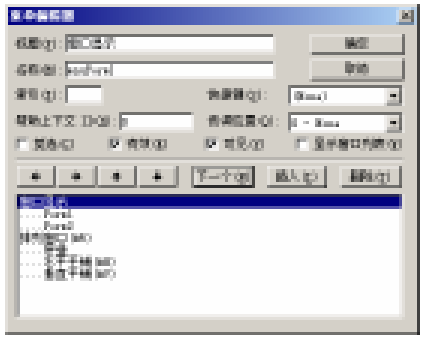


图9-19　【菜单编辑器】的最终样式

6. **单击窗体资源管理器的查看代码图标，打开多文档窗体的【代码】窗口，删除【代码】窗口中的所有代码，然后向【代码】窗口添加如下代码：**

```
Option Explicit
Private Sub mnuForm1_Click()
    ' 【Form1】菜单在有无选中标示之间切换
    mnuForm1.Checked = Not mnuForm1.Checked
    '窗体 Form1 在显示和隐藏之间切换
    Form1.Visible = Not Form1.Visible
End Sub

Private Sub mnuForm2_Click()
    '【Form2】菜单在有无选中标示之间切换
    mnuForm2.Checked = Not mnuForm2.Checked
    '窗体 Form2 在显示和隐藏之间切换
    If Form2.Visible Then
```

```
        Form2.Hide
    Else
        Form2.Show
    End If
End Sub

Private Sub mnuWindowC_Click()
    '以层叠方式排列窗口
    MDIForm1.Arrange 0
End Sub

Private Sub mnuWindowH_Click()
    '以水平平铺方式排列窗口
    MDIForm1.Arrange 1
End Sub

Private Sub mnuWindowV_Click()
    '以垂直平铺方式排列窗口
    MDIForm1.Arrange 2
End Sub
```

7. **在窗体资源管理器中双击图标 Form1 (Form1) ，选中窗体【Form1】，并将它置于最上层。**
8. **向窗体【Form1】中添加一个文本框控件，并将文本框的“ MutiLine ”属性设为“ True ”，并调整尺寸至如图 9-20 所示。**

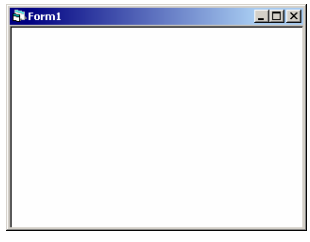


图9-20　调整尺寸后的【Form1】窗体

9. **在窗体资源管理器中，单击查看代码图标，打开窗体【Form1】的代码窗口，并向其中添加如下代码：**

```
Private Sub Text1_Change()
    '在窗体 Form1 的文本框中输入内容时，
    '窗体 Form2 的文本框中的内容也随着改变
    Form2.Text1.Text = Form1.Text1.Text
End Sub
```

10. **在窗体资源管理器中，双击图标 Form2 (Form2)，选中窗体【Form2】，并将窗体【Form2】置于最上层。**

11. 向窗体【Form1】中添加一个文本框控件，并将文本框的“MutiLine”属性设为“True”，并调整尺寸至如图 9-20 所示。
12. 在窗体资源管理器中，单击的查看代码图标，打开窗体【Form1】的【代码】窗口，并向其中添加如下代码：

```
Private Sub Text1_Change()
    '在窗体 Form2 的文本框中输入内容时，
    '窗体 Form1 的文本框中的内容也随着改变
    Form1.Text1.Text = Form2.Text1.Text
End Sub
```

13. 单击【工程】/【工程属性】菜单，打开【工程属性】对话框，然后将【MDIForm1】设为启动对象。
14. 运行程序，分别单击【窗口显示】/【Form1】菜单和【窗口显示】/【Form2】菜单，将两窗体显示在多文档窗体上。子窗口显示到父窗口之后，再分别单击【窗口显示】/【Form1】菜单和【窗口显示】/【Form2】菜单，便将两个子窗口隐藏起来了。
15. 将两个子窗口显示到父窗口上，然后单击【排列窗口】/【层叠】菜单，两子窗口以层叠的形式排列在父窗口上，如图 9-21 所示；单击【排列窗口】/【水平平铺】菜单，两子窗口以水平平铺方式排列在父窗口上，如图 9-22 所示；单击【排列窗口】/【垂直平铺】菜单，两子窗口以垂直平铺方式排列在父窗口上，如图 9-23 所示。

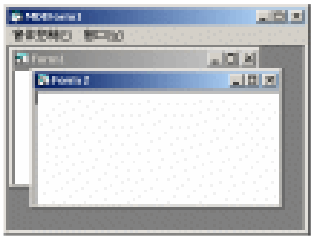


图9-21　以层叠方式排列子窗口

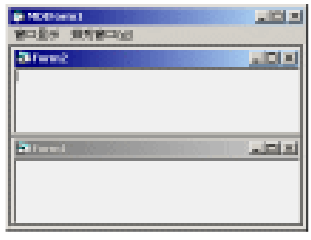


图9-22　以水平平铺方式排列子窗口

16. 以垂直平铺的方式将两子窗口排列在父窗口中，然后单击【Form1】子窗口，激活该子窗口，在【Form1】子窗口的文本框中输入文字，所输入的文字也同时会显示在【Form2】子窗口的文本框中，如图 9-24 所示（这里假设输入的是“Visual Basic 6.0”）。同样在【Form2】子窗口的文本框中输入文字，所输入的文字也同时会显示在【Form1】子窗口文本框中。

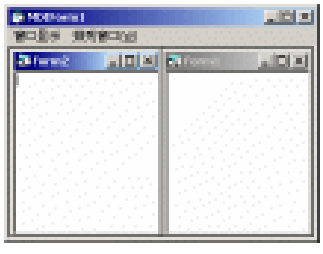


图9-23　以垂直平铺的方式排列子窗口

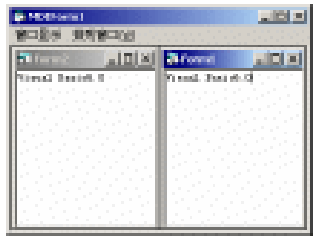


图9-24　子窗口之间互访

在多文档界面程序中，在程序运行之时，子窗口一般不显示在父窗口之上，通过 Show 方法便可以将子窗口显示在父窗口之上。【例 9-5】便是通过

```
Form2.Show
```

语句将子窗口【Form2】显示在父窗口之上的。子窗口显示在父窗口之上后，如果要隐藏子窗口，便可以通过 Hide 方法来实现。在【例 9-5】中，隐藏子窗口【Form2】便是通过

```
Form2.Hide
```

语句来实现的。除了使用 Show 和 Hide 方法可以显示和隐藏子窗口之外，还可以通过设置子窗口的“ Visible ”属性来显示或隐藏子窗口。当“ Visible ”属性为“ True ”时，显示子窗口；当“ Visble ”属性为“ Fasle ”时，隐藏子窗口。在【例 9-5】中，通过设置子窗口【Form1】的“ Visible ”属性

```
Form1.Visible＝Not  Form1.Visible
```

来显示或隐藏子窗口【Form1】的。

除了使用以上两种方法可以隐藏子窗口之外，还可以用 Unlaod 方法来隐藏子窗口。

如果要将多个子窗口显示在父窗口之上时，便要考虑如何排列子窗口了。通过使用多文档窗体的 Arrange 方法，便可以很容易管理多个子窗口。Arrange 方法的语法结构如下：

```
多文档窗体名.Arrange  style
```

其中参数 style 用于选择排列样式，style 常用属性值见表 9-2 所示。

表 9-2　　style 常用参数值

参数值	常量	说明
0	vbCascade	层叠排列子窗口
1	vbTileHorizontal	水平平铺排列子窗口
2	vbTileVentical	垂直平铺排列子窗口
3	vbArrangeIcons	排列图标

在【例 9-5】中，通过

```
MDIForm1.Arrange  0
```

以层叠的方式排列子窗口【Form1】和子窗口【Form2】。

和设计多窗体应用程序一样，在设计多文档应用程序时，可以按设计单窗体的方法，分别为父窗口和子窗口设计属于它们自己的界面形式和程序代码，并且它们之间还可以相互访问，为其他子窗口上的控件设置有关属性。在【例 9-5】子窗口【Form1】的程序代码中，通过代码：

```
Form2.Text1.Text= Form1.Text1.Text
```

为子窗口【Form2】上的文本框设置“ Text ”属性。

9.3 小结

本章所讲的内容是 Visual Basic 6.0 高级应用的基础部分，向应用程序中添加多个窗体或多文档窗体，不仅可以丰富应用程序的界面内容，而且还可以使应用程序的功能进一步加强。本章主要介绍了以下内容：

- 如何向应用程序添加窗体。
- 启动窗体的设置。

- 窗体的管理。
- 如何向应用程序中添加多文档窗体。
- 如何向多文档窗体中添加子窗口。
- 子窗口的管理。

9.4 习题

一、 填空题

1. 显示窗体所使用的方法为_____；隐藏窗体但并不从内存中将其删除所使用的方法是_____；隐藏窗体并从内存中将其删除所使用的方法是_____。
2. 要想让一个标准的窗体变为多文档窗体的子窗口，必须将标准窗体的_____属性设为“True”。
3. 可以通过使用_____方法将子窗口显示在父窗口之上；让子窗口从父窗口中消失，可以使用_____方法或_____方法，另外通过将子窗口的“Visible”属性设为_____让子窗口消失。

二、 问答题

1. 如何向工程中添加标准窗体和多文档窗体？
2. 多窗体界面和多文档界面有什么区别？
3. 在设计多窗体应用程序时，如何设置启动窗体？
4. 在设计多文档界面时，如何在一个子窗口中访问其他子窗口？
5. 排列子窗口有几种方式，分别如何实现？

第10章　文件操作

能对文件进行管理，是每个应用程序必备的功能。在 Visual Basic 6.0 的菜单栏中，专门有一个用于文件管理的下拉式菜单，如图 10-1 所示，与文件有关的操作都可以在图 10-1 所示的菜单栏中完成。Visual Basic 6.0 为用户提供了强大的文件管理功能，不仅为用户提供了专门的文件控件，而且还提供了专门用于文件操作的函数和语句。使用这些函数，可以很容易地实现对文件进行保存、打开、删除等操作。

本章学习目标

- 文件基本知识。
- 与文件操作有关的控件。
- 与文件操作有关的语句和函数。
- 文件的基本操作。

图10-1　下拉【文件】菜单

10.1　文件基本知识

用 Visual Basic 6.0 设计的程序，一般都是交互式的界面，既有数据的输入，也有数据的输出。前面几章所涉及的数据的输入和输出，都是通过很简单的操作来实现的。只是通过键盘或鼠标完成输入，在显示器上完成输出，这些输入或输出随着程序的关闭而消失，不能够被永久的保存。另外如果输入的数据比较多时，用键盘来输入很费时。如果使用文件来完成输入和输出，不仅可以永久性的保存输入或输出的数据，而且还可以一次性的完成大量数据的输入或输出。

10.1.1　文件及文件的命名

所谓的文件是指记录外部介质上的数据的集合，通常存放在磁盘上，并且每个文件都有一个文件名。一个完整的文件名包括主文件名和扩展名两部分，主文件名和扩展名以圆点相间，主文件名是文件的“名字”，扩展名决定着文件的类型，如，“Form1.frm”，其中“Form1”为主文件名，“.frm”为扩展名，表示该文件为窗体文件。由于每个文件在计算机上都有一个存储的地址，因此要访问或保存某个文件必须指明该文件的详细路径，格式如下：

磁盘驱动器名：\文件夹 1\文件夹 2\…\文件名

其中“磁盘驱动器名”用来指定文件所在的磁盘，“\文件夹 1\文件夹 2\…\”用来指明文件所在的详细位置。例如：假设访问文件“Form1.frm”所需的路径为 E：\资料\vb 资料\vb 代码\Form1.frm，其中 E 表示为 E 盘，“资料”、“vb 资料”、“vb 代码”为文件夹名。

10.1.2　字符、字段、记录

字符是构成文件的最基本单位，一个汉字或一个英文字母都可看成是一个字符。字段是由若干个字符组成的，用来表示某一项数据，并且这些字符不能拆开，例如：一个学生的姓名，便是一个字段。若干相关字段的组合便构成了记录。例如，在如下所示的学生的信息管理文件中，每一行便是一条记录，每个记录由学号、姓名、班级、专业 4 个相关的字段组成，而每个字段，又是由若干个字符所组成。

学号	姓名	班级	专业
2001001	张三	管理 2001	工商管理
2001002	李四	管理 2001	市场营销
…	…	…	…
2001080	王小二	计算机 2002	计算机

10.1.3　文件分类

依据不同的分类标准，可将文件分为各种不同的类别，如按文件的性质可将文件分为数据文件和程序文件，按文件是否可直接运行，可将文件分为可执行文件和非可执行文件等。在 Visual Basic 6.0 中，按访问方式的不同，将文件分为顺序文件、随机文件、二进制文件 3 种类型。

顺序文件以行为单位，按顺序逐行写入数据，每次读入的数据的长度可以不一样，但必须是按顺序的写入，并且只提供第一个数据的存储的位置，而其他数据的存储位置是不知道的，因此要访问文件中的某个数据时，必须从文件的第一个数据开始按顺序查找，直到找到该数据为止。虽然顺序文件的存储方式是有规律的，并且简单，但如果要对某一个数据进行操作时，及其不方便，因此顺序文件只适用于存储有规律的并且不修改的数据。

随机文件是以记录为单位写入数据的，读写的位置可以是任意的，但数据的长度必须是固定的。正由于每次读入的数据的长度是固定的，因此定位某个数据就非常容易，访问或写入数据也就非常快，但它是以牺牲存储空间为代价的，一般适用于存储长度固定，并且不是很庞大的数据。

二进制文件是以字节为单位来存储数据的，它既可以在任意位置读写数据，也可以读写任意长度的数据，但在读写数据之前必须精确的知道数据写入文件的方式，这样就使得操作起来及其不方便。

10.2　文件管理控件

如果已经知道要访问某个文件，必须知道文件所在的驱动器、文件夹以及文件名，与此相对应的，Visual Basic 6.0 为用户提供了 3 个常用的控件：驱动器列表控件、文件夹列表控件、文件名列表控件，如图 10-2 所示。这 3 个控件既可以单独使用，也可以组合起来使用。如果将 3 个控件组合起来使用，便可以很方便地显示磁盘上的文件，如图 10-3 所示。

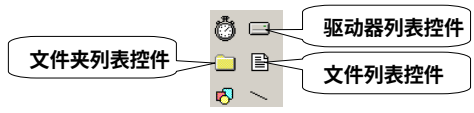


图10-2　文件控件

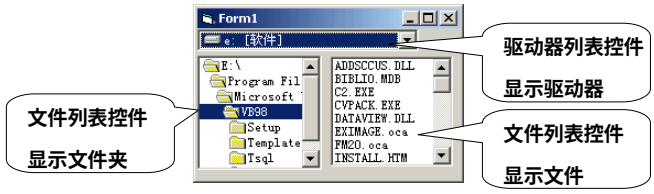


图10-3　用 3 个控件来管理文件

10.2.1　驱动器列表控件

驱动器列表控件是一个下拉式列表框，用于选择驱动器，如图 10-4 所示。和其他基本控件一样，驱动器列表控件也有一些常用属性，另外它还有一个特殊的属性“ Drive ”，该属性用于设置或返回要操作的驱动器，用户可以通过设置该属性来改变默认的驱动器。

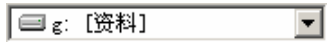


图10-4　驱动器列表控件显示驱动器

【例10-1】 向窗体中添加一个驱动器列表控件，并将 D 盘设置默认要操作的盘。

1. 新建一个工程。
2. 在工具箱中双击文件列表控件，向窗体中添加一个文件列表控件，如图 10-5 所示。

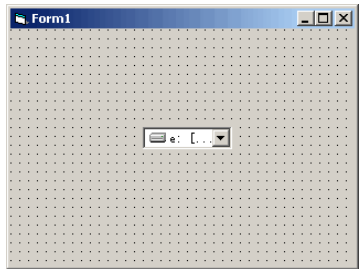


图10-5　添加文件列表控件后的窗体

3. 双击窗体，为窗体添加 Load 事件，并在代码窗口添加如下代码：

```
Private Sub Form_Load()
    Drive1.Drive = "D"
End Sub
```

4. 运行程序，驱动器 D 盘便显示在文件列表控件中。
5. 单击驱动器列表控件右端的箭头，打开下拉列表，这时所有有效的驱动器都在下拉列表中显示。单击某个驱动器，该驱动器便显示在驱动器列表框中。

由于“ Drive ”属性是不显示在属性窗口的，因此只能通过代码来设置，具体语法结构如下：

```
文件列表控件名.Drive＝"驱动器名"
```

在【例 10-1】中，通过代码：

```
Drive1.Drive = "D"
```

来改变默认的驱动器。另外在设置“ 驱动器名 ”时，不能将它设为不存在的驱动器名，例如：

某计算机只有 C、D 和 E 这 3 个驱动器，如果将“Drive”属性设为 F，程序在运行时便会出错。

在设置“Drive”属性时，驱动器名是不分大小写的，即“D”和“d”是等价的。

10.2.2　文件夹列表控件

文件夹列表控件用于显示当前驱动器上的文件夹结构。文件夹列表控件在显示文件夹时，是有一定层次的，根目录显示在最上层，然后依次缩进显示各个层次的子目录，如图 10-6 所示。

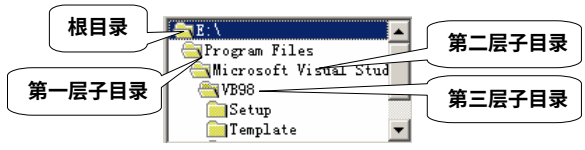


图10-6　层次结构的文件夹

【例10-2】在【例 10-1】的基础上，向窗体中添加一个文件夹列表控件，并实现以下功能：在驱动器列表控件中选定驱动器之后，在文件夹列表控件中显示该驱动器中所有的文件夹。

1. 打开【例 10-1】所建的工程。
2. 在工具箱中双击文件列表控件，向窗体中添加一个文件夹列表控件，调整控件大小至如图 10-7 所示。

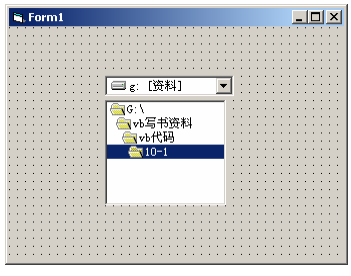


图10-7　添加文件列表控件后的窗体

3. 在窗体上，双击驱动器列表控件，为驱动器列表控件添加 Change 事件，并在【代码】窗口新增如下代码：

```
Private Sub Drive1_Change()
    Dir1.Path = Drive1.Drive
End Sub
```

4. 运行程序，在驱动器列表中显示驱动器 D 盘，在文件夹列表中显示 D 盘的根目录及第一层文件夹，如图 10-8 所示。双击其中某个文件夹便可以打开该文件夹，并以缩进的形式显示其包含的下一层文件夹。
5. 单击驱动器列表控件右端的箭头，打开下拉列表，然后单击某个驱动器，选中该驱动器，这时文件夹列表控件中所显示的文件夹也跟着改变，如图 10-9 所示。

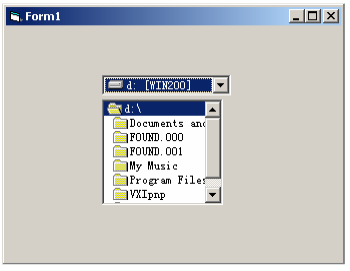


图10-8　显示默认驱动器及其所有的文件夹

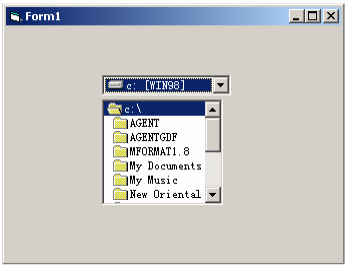


图10-9　驱动器更改所显示的文件夹也更改

在文件夹列表控件中，双击某个文件夹便可以选中该文件夹，并以图标的形式显示该文件夹，表示该文件夹被打开。当前被选中的文件夹，被文件夹列表控件的“ Path ”属性记录下来。“ Path ”属性不仅可以用来返回当前被选中的文件夹，而且还可以用于设置当前被选中的文件夹，但“ Path ”属性并不被显示在属性窗口，只能通过代码来设置“ Path ”属性。例如，在【例 10-2】中，代码

```
Dir1.Path = Drive1.Drive
```

便是用来设置文件夹列表的当前文件夹，以便文件夹列表跟随驱动器一起更改。

10.2.3　文件列表控件

文件列表控件用于显示当前路径下的部分或所有文件，如图 10-10 所示。在用文件列表控件显示文件时，必须先为所显示的文件指定详细的路径。文件列表控件的“ Path ”属性便是用来设置或返回所要显示文件的详细路径，但只能在代码中设置该属性。

【例10-3】在【例 10-2】的基础上，向窗体中添加一个文件列表控件，并实现以下功能：当在目录列表控件双击某个文件夹时，文件列表控件中便显示该文件夹中的文件；在文件列表控件中只显示扩展名为“ .exe ”的文件，并且双击某个文件，便执行该程序。

1. 打开【例 10-2】所建的工程。
2. 向窗体中添加一个文件列表控件，并调整控件的尺寸至如图 10-11 所示。

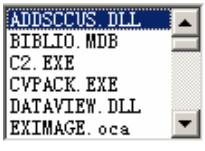


图10-10　文件列表控件

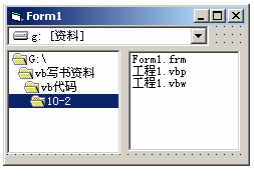


图10-11　添加文件列表控件后的窗体

3. 在窗体上，双击文件夹列表控件，为文件夹列表控件添加 Change 事件，并在【代码】窗口增加如下带底纹代码：

```
Private Sub Dir1_Change()
    File1.Path = Dir1.Path
End Sub
```

```
Private Sub File1_DblClick()
    Dim getfile As String
    '得到可执行文件的详细路径
```

```
        getfile = Dir1.Path + "\" + File1.FileName
        '运行可执行文件
        Shell getfile, 1
    End Sub

    Private Sub Form_Load()
        Drive1.Drive = "D"
        '在文件列表中只显示可执行文件
        File1.Pattern = "*.exe"
    End Sub
```

4. 运行程序，在文件夹列表控件中双击某个文件夹，选中该文件夹，此时在文件列表控件中便会显示该文件夹中的所有可执行文件，如图 10-12 所示。
5. 在文件列表中，双击某个可执行文件，便会弹出该应用程序的运行界面。

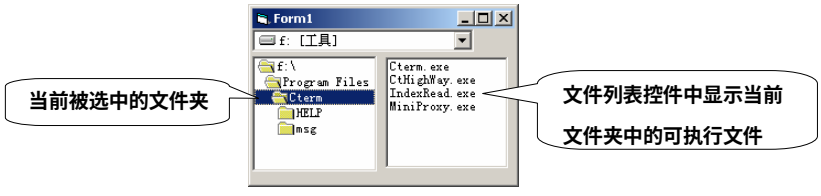


图10-12　程序运行后的界面

在默认情况下，文件列表控件中所显示的文件是当前文件夹中的所有文件，包括各种类型的文件，但有时为了查找文件的方便，在文件列表控件中只需显示某一类型的文件，这时就要用到文件列表控件的“ Pattern ”属性了。“ Pattern ”属性在属性窗口可以看到，其默认值为“ *.* ”，表示显示全部的各种类型的文件，如果要改变“ Pattern ”属性值，必须按给文件命名的形式为其赋值，既要给出文件的主文件名，还要给出文件的扩展名，但可以含有通配符“ * ”或“ ？ ”。例如：在【例 10-3】中，将文件列表控件的“ Pattern ”属性设为了“ *.exe ”，这时在文件列表控件中只显示扩展名为“ .exe ”的文件。另外，除了可以在文件列表控件中显示某一类型的文件外，还可以在其中显示某几种的文件，但中间必须以分号隔开。例如：在【例 10-3】中，如果将文件列表控件的“ Pattern ”属性设为了“*.frm;*.vbp”，则在文件列表控件中显示扩展名为“ .frm ”或“ .vbp ”两种类型的文件。

在文件列表控件双击某个文件，便选中该文件，同时该文件被文件列表控件的“ FileName ”属性记录下来。“ FileName ”属性除了可以返回文件列表控件中被选中的文件之外，还可以用来设置所要显示的文件的类型，但“ FileName ”属性只能通过代码来设置。例如，如果将“ FileName ”属性设为了如下形式

```
File1.FileName＝"*.frm"
```

则文件列表控件中只显示扩展名为“ .frm ”的文件。

由于驱动器列表控件、文件夹列表控件、文件列表控件这 3 种控件都是列表控件，因此都可以响应 Change 事件。单击驱动器列表控件的列表项时，便会引发 Change 事件，而对于文件夹列表控件、文件列表控件这两种控件，只有通过双击某一列表项才会引发 Change 事件。例如，在【例 10-3】中，单击驱动器列表控件中的列表项，文件夹列表控件中的内容就会发生改变，而只有在双击文件夹列表控件中的列表项时，文件列表控件中的内容才会改变。

文件列表控件除了可以响应 Change 事件之外，在设置“FileName”属性时，还会激发一个或多个如下事件：DblClick 事件、PatternChang 事件、PathChange 事件。

在设置“FileName”属性时，除了给出文件名之外，如果还给出了文件的详细路径，例如 File1.FileName＝”d:\programe files\Form1.frm”，则会激发 PathChange 事件；如果文件名中还含有通配符，如“*”或“？”，则还会激发 PatternChange 事件；如果文件是当前路径下的一个有效文件时，就会激发 DblClick 事件。例如，在【例 10-3】中，为文件列表控件添加 DblClick 事件，用来打开可执行文件的运行界面。

10.3 与文件操作有关的语句

前面所讲的文件管理控件只能实现显示的功能，还不能实现对文件进行打开、读写、关闭、删除等基本的操作。要想对文件实行一些基本的操作，还必须使用 Visual Basic 6.0 为用户提供的专门的语句或函数。本节将主要介绍与文件操作有关的语句和函数。

- Open 语句

Open 语句用于打开某个文件，实现对文件的输入/输出操作，对文件做任何的输入/输出操作之前都必须使用 Open 语句将文件打开。Open 语句的语法结构如下：

```
Open filename For mode [Access access] [lock] As [#]filenumber [Len=reclength]
```

各参数的说明见表 10-1 所示。

表 10-1　　Open 语句的参数说明

参数	说明
filename	必要参数。为字符串表达式，用于指定文件名，文件名还可以包括文件的详细路径
mode	必要参数。用于指定文件打开方式，可取 Append（附加方式）、Binary（二进制方式）、Input（读入方式）、Output（输出方式）、 Random （随机方式）等值。如果未指定存取方式，则以 Random（随机）方式打开文件
access	可选参数。用于说明打开的文件可以进行的操作，可取 Read（只读）、Write（只写）、或 Read Write（读写均可）等值
lock	可选参数。用于指定文件是否可共享，可取 Shared（共享）、Lock Read（锁定读取）、Lock Write（锁定写入）、和 Lock Read Write （锁定读写）等值
filenumber	必要参数。其值为一个有效的文件号，取值范围在 1 ～511 之间。使用 FreeFile 函数可得到下一个可用的文件号
reclength	可选参数。小于或等于 32 767（字节）的一个数。对于用随机访问方式打开的文件，该值就是记录长度。对于顺序文件，该值就是缓冲字符数

Open 语句为所打开的文件分配一个缓冲区以方便文件写入数据或读出数据，在用 Open 语句打开文件时，如果 filename 所指定的文件已经打开，则打开操作失败。

【例10-4】以下语句可用来以输出的方式打开 mfile.txt 文件。

```
Open "c:\mydocumen\mfile.txt" For Output As #1
```

其中“c:\mydocumen\mfile.txt”用于指定文件的详细路径，“Output”表示以输出方式打开，“1”为文件号。

- Close 语句

Close 语句用于关闭 Open 语句所打开的文件，其语法结构如下：

```
Close [filenumberlist]
```

其中参数 filenumberlist 为可选参数，其值为一个或多个有效的文件号。当 filenumberlist

为多个文件号，其语法形式如下：

#文件号,#文件号,……

如果省略参数 filenumberlist，则关闭用 Open 语句打开的所有文件。

以下代码可以用来关闭【例 10-4】所打开的文件

```
Close #1
```

其中“1”为文件号，对应于 Open 语句所使用的文件号。

- FreeFile 函数

FreeFile 函数返回下一个可供 Open 语句所使用的文件号，其语法结构如下：

```
FreeFile[(rangenumber)]
```

其中参数 rangenumber 为可选参数，用于指定文件号的取值范围，以便 FreeFile 函数返回在该范围内的下一个可用的文件号。如果 rangenumber 取为 0，表示 FreeFile 函数返回一个介于 1～255 之间的有效文件号；如果 rangenumber 取为 1，表示 FreeFile 函数返回一个介于 256～511 之间的有效文件号。

【例10-5】以下代码可以以输入方式打开 myfile.txt 文件。

```
Dim FileNum  As Interger
FileNum = FreeFile(0)
Open "c:\mydocumen\mfile.txt" For Input As #FileNum
```

在本例中，通过 FreeFile 函数返回下一个可用的文件号，因此 Open 语句所使用文件号必定是可用的，而在【例 10-3】中，所使用的文件号“1”不一定是可用的。在文件操作中，一般地，文件号都是通过 FreeFile 函数来产生。

- FileLen 函数

FileLen 函数返回一个表示文件大小的长整型数据，语法结构如下：

```
FileLen(pathname)
```

其中参数 pathname 为必选参数，为一字符串表达式，用于指定文件的详细路径。使用 FileLen 函数来获取文件大小时，可以不必先打开相应的文件。如果所指定的文件已经被打开，则 FileLen 函数返回的是文件打开前的大小。

【例10-6】以下代码可以得到文件 mfile.txt 的大小。

```
Dim FSize As Long
FSize=FileLen("c:\mydocument\myfile.txt")
```

- LOF 函数

LOF 函数返回一个表示文件大小的长整型数据，语法结构如下：

```
LOF(filenumber)
```

其中参数 filenumber 为必选参数，为用 Open 语句打开的文件的文件号。

【例10-7】以下代码也可以得到文件 mfile.txt 的大小。

```
Dim FSize As Long
Dim FileNum  As Interger
FileNum = FreeFile(0)
Open "c:\mydocumen\mfile.txt" For Output As #FileNum
FSize = LOF(FileNum)
```

LOF 函数虽然也可以得到文件的大小，但文件必须先使用 Open 语句打开。

- EOF 函数

EOF 函数返回一个布尔型或逻辑型的数据，用于测试是否已经到达文件结束部分。语法结构如下：

```
EOF(filenumber)
```

其中参数 filenumber 为必选参数，对应于用 Open 语句打开文件时所设的文件号。只有到达文件的结尾部分，EOF 才返回 True，否则返回 False。在对文件进行操作时，可使用 EOF 函数来判断是否到达文件尾部，以避免因试图在文件结尾处写入数据而产生错误。

以上的 6 种函数和语句是文件操作最常用的语句，其他与文件操作有关的语句和函数，读者可参看附表三。

10.4 文件的基本操作

虽然文件有各种不同的类型，并且文件的类型不同，其读写操作所使用的方法也不同，但不论是那种类型的文件，对其进行读写操作时，大概都要经历以下过程：

- 打开文件。
- 对文件进行读写操作。
- 关闭文件。

10.4.1 顺序文件的读写操作

要对顺序文件进行读写操作，必须先使用 Open 语句将顺序文件打开。在用 Open 语句打开顺序文件的同时，还必须指定文件打开的方式。如果要读取文件中的数据，必须以 Input 的方式打开文件，如：Open "c:\mydocumen\mfile.txt" For Input As #1；如果要将数据写入文件中必须以 Output 或 Append 的方式打开文件，如：Open "c:\mydocumen\mfile.txt" For Output As #1。

首先介绍顺序文件的读操作。

以 Input 的方式打开顺序文件后，便可以对顺序文件进行读操作了。要读取文件中的内容，可以使用 Input 函数、Line Input#语句、或 Input#语句来实现。

- Input 函数

功能：用来从顺序文件读取指定长度的字符串。

语法结构：

```
字符串变量名＝Input(number,[#]filenumber)
```

其中参数 number 为必要参数，用于指定要读取的长度；参数 filenumber 为必选参数，对应于用 Open 语句打开文件时所指定的文件号。

【例10-8】 先利用记事本创建一个文本文件并保存，文件的内容如下：

```
Option Explicit
Private Sub mnuFileOpen_Click()
Dim fName As String
CommonDialog1.Filter = "文本文件(*.txt)|*.txt"
CommonDialog1.ShowOpen
fName = CommonDialog1.FileName
If fName <> "" Then
```

```
        Open fName For Input As #1
        Text1.Text = Input(LOF(1), 1)
        Close #1
    End If
    End Sub
```

然后在 Visual Basic 6.0 中创建一个简单的文本编辑器，在文本框中显示文本文件中的内容。

1. 新建一个工程。
2. 向窗体中添加一个文本框控件，并将文本框的“MultiLine”属性设为“True”，“ScrollBars”属性设为“3－Both”。
3. 单击【工程】/【部件】菜单，打开【部件】对话框，从中选择“Microsoft Common Dialog Control 6.0”，然后单击 确定 按钮，向工具箱中添加通用对话框控件。
4. 在工具箱中双击通用对话框控件，向窗体中添加通用对话框控件。
5. 单击【工程】/【菜单编辑器】菜单，打开【菜单编辑器】，按表 10-2 的顺序新建两个菜单，【菜单编辑器】的最终样式如图 10-13 所示。

表 10-2　　菜单的属性

序号	属性	属性值
1	标题	文件(&F)
	名称	mnuFile
2	标题	打开
	名称	mnuFileOpen

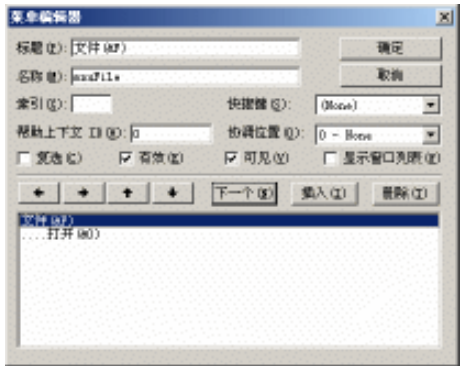


图10-13　【菜单编辑器】

6. 单击【菜单编辑器】对话框的 确定 按钮，生成菜单栏，并调整文本框的尺寸至如图 10-14 所示。
7. 在窗体上单击【文件】/【打开】菜单，为菜单添加 Click 事件，并在代码窗口添加如下代码：

```
Option Explicit
Private Sub mnuFileOpen_Click()
    Dim fName As String
    '设置文件过滤器
```

```
        CommonDialog1.Filter = "文本文件(*.txt)|*.txt"
        '显示“打开”对话框
        CommonDialog1.ShowOpen
        fName = CommonDialog1.FileName
        If fName <> "" Then
            '打开顺序文件
            Open fName For Input As #1
            '读取顺序文件中的内容，并将它显示到文本框中
            Text1.Text = Input(LOF(1), 1)
            '关闭文件
            Close #1
        End If
    End Sub
```

8. 运行程序，单击【文件】/【打开】菜单，弹出【打开】对话框，从文件列表框中选择先前所建立的文本文件，然后单击 打开(O) 按钮，文本框中便会显示文件中的内容，如图 10-15 所示。

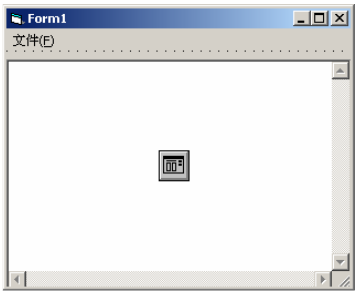


图10-14　窗体的最终样式

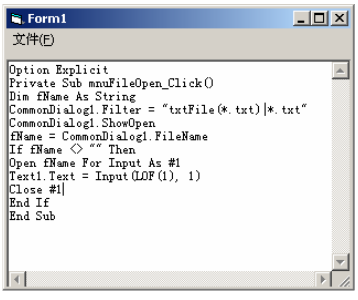


图10-15　文本框中显示文件的内容

此例中只能打开全是英文的文本文件，如果文本文件中包含有非英文的字符，则会出现打开错误。要想在文本框中显示非英文的文本文件，则可以使用以下代码来实现：

```
Text1.Text = StrConv(InputB(LOF(1), 1), vbUnicode)
```

在用 Input 函数读取文件中的字符串时，如果将参数 number 设为了文件的大小，则可以读取文件中的全部内容。例如，在【例 10-8】中，

```
Text1.Text = Input(LOF(1), 1)
```

便是先用 LOF 函数获取文件的大小，然后将它赋给 number 参数，这样文件的全部内容才会显示到文本框中，如图 10-15 所示。

- Line Input＃语句

功能：用于从顺序文件中读取完整的一行数据，并将它赋给一个字符变量。

语法结构：

```
Line Input #filenumber,varname
```

其中参数 filenumber 为必选参数，对应于用 Open 语句打开文件时所指定的文件号；参数 varname 为必选参数，用来保存从文件中读出的数据。

【例10-9】在【例 10-8】的基础上，在文本框中能够显示包括非英文字符的文件。

1. 打开【例 10-8】所建的工程。
2. 在窗体资源管理器中单击查看代码图标，打开代码窗口，并将代码窗口中的代码改为如下代码：

```
Option Explicit
Private Sub mnuFileOpen_Click()
    Dim fName As String
    Dim text As String
    Dim textbuff As String
    '设置文件过滤器
    CommonDialog1.Filter = "文本文件(*.txt)|*.txt"
    '显示“打开”对话框
    CommonDialog1.ShowOpen
    fName = CommonDialog1.FileName
    If fName <> "" Then
        '打开顺序文件
        Open fName For Input As #1
            '读取顺序文件中的内容，并将它显示到文本框中
            Do While Not EOF(1)
                Line Input #1, text
                textbuff = textbuff + text
                Text1.text = textbuff
            Loop
        Close #1
    End If
End Sub
```

3. 运行程序，单击【文件】/【打开】菜单，弹出【打开】对话框，从文件列表框中选择一个扩展名为 txt 的文件，然后单击 打开(O) 按钮，文本框中便会显示文件中的内容，如图 10-16 所示。

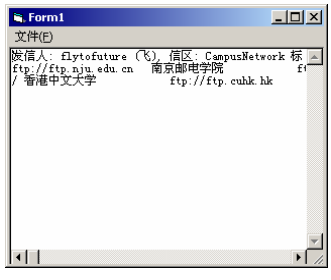


图10-16　显示包含非英文的文本文件

用 Line Input＃语句可按行来读取顺序文件中的数据，这里所说的行是指从当前位置开始，直到遇到第一个回车符或回车换行符为止，回车符或回车换行符将被跳过，而不会被附加到字符串上。例如：在【例 10-9】中，便是使用循环语句 Do While……Loop 逐行的从文件中读取数据，然后再将数据显示到文本框中。

- Input＃语句

功能：读取顺序文件中的数据，并将数据赋给变量。

语法结构：

```
Input #filenumber,varlist
```

其中参数 filenumber **为必选参数，对应于用** Open **语句打开文件时所指定的文件号；参数** varlist **为必选参数，用来保存从文件中读出的数据，变量之间以逗号相间。**

【例10-10**】 利用记事本建立一个学生成绩文件并保存，文件内容如下：**

张三 80
李四 85
王五 75
李六 80

然后创建一个工程，在工程的窗体上显示该文件中的内容。

1. **新建一个工程，并将窗体的“** AutoRedraw **”属性设为“** True **”。**
2. **向窗体上添加** 1 **个命令按钮，并将命令按钮的“** Caption **”属性设为“显示成绩”。**
3. **单击【工程】/【部件】菜单，打开【部件】对话框，从中选择“** Microsoft Common Dialog Control6.0 **”，然后单击** 确定 **按钮，向工具箱中添加通用对话框控件**  **。**
4. **在工具箱中双击通用对话框控件**  **，向窗体中添加通用对话框控件，并调整窗体大小如图** 10-17 **所示。**
5. **在窗体上双击命令按钮，为命令按钮添加** Click **事件，并在代码窗口添加如下代码：**

```
Option Explicit
Private Sub Command1_Click()
    Dim fName As String
    Dim text As String
    '设置文件过滤器
    CommonDialog1.Filter = "文本文件(*.txt)|*.txt"
    '显示“打开”对话框
    CommonDialog1.ShowOpen
    fName = CommonDialog1.FileName
    If fName <> "" Then
        '打开顺序文件
        Open fName For Input As #1
        '读取顺序文件中的内容，并将它显示到文本框中
            Do While Not EOF(1)
                Input #1, text
                Form1.Print text
            Loop
        Close #1
    End If
End Sub
```

6. 运行程序，单击【文件】/【打开】菜单，弹出【打开】对话框，从文件列表框中选择先前所建立的成绩文件，然后打击 打开(O) 按钮，窗体上便会显示文件中的内容，如图 10-18 所示。

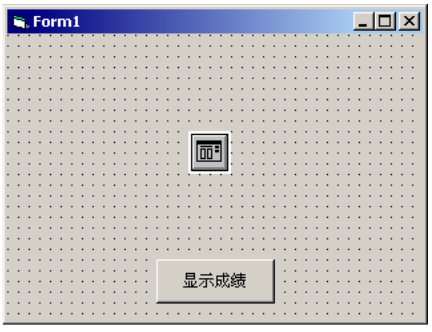


图10-17　调整尺寸后的窗体

图10-18　窗体上显示文件中的内容

在用 Input＃语句读出的数据时，将第一个非空格、非回车和非换行符作为读出数据的开始位置，第一个遇到的回车符或换行符作为数据的结束位置。例如，在【例 10-10】中，使用循环语句 Do While……Loop 逐行的读取文本文件中的数据，然后再将数据显示到窗体上。

下面介绍顺序文件的写操作。

向顺序文件中写入数据，必须先使用 Open 语句以 Output 或 Append 方式打开文件，然后再将数据写入文件，最后使用 Close 语句关闭文件。向文件中写数据可由 Print＃或 Write＃语句来完成。

- Print＃语句

功能：将格式化的数据写入顺序文件。

语法结构：

```
Print #filenumber,printlist
```

其中参数 filenumber 为必选参数，对应于用 Open 语句打开文件时所指定的文件号；参数 printlist 为可选参数，为将要被写入文件的数据列表。

【例10-11】 在【例 10-9】的基础上，将文本框中输入的内容保存到文件中。

1. 打开【例 10-9】所建的工程。
2. 为工程添加一个标题为“另存为(&S)”，名称为“mnuFileSaveAs”的菜单，【菜单编辑器】的最终样式如图 10-19 所示。
3. 在窗体上单击【文件】/【另存为】菜单，为【另存为】菜单添加 Click 事件，并在【代码】窗口增加如下代码：

```
Private Sub mnuSaveAs_Click()
    Dim fName As String
    Dim text As String
    Dim textbuff As String
    '设置文件过滤器
    CommonDialog1.Filter = "文本文件(*.txt)|*.txt"
    '显示“另存为”对话框
    CommonDialog1.ShowSave
    fName = CommonDialog1.FileName
```

```
    If fName <> "" Then
        '打开顺序文件
        Open fName For Output As #1
        '将文本框中的内容写入文件
        Print #1, Text1.text
        '关闭文件
        Close #1
    End If
End Sub
```

4. 运行程序，在文本框中输入文字（假定输入的内容为“可视化编程 Visual Basic6.0 教程”），然后单击【文件】/【另存为】菜单，打开【另存为】对话框，在对话框的【文件名】列表栏中输入要保存的文件，并记住文件保存的位置。（这里假定输入的文件名为“33.txt”，并且保存的路径为 D:\vb 资料\33.txt）。
5. 单击【另存为】对话框的 保存(S) 按钮，便将文本框中输入的内容保存到文件中去了。在 D 盘的“vb 资料”文件夹下，便可以找到“33.txt”文件，双击该文件，打开“33.txt”文件，在文件中便可以看到文本框中所输入的内容，如图 10-20 所示。

图10-19 【菜单编辑器】的最终样式

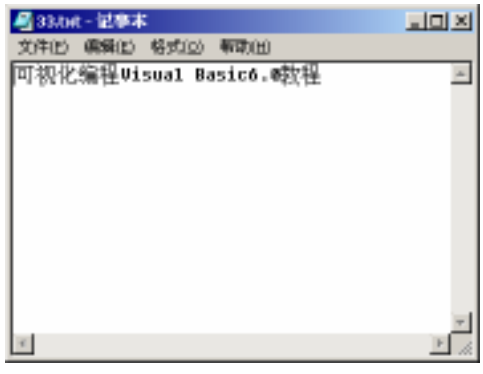


图10-20 文件中所保存的内容

当用 Print＃语句向文件中写入多个数据时，各个数据之间可以用分号相间，也可以用逗号相间。用分号相间时，表示数据以紧凑的格式写入文件，即数据之间没有分隔符；用逗号相间时，表示数据以标准的格式写入文件，数据之间以分隔符相间。例如，在【例 10-11】中，如果将写文件的代码改为如下形式：

```
Print #1, "文本框中输入的内容："; Text1.text
```

运行程序后，同样在文本框中输入“可视化编程 Visual Basic 6.0 教程”，然后单击【文件】/【另存为】菜单，保存文件。这时文件的内容变为如图 10-21 所示，数据中间无分隔符号；如果将写文件的代码改为如下形式：

```
Print #1, "文本框中输入的内容：",Text1.text
```

则文件中显示的内容如图 10-22 所示，数据之间加入了分隔符。为了便于以后读取数据，用 Print＃语句向文件中写入多个数据时，各个数据之间最好以逗号相间。

图10-21　以紧凑的格式写入数据

图10-22　以标准的格式写入数据

用 Print＃语句写入的数据一般用 Line Input＃或 Input 语句读出。

- Write＃语句

语法结构：

```
Write #filenumber,printlist
```

各个参数的说明和 Print＃语句一样。Write＃语句与 Print＃语句所实现的功能基本是一样，但它们之间还是有一定的差别的，主要区别如下：

(1) **用 Write＃语句写入多个数据时，数据之间除了可以用逗号或分号相间之外，还可以用空格相间，而 Print＃语句只能以逗号或分号相间。**

(2) **用 Write＃语句写入多个数据时，数据以紧凑的格式存入，即数据之间以逗号、分号、空格相间是等效的；而 Print＃语句既可以以紧凑的格式存入数据，而且还可以以标准的格式存入数据，即数据之间以逗号、分号相间是有区别的。**

(3) **用 Write＃语句写入数据时，数据之间自动以逗号相间，另外还自动给字符串两端加上双引号，而 Print 语句却不具有这个功能。**

如果将【例 10-11】的保存功能用 Write＃语句来实现，即将代码：

```
Print #1,Text1.text
```

改为如下形式：

```
Write #1, "文本框中输入的内容：",Text1.text
```

则得到的“ 33.txt ”文件如图 10-23 所示。

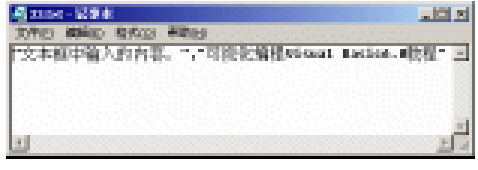


图10-23　文件中所保存的内容

用 Write＃语句写入的数据通常用 Input＃语句读出。

10.4.2　随机文件的读写

由于随机文件是以记录为单位进行操作的，每个记录包括一定数量的字段，因此在对随机文件进行操作之前，除了用 Open 语句打开文件之外，还必须先定义一个记录类型，然后再对字段所组成的记录进行读写操作。

对随机文件进行读写操作，一般都要经历以下几个过程：

- 定义记录。
- 使用 Open 语句，以随机方式打开随机文件。
- 操作记录（将文件中的记录读出或将外部记录写入文件）。
- 关闭随机文件。

【例10-12】 建立一个学生成绩录入系统。

1. 新建一个工程。
2. 向窗体上添加两个文本框、两个选项卡、两个命令按钮，按表 10-3 设置有关控件的属性，并调整控件尺寸如图 10-24 所示。

表 10-3　　有关控件属性的设置

控件	属性	属性值	说明
标签 1	Caption	学生姓名	
标签 2	Caption	成绩	
文本框 1	名称	txtName	用于学生姓名的输入
	Text	删除 Text1	
文本框 2	名称	txtScore	用于学生成绩的输入
	Text1	删除 Text1	
命令按钮 1	Caption	写入成绩	单击此按钮将学生姓名和成绩写入文件
命令按钮 2	Caption	显示成绩	单击此按钮将文件中的学生姓名和成绩分别显示在对应的文本框

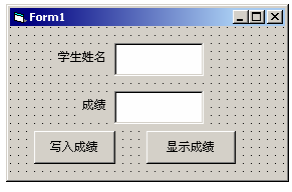


图10-24　调整后的窗体

3. 分别在窗体上，双击两个命令按钮，为命令按钮添加 Click 事件，并在【代码】窗口添加如下代码：

```
Option Explicit
'自定义记录类型
Private Type stu
    sName As String * 20
    Score As String * 4
End Type
'声明一个记录
Dim student As stu
'定义一个用于记录学生个数的变量
Dim stuNum As Integer
'定义一个用于保存记录字段的变量
```

```
Dim recordlen As Long

Private Sub Command1_Click()
    recordlen = Len(student)
    '打开文件，注意文件的路径
    Open "g:\33.txt" For Random As 1 Len = recordlen
    '处理记录
    student.sName = txtName.text
    student.Score = txtScore.text
    stuNum = stuNum + 1
    '写记录
    Put #1, stuNum, student
    '关闭文件
    Close #1
End Sub

Private Sub Command2_Click()
    '定义一个用于保存记录个数的变量
    Static numb As Integer
    recordlen = Len(student)
    '打开文件，注意文件的路径
    Open "g:\33.txt" For Random As 1 Len = recordlen
    numb = numb + 1
    '读记录
    Get #1, numb, student
    '显示从文件中读出的记录
    txtName.text = student.sName
    txtScore.text = student.Score
    '关闭文件
    Close #1
End Sub
```

4. **运行程序，在【学生姓名】栏中输入学生的姓名，在【成绩】栏中输入成绩（这里假设输入的姓名为“李四”，成绩为“85”），然后单击** 写入成绩 **按钮，便将输入学生名和成绩写入了文件“33.txt”中，如图 10-25。**

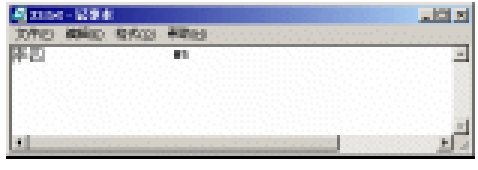


图10-25　随机文件的写入

5. 重复第 4 步，可以继续向文件“33.txt”中写入学生成绩。
6. 在写入成绩之后，将【学生姓名】栏中的学生姓名和【成绩】栏中的成绩都删除掉，然后单击 显示成绩 按钮，便可以将文件“33.txt”中的学生姓名和成绩对应地显示在文本框中，如图 10-26 所示。
7. 如果文件中写入了多个学生的成绩，重复第 6 步便可以将所有学生的姓名和成绩在文本框中对应地显示出来，只不过每次只显示一位学生的姓名和成绩。

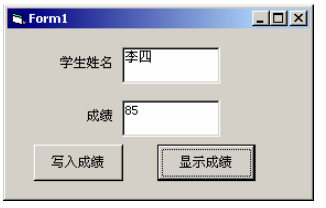


图10-26 随机文件的读出

随机文件是由固定长度的记录所组成的，每个记录又是由若干固定长度的字段所组成，因此在对随机文件进行操作之前，必须先定义一个记录变量。由于记录是用户自定义的数据类型，因此在程序的开始必须对记录进行声明。记录的声明可以用以下语法结构来完成：

```
Private Type 记录名
    字段1名 As 数据类型*长度
    字段2名 As 数据类型*长度
     …
    字段n名 As 数据类型*长度
End Type
```

在定义各个字段时不仅要给出字段的数据类型，还要给出字段的长度。记录类型的数据被声明之后，才可以来定义一个记录类型的变量。在【例 10-12】，在程序的开始，便声明了一个名为“stu”的记录类型，然后在程序中使用语句：

```
Dim student As stu
```

定义了一个记录类型的变量。记录“stu”包含两个字段，其中一个字段名为“sName”，数据类型为字符型，长度为 20 个字节，用于存放学生的姓名；另外一个字段名为“Score”，数据类型为字符型，长度为 4 个字节，用于存放学生的成绩。

定义好记录后，便可以将数据以字段的形式保存在记录中，然后对记录进行读写操作。

- 随机文件的读操作

随机文件的读操作是通过 Get#语句来完成的，Get#语句的语法结构如下：

```
Get #filenumber,[recnumber],varname
```

其中参数 filenumber 为必选参数，对应于用 Open 语句打开文件时所指定的文件号；参数 recnumber 为可选参数，为读出记录的编号，用来表示读取数据的位置。参数 varname 为必选参数，为一个有效的变量名，用来存储读出的数据。例如：在【例 10-12】中，读操作所对应的语句为：

```
Get #1,numb ,student
```

其中“1”为文件号，“numb”对应于记录的位置，“student”用于存放读出的数据。在随机文件中，每个记录都对应一个位置，第一个记录位于位置 1，第二个记录位于位置 2，依此类推，因此只要指定记录的位置，便可以得到该位置的记录。

- 随机文件的写操作

随机文件的写操作是通过 Put#语句来实现的，Put#语句的语法结构如下：

```
Put #filenumber,[recnumber],varname
```

各参数的说明和 Get#一样，这里不再重复。

在进行写操作之前，必须先按要求，将写入的数据组织成一个记录，然后才能写入随机文件中，在写入文件时，还必须为记录指定一个写入位置。例如，在【例 10-12】中，实现随机文件写操作的语句为：

```
Get #1,numb ,student
```

其中变量“numb”便是用来指定记录写入的位置，并且在将数据写入文件之前，使用以下语句：

```
student.sName = txtName.text
student.Score = txtScore.text
```

将写入的数据按记录的要求赋给记录的各个字段。

10.4.3　二进制文件的读写操作

二进制文件是以字节为单位来访问文件，允许用户随意的组织或访问数据。在用 Open 语句以 Binary 方式打开文件后，我们便可以在文件的任何位置读写任何形式的数据，因此二进制文件是 3 种文件种最为灵活的一种。

【例10-13】 在【例 10-3】的基础，实现文件拷贝的功能，即在文件列表中双击某个文件，选中拷贝的源文件，在弹出一个拷贝文件的对话框中输入拷贝的目标文件。

1. 打开【例 10-3】所建的工程。
2. 单击【工程】/【部件】菜单，打开【部件】对话框，从中选择“Microsoft Common Dialog Control 6.0”，然后单击 确定 按钮，向工具箱中添加通用对话框控件 。

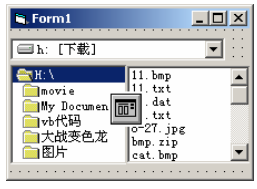


图10-27　添加通用对话框后的窗体

3. 在工具箱中双击通用对话框控件 ，向窗体中添加通用对话框控件，添加通用对话框控件后的窗体如图 10-27 所示。
4. 在窗体资源管理器中，单击查看代码图标 ，打开代码窗口。
5. 在代码窗口的对象列表栏中，选择“File1”文件列表控件，然后在事件列表框中选择“DblClick”事件，为文件列表控件添加 DblClick 事件，并在【代码】窗口增加如下代码：

```
Private Sub File1_DblClick()
    '定义一个字节变量，用于保存从文件中读出的数据
    Dim temp As Byte
    Dim i As Integer
    '获得源文件的路径
    If Right(Dir1.Path, 1) = "\" Then
       fPath = Dir1.Path + File1.FileName
    Else
       fPath = Dir1.Path + "\" + File1.FileName
    End If
    '打开文件
```

```
    Open File1.FileName For Binary As #1
    CommonDialog1.Filter = "文本文件(*.txt)|*.txt"
    '将对话框的标题设为“拷贝文件”
    CommonDialog1.DialogTitle = "拷贝文件"
    CommonDialog1.ShowSave
    '在对话框的文件名栏中输入文件名后执行文件拷贝功能
    If CommonDialog1.FileName <> "" Then
        Open CommonDialog1.FileName For Binary As #2
            For i = 1 To LOF(1)
            '逐字节的从文件 33.txt 中读出数据
                Get #1, i, temp
                '逐字节的将数据读入文件 44.txt 中
                Put #2, i, temp
            Next i
    End If
    '关闭文件
    Close #1
    Close #2
End Sub

Private Sub Form_Load()
    '将 D 盘设为默认盘
    Drive1.Drive = "D"
    '将文本文件设为默认的文件类型
    File1.Pattern = "*.txt"
End Sub
```

6. 运行程序，在文件列表中双击其中的某个文本文件选中源文件（注意：要记住这个文件所在驱动器和文件夹），并且同时会弹出一个【拷贝文件】的对话框，在对话框的文件名列表中输入目标文件的文件名，并记住文件所在的位置，然后单击 保存(S) 按钮，在文件列表框中所选中的源文件便会拷贝到在【拷贝文件】对话框中所输入的目标文件中。
7. 在计算机的硬盘中找到这个文件，然后分别双击文件打开文件，这时便会看到两个文件中的内容是一样的，如图 10-28，10-29 所示（这里假设在文件列表控件中选择的是“11.txt”文件，在对话框输入的是“22.txt”）。

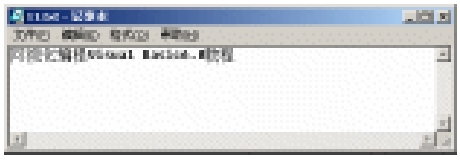


图10-28　源文件“11.txt”中的内容

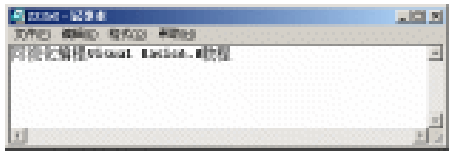


图10-29　目标文件“22.txt”中的内容

- 二进制文件的读操作

二进制文件的读操作和随机文件所使用的语句是一样的，都是使用 Get#语句，但参数的含义有所不同。参数 recnumber 表示的不是第几个记录，而是第几个字节，另外参数 varname 可以是任意类型的变量，而不仅仅可以是记录型变量，但通常把参数 varname 定义为字节型变量。例如，在【例 10-13】中，使用以下代码逐字节的从源文件中读出数据：

```
Get #1, i, temp
```

其中变量 i 是用来指定字节的位置，变量 temp 为存放读出数据的字节变量。

二进制文件除了可以使用 Get#语句读出数据之外，还可以使用 Input#语句来读出数据。

- **二进制文件的写操作**

二进制文件的写操作是通过 Put#语句来完成，例如，在【例 10.13】中，如下代码：

```
Put #1, i, temp
```

便是用来将从源文件中读出的字节数据写入目标文件的指定字节位置。

随机文件只能对定长的记录进行读写操作，而二进制文件可以对不定长的记录进行读写操作，这样便可以节省大量的磁盘空间。例如，在【例 10-12】中，在程序的开始声明了一个定长的记录：

```
Private Type stu
    sName As String * 20
    Score As String * 4
End Type
```

在程序执行时，即使写入的数据没有 24 个字节的长，系统也会为其分配 24 个字节的存储空间，因此这样就会浪费大量的磁盘空间。如果使用二进制文件来存储以上记录时，便可以按以下方式来声明记录类型：

```
Private Type stu
    sName As String
    Score As String
End Type
```

这样系统便会根据数据的长度为其分配相应的存储空间。

10.4.4　文件的其他操作

文件除了读写这两种基本操作之外，还有文件的删除、拷贝、以及改名等其他基本操作。

- **文件的删除**

语法结构：

```
Kill 文件名
```

功能：用来删除“文件名”所指定的文件。

说明：在指定“文件名”时，必须给出文件的详细路径，并且文件名中还可以含有通配符“*”和“？”。例如：

```
Kill  "d:\myfile\*.txt"
```

语句便可以删除 D 盘“myfile”文件夹下的所有文本文件。另外在使用 Kill 语句删除文件时，并不会像在 Windows 系统中删除文件那样，会给出一个提示信息，因此使用该语句时必

须十分小心，最好在删除文件之前给出相应的提示信息。

不能使用 Kill **语句来删除已经打开的文件，否则会产生错误。**

- **文件的拷贝**

语法结构：

```
FillCopy  源文件名,目标文件名
```

功能：将源文件中的内容拷贝到目标文件中去。

说明：在指定目标文件和源文件时，最好是给出详细的路径，并且文件名中不能含有通配符。例如：

```
FileCopy "d:\myfile\11.txt", "c:\mydocument\22.txt"
```

语句便可以将 D **盘“** myfile **”文件夹下的“** 11.txt **”文件拷贝到** C **盘“** mydocument **”文件夹下，并以“** 22.txt **”来命名。**

不能使用 FileCopy **语句来拷贝已经打开的文件。**

- **文件的改名**

语法结构：

```
Name 原文件名 As 新文件名
```

功能：将“原文件名”改名为“新文件名”

说明：“原文件名”必须是已经存在的文件名，而“新文件名”必须是一个不存在的新的文件名，并且两个文件的路径必须是一样的。如果“新文件名”的路径与“原文件名”的路径不一样时，则将“原文件”移动“新文件名”所指定的路径下，并将文件改名为“新文件名”。例如：

```
Name "d:\myfile\11.txt" As "d:\myfile\22.txt"
```

语句便可以将文件“ 11.txt **”改名为“** 22.txt **”。**

```
Name "d:\myfile\11.txt" As "c:\mydocument\33.txt"
```

语句便可以将文件 11.txt **从** D **盘“** myfile **”文件夹下移动** C **盘“** mydocument **”文件夹下，并改名为“** 33.txt **”。**

不能使用 Name **语句来更改或移动已经打开的文件的名字。**

10.5 小结

Visual Basic 6.0 **为用户提供了强大文件处理功能，使用** Visual Basic 6.0 **所提供的文件处理控件或语句、函数，便可以让应用程序具有数据保存和打开的功能。由于不同的文件类型，所使用的操作方法是不一样的，因此编写一个完整的文件处理的程序是一项比较复杂的任务，文件处理过程是最为基础的部分。**

本章主要介绍了以下内容：

- **文件的基本概念。**

- 文件的分类。
- 3 种常用的文件管理控件。
- 与文件处理相关的一些基本语句和函数。
- 3 种文件的读写操作。
- 文件的删除、拷贝、改名等基本操作。

10.6　习题

一、 填空题

1. 文件是由_____组成，而_____是由字段组成，字段是由_____组成。
2. Visual Basic 6.0 按访问文件的方式的不同将文件分为_____、_____、_____3 种类型。其中_____文件是以行为单位来访问文件的，_____文件是以记录为单位来访问文件的，_____文件是以字节为单位来访问文件的。
3. 在改变默认的驱动器，可以通过设置驱动器控件的_____属性；文件夹列表控件的当前路径被_____属性所记录；文件列表控件中被选中的文件被_____属性所记录。
4. 使用 Open 语句打开文件，可以_____、_____、_____、_____、_____5 种方式打开文件。
5. 可以使用_____函数来获取下一个可用的文件号；可以使用_____函数来检验是否到达文件的结尾部分；关闭文件可以使用_____语句。
6. 顺序文件可以通过_____语句或_____语句将数据写入文件，而读取文件中的数据可以使用_____语句、_____语句或_____函数来实现。随机文件和二进制文件的读操作可以通过_____语句来实现，写操作可以通过_____语句来实现。
7. 要删除掉一个文件，可以使用_____语句；要将一个文件改名可以使用_____语句；要拷贝一个文件可以使用_____语句。

二、 选择题

1. 下面（）文件命名的方式是错误的。
   A. d:\myfile\11.txt　　B. d:\11.txt　　C. d:\myfile\11　　D. d:\myfile\11\11.txt
2. 要想获得使用 Open 语句所打开的文件的大小可以使用（）。
   A. LOF 函数　　B. Len 函数　　C. FileLen 函数　　D. EOF 函数
3. （）只能从顺序文件中读出英文字符，非英文字符不能读出。
   A. Input＃语句　　B. Input 函数　　C. Line Input＃语句　　D. Get 语句
4. 二进制文件除了可以使用 Get#语句读出数据之外，还可以用（）来读出数据。
   A. Print 语句　　B. Input 函数　　C. Line Input＃语句　　D. Input＃语句
5. 如果要将 D 盘“ myfile ”文件夹下名为“ 11.txt ”的文件拷贝到 C 盘“ myfile ”文件夹下，并改名为“ 22.txt ”，则下面的代码（）是正确的。
   A. Name "d:\myfile\11.txt" As "c:\myfile\22.txt"　　B. Name "11.txt" As "22.txt"
   C. Name "11.txt" As "c:\myfile\22.txt"　　D. Name "d:\myfile\11.txt" As "22.txt"

三、 问答题

1. 记录、字段、字符之间的关系？
2. 文件的读写操作一般要经历哪几个过程？
3. 使用 Print#语句和 Write#语句将数据写入顺序文件中有什么区别？
4. 随机文件的读写操作要经历哪几个过程？

四、 编程题

1. **使用 3 个文件控件，编写一个简单的文件显示的界面，并且在文件列表控件中只显示文本文件（*.txt）。**
2. **编写一个简单文本编辑器，能实现简单的打开和保存功能。**

第11章　Visual Basic 6.0 数据库编程

本章在讲解了一定的数据库基本知识之后，详细地讲解了使用 Visual Basic 6.0 进行数据库编程的过程。通过本章的学习，学生能够创建数据库，并能对数据库进行维护。

本章学习目标

- 数据库的基本组成及数据库的设计过程。
- 数据库设计标准语言 SQL。
- Visual Basic 6.0 可视化数据管理器。
- 使用控件访问数据库。
- 数据库应用程序设计方法。

11.1　数据库的基本知识

随着信息处理系统的大量推广和应用，数据库发展成为了一门专门的学科，数据库应用技术越来越成为人们普遍关注的问题。对于大量的数据来说，使用数据库存储比通过文件存储有更高的效率。

Visual Basic 6.0 提供了功能强大的数据库管理功能，这里只做简单的介绍。

数据库是按一定的结构和规则组织起来的相关数据的集合，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。

11.1.1　数据库的基本组成

数据库中数据的组织形式有多种，最近几年来，关系模型已经基本成为数据库设计的标准（这里介绍的数据库知识都是指的关系数据库）。所谓关系数据库就是将数据表示为表的集合，通过建立简单表之间的关系来定义结构的一种数据库。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。

一个数据库含有各种成分，包括表、记录、字段、索引等。下面介绍关系数据库中的一些组成部分的基本概念。

(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。

表是一种按行与列排列的相关信息的逻辑组，实际上就是一个二维表格。例如，表 11-1 所示的是一个职工信息表，其中包含职工号、姓名、性别、年龄、政治面貌、职务及特长等信息。

表 11-1　　职工信息表

职工号	姓名	性别	年龄	政治面貌	职务	特长
1234	李铭	男	23	团员	技术员	计算机
3255	胡兵	男	43	党员	处长	跳舞

续表

职工号	姓名	性别	年龄	政治面貌	职务	特长
3466	赵明	男	28	党员	科长	文艺、棋牌
1236	邓丽	女	20	团员	技术员	文艺、美术
4356	黎平	女	29	党员	会计	篮球
2351	孙灵	男	25	群众	科员	棋牌

(2) **记录**

与每一个职工有关的信息被存放在表中，被称为一个记录（Record），即表的每一行数据就是一个记录，而且一般来说，表中的记录必须是惟一的。

(3) **字段**

表中的每一列称作一个字段（Field），描述了它所含有的数据。表是由其包含的各种字段定义的，每个字段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惟一的。

(4) **关键字**

关键字就是表中的某个字段（或多个字段），它（们）为快速检索而被索引。关键字可以是惟一的，也可以是非惟一的，取决于它（们）是否允许重复。惟一关键字可以指定为主关键字，用来惟一标识表的每行。每个表都应该有一个主关键字，它是记录的惟一标识符。例如，在职工信息数据库中，可以将职工号作为主关键字。对于每个记录来说，主关键字必须具有一个惟一的值，并且主关键字不能为空值。

(5) **索引**

数据库建成之后，为了提高访问数据库的效率，可以对数据库使用索引。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。当数据库较大时，为了查找指定的记录，则使用索引和不使用索引的效率有很大差别。

索引实际上是一种特殊类型的表，其中含有关键字段的值（由用户定义）和指向实际记录位置的指针，这些值和指针按照特定的顺序（也由用户定义）存储，从而可以以较快的速度查找到所需要的数据记录。

11.1.2 数据库设计过程

要建立一个数据库，就要先有一个设计方案。这就好像在建造一栋房屋之前，必须有一个设计蓝图。像大多数工作一样，要创建一个成功的数据库，有一个好的设计方案是非常重要的。

怎样才能设计一个好的方案呢？在设计数据库应用程序时，既要注意创建能发挥数据库最优性的程序代码，也要安排好数据库存放的物理布局和逻辑布局。一个好的数据库设计方案应包括以下几点：

- **能用最少的时间定位特定记录。**
- **以最有效的方式存放数据，以节省存储空间。**
- **能使数据更新以尽量简单的方式进行。**
- **在包含程序所需的新功能时应有足够的灵活性。**

11.1.2.1 设计原则

只有事先把握一定的原则，才能设计一个合理的数据库。有时各个设计原则间是相互排斥的，这时就需要权衡利弊，寻求个别原则的最佳结合点。设计一个数据库的基本原则如下：

- 能快速定位单个记录。
- 能删除多余的数据。
- 易于改进数据库。
- 易于维护数据库。

11.1.2.2 设计步骤

一个标准的数据库方案创建过程，一般应包括以下几个主要步骤。

1. 应用程序的建模

在为应用程序建模时，应先确定应用程序要执行的任务。例如，要维护一个学生成绩表，必须知道想要创建的学生信息表和课程信息表。接下来就要建立功能设计说明书。对于一个要创建的工程来说，即使用户知道所要执行的任务，将这些任务写在说明书中也是很有必要的，因为说明书可以帮助用户将注意力集中在程序所要执行的任务上。而对于其他人来说，功能设计说明书就是程序将要包含的功能和协约。同时，这个设计说明书是在一个预定时期内必须完成的日程表。

2. 确定应用程序所需的数据

在完成应用程序的建模后，就可以开始确定需要什么数据了。在前面提到的学生成绩应用程序中，创建数据库需要包含学生的学号、姓名、性别、年龄、所选课程号、课程名、成绩、任课教师号、任课教师名等。

3. 将数据组织成表

数据在数据库中的组织方式，是设计好一个数据库的重要方面。数据组织的基本原则就是要使数据库易于维护。在一个数据库中，数据存放在一个或多个表中。对一个复杂的数据系统，通过将数据存放在多个表中，并通过在表间建立关联来进行管理，往往能更有效地利用空间。

尽管在组织表方面并没有按部就班的严格规则，但要设计一个合理高效的数据库，还应参照以下常用的基本规则：

- 为每一个表确定一个标题，确保表中所有的数据都与这个主题有关。

根据这个表的主要标题，可以确定应该将哪些数据放入表中。例如，学校想要建立有关学生与老师的表，由于两者都是学校的人员，可以将它们放入一个表中。如表 11-2 所示。

表 11-2　　学生和教师信息表

姓名	性别	身份	教师号	学号	班号
黎明	男	学生		04001	001
胡兵	男	教师	99006		
康丽	女	学生		04012	006
赵明	男	学生		04652	0013
孙立	男	学生		04221	0010
李晓	女	教师	96210		

- 如果一个数据表中有很多记录都出现了空白单元格，则应考虑把这个数据表分成两个或多个类似的表。

对于表 11-2（表的序号依次后推），再进一步地分析，除了姓名和性别是两者共有的信息外，有用的学生信息还有学号、班号等，老师信息还有教师号等。如果将两者放入一个表中，必然就会有很多空条目。这样不但会造成大量的空间浪费，还会导致事务处理地缓慢，因为程序不得不跳过表中的很多记录。这时就应该分别为学生和老师各建一个表，如表 11-3 和 11-4 所示。虽然表的数目增加了，但存储空间却大大减少了。

表 11-3　　学生信息表

姓名	性别	学号	班号
黎明	男	04001	001
康丽	女	04012	006
赵明	男	04652	0013
孙立	男	04221	0010

表 11-4　　教师信息表

姓名	性别	教师号
胡兵	男	99006
李晓	女	96210

很显然，第一个表中的空格占据了存储空间，却没有内容，实际上也就是浪费了存储空间。而后两个表则弥补了这一缺陷。因此，对于某一特定信息，如果它对于许多记录来说导致了空间的浪费，就应该考虑将其放入另一个表中。所以说，节约空间是建表的一个基本原则之一。

- 如果一些记录中的信息是重复的，应该把这些信息移入另一个表中，并设置两个表之间的联系。

如果使每一条信息都能在数据库中只显示一次的话，那么这个数据库无疑是非常成功的。然而，这实际上是不太可能做到的，但应尽量避免冗余数据。

下面将通过一个学生成绩表的事例加以说明。

对于每一个学生，都需要学号、姓名、性别、年龄，同时也需要课程号、课程名、任课老师和成绩。用表 11-5 储存这些信息，如下。

表 11-5　　学生成绩信息表

学号	姓名	性别	年龄	课程号	课程名	任课教师	成绩
04001	黎明	男	16	02	数学	胡兵	94
04012	康丽	女	15	01	语文	李晓	88
04012	康丽	女	15	03	英语	王东华	93
04001	黎明	男	16	03	英语	王东华	96
04652	赵明	男	15	02	数学	胡兵	52
04012	康丽	女	15	02	数学	胡兵	91
04652	赵明	男	15	03	英语	王东华	53
04221	孙立	男	16	01	语文	李晓	68
04652	赵明	男	15	01	语文	李晓	67

续表

学号	姓名	性别	年龄	课程号	课程名	任课教师	成绩
04221	孙立	男	16	02	数学	胡兵	95
04001	黎明	男	16	01	语文	李晓	78
04221	孙立	男	16	03	英语	王东华	87

可以看出，表中有很多信息重复。这一方面导致了空间的浪费，一方面使数据维护复杂化，且使数据的准确性受到影响。如果某个学生信息发生变化，因为学生信息可能在很多地方出现，所以更改时总有出现遗漏的时候。

现在尝试将学生信息与课程信息分离开来。用学号来惟一标识一个学生，也就是说只要学号不同，就是不同的学生，即使学生的其他信息相同（如同名同姓），具有这种特征的信息称为主关键字。经过这种处理，学生信息除主关键字外，都只显示一次。当其年龄改变时，只需改一条记录即可。同时我们也将课程独立起来，最后得到 3 个表（表 11-6、11-7、11-8）如下：

表 11-6　　学生信息表

学号	姓名	性别	年龄
04001	黎明	男	16
04012	康丽	女	15
04652	赵明	男	15
04221	孙立	男	16

上表中每一位学生都有一个学号，在这个表中，学号绝对不会重复，这就是主关键字的惟一性。利用主关键字的这个性质，可以检查表的正确性，只要发现一个表中有重复的主关键字，就可以判断表错。

表 11-7　　课程信息表

课程号	课程名	任课教师
01	语文	李晓
02	数学	胡兵
03	英语	王东华

表 11-8　　学生成绩表

学号	课程号	成绩
04001	02	94
04012	01	88
04012	03	93
04001	03	96
04652	02	52
04012	02	91
04652	03	53

续表

学号	课程号	成绩
04221	01	68
04652	01	67
04221	02	95
04001	01	78
04221	03	87

上面 3 个表间是有联系的，如果想知道某个学生的信息，可通过学号把学生信息表与学生成绩表联系起来，然后由课程号便可找到课程信息，就好像原来一个表一样。可以看出其中主关键字起了至关重要的作用，这也是主关键字的功能之一。

- 可以通过建立子表及查询表的方式，减少数据量，并增加数据输入的准确性。

子表（Child Table）是一个所有条目共享并存放在另一个表中的公用信息表。而存放公用信息的表，则称为父表。例如，一个班有班号、系号、班主任，这些信息对班上的每一个成员都是一样的。由此可建立一个有关班级信息的父表。如表 11-9 所示。

表 11-9　　班级信息表

系号	班号	班主任
001	0412	王春华
003	0425	肖晓
010	0430	李兵
010	0433	李光

每个班的各个成员，由不同学号、姓名、性别、年龄，依次可建立相对上面父表的子表。如下表 11-10 所示。

表 11-10　　学生信息表

学号	姓名	性别	年龄	班号
04001	黎明	男	16	0412
04012	康丽	女	15	0412
04652	赵明	男	15	0433
04221	孙立	男	16	0430

由此可见用查询表存储信息可以防止冗余数据，并提高数据输入的准确性。

4.　在表与表之间建立关联

对数据库进行规范化处理后，有时会产生把数据从一个表移到另一个表的需要，这可以通过在表间建立关系来实现。表之间的关系通过关键数据来建立。

关键数据分为主关键字和外部关键字。其中主关键字是对数据表内一个记录进行惟一标识的信息，而外部关键字是把一个记录与另外某个数据表中的关键字联系起来的信息。

各个数据表通过关键字段相互联系。在一个数据库中，关键字段用来标识出某个记录。它可以是含有特殊意义的数据，也可以由编程人员建立。要求关键字段必须是惟一的，即一个关键字必须能惟一地标识某一个记录，只与某一个记录相对应。

例如，在学生信息表中由于学生太多，可能存在同名同姓的情况，因此不宜用姓名作为关键字段，学号则完全可能惟一地标识一个学生。如表 11-6。

外部关键字是表之间连接的纽带。根据数据表中某个记录与另外一个表中能发生关系的记录个数，可以分为“一对多”关系与“多对多”关系。如表 11-8 为“多对多”关系，学号和课程号为主关键字，同时学号、课程号又分别为外部关键字。

5.　为数据设置索引。

当信息输入到一个表中时，记录通常是按增序存放。这样的顺序称为数据的物理顺序。然而，在查看或处理数据时，常常不按输入顺序进行，而是按照自己定义的逻辑顺序。在表中查找某一特定记录，如果按照物理顺序进行非常费时。例如，如果字典把所有的词汇杂乱无章的罗列在一块，当要查找某一个词时，就不得不从开头一个一个地对照，直到查出那个单词为止，这样很麻烦。

索引（Index）就提供了以特殊顺序存放表中信息的方法。索引也是一种表，它包含了数据中每一个记录的键值（通常来源于一个或多个字段的值）。索引本身是以一种特殊逻辑顺序存放的，同时还包含告诉数据库引擎实际存放位置的指针。

索引的存在，使快速地查询和提取数据成为可能。以查字典的例子来说明。当将词汇按字母顺序排序后，要查找一个单词，根据首字母所在的页码，再依次根据其后面字母便可快速定位。

正如词汇可按首字母归类排序，也可按照词性归类一样，一个表可以有很多索引，以便根据这些索引提供不同的数据组织方式。采取什么样的索引，主要由目的决定。例如学生信息可以按学号、成绩等来建立索引，这都由不同的目的决定。

索引可以分为单键索引和多键索引。

单键索引（Single-key Index）是最普通的索引，它是基于表中的一个字段的值。如将学生成绩表 11-8 按学号排序后如表 11-11 所示。

表 11-11　　按学号排序后的学生成绩表

学号	课程号	成绩
04001	02	94
04001	03	96
04001	01	78
04012	01	88
04012	03	93
04012	02	91
04221	01	68
04221	02	95
04221	03	87
04652	02	52
04652	03	53
04652	01	67

显然，排序后的学生成绩表显得规则多了。但对于相同学号的学生，其课程仍然是杂乱无章的，可以对其进行再一次的排序，这就是下面要介绍的多键索引。

多键索引（Multiple-key Index）是基于表中两个或两个以上的值。如上例，在同一学号的学生下的课程号再排一次序，结果如表 11-12 所示。

表 11-12　　按学号与课程号排序后的学生成绩表

学号	课程号	成绩
04001	01	78
04001	02	94
04001	03	96
04012	01	88
04012	02	91
04012	03	93
04221	01	68
04221	02	95
04221	03	87
04652	01	67
04652	02	52
04652	03	53

可以看出学生成绩表更规则了。在这里用了两个索引键，一个是学号，一个是课程号。在处理这类问题时，要处理好索引键的先后次序。

6. 为数据设置有效性规则

在应用程序中，每个数据都有其各自的数据类型或取值范围，数据库中的每个数据也有其自己的有效性规则，在 Visual Basic 6.0 中，其数据的类型与变量的类型完全相同，另外用户还可以根据具体情况确定各数据的长度。

7. 为应用程序创建和保存必需的查询方式

当进行数据规范化时，一般要将相关的信息放入多个表中。当访问数据时，有时想在一处看到所有表中的信息。这可以通过记录集来实现，而记录集可用 SQL 语句来创建，或将 SQL 语句放在用来创建记录集的 Database 对象的 OpenRecordSet 方法中。然而，比较理想的做法是将 SQL 语句存放为数据库的一个查询。这种查询方法有如下优点：

- 可以容易地在程序的多个位置或多个程序中使用 SQL 语句。
- 仅需在一处修改 SQL 语句。
- 保存查询的方式比从代码中解释语句的方式运行速度快。
- 将应用程序上载到一个客户/服务器环境中比较容易。

SQL 语句将在下节介绍。

8. 设计方案的复查

在数据库设计初步完成之后，一定要对数据库进行复查核对，使其能够安全有效地运行使用。

11.1.3 数据库设计标准语言 SQL 简介

SQL 语言是一种用于和关系数据库进行交互通信的计算机语言。SQL 是数据库管理系统的一个重要组成部分，是用户与数据库引擎进行通信的语言和工具。当用户想检索数据库中的

数据时，就可以用 SQL 语言发出此请求，DBMS 对 SQL 请求进行处理，检索到所要求的数据，并将其返回给用户。这个向数据库请求并得到数据的过程称为数据库查询，这就是 SQL 语言（Structure Quary Language）这一名称的由来。

如今，SQL 语言已不仅仅是一个查询工具，它已成为可以对关系数据库进行组织、管理和检索的主要工具。以下是 SQL 语言的主要特点：

- SQL 是一种交互式查询语言：用户可以通过键入 SQL 命令来检索数据，并将其显示在屏幕上。这是一种简单易用的数据查询方法。
- SQL 是一种数据库编程语言：程序员可以使用 SQL 命令存取数据库中的数据。用户程序和数据库应用程序都采用这种方法进行数据操纵的。
- SQL 是一种数据库管理语言：数据库管理员可以利用 SQL 来定义数据库组织结构，控制数据存取等，从而实现对大型数据库系统的管理。
- SQL 是一种客户/服务器语言：个人计算机利用 SQL 与存放有共享数据的服务器通过网络进行交互通信。目前有许多应用都采用这种客户/服务器模式，以减轻网络的拥挤状况，使 PC 机和服务器各显其能。
- SQL 是一种分布式数据库语言：分布式 DBMS 利用 SQL 将数据分配给多台通过网络连在一起的所构成的分布式计算机系统。每台计算机上的 DBMS 都用 SQL 和其他计算机通信，发送数据存取请求。
- SQL 是一种数据库网关语言：在混用不同 DBMS 产品的网络中，SQL 通常被用来做网关，以使这些 DBMS 间能相互通信。

本书只介绍 SQL 语言中最常用最简单的一种功能——交互式查询功能。查询是 SQL 语言的核心，SQL 语言只提供惟一一个用于数据库查询的语句，即 SELECT 语句。用于表达 SQL 查询的 SELECT 语句是功能最强也是最复杂的 SQL 语句，它提供了很多选项和使用方法。SELECT 语句的命令格式如下：

```
SELECT[ALL|DISTINCT]<关系表字段(表达式)列表|*>
FORM[关系表名(别名)列表]
[WHERE 查询条件]
[GROUP BY 分组要求]
[HAVING 分组搜索条件]
[ORDER BY 排序要求]
[INTO<新关系表名>]
```

SELECT 查询语句是由 7 个子句构成，其中 SELECT 和 FROM 子句是一个完整 SELECT 查询语句必须要有的，其他的子句可以根据具体要求任选。上述的每个子句功能说明如下：

(1) SELECT 子句

列出所有要求 SELECT 语句进行检索的数据项。这些项可能取自数据库中关系表的列，也可以是 SQL 在执行查询时需要进行计算的表达式。这里的“ALL”和“DISTINCT”选项，表示查询出的结果中是否容许有内容重复的行出现，缺省时是“ALL”选项，表示容许有内容重复的行出现，而“*”则表示查询出指定关系表中的所有列。

(2) FROM 子句

FROM 子句列出包含所要查询的数据关系表。

(3) WHERE 子句

WHERE 子句告诉 SQL 只查询某些关系表中满足一定要求的行的数据，查询要求由 WHERE 子句中的查询条件确定。

在 WHERE 子句中条件表达式可以使用以下的关系运算符：=（等于）、>（大于）、<（小于）、>=（大于等于）、<=（小于等于）、<>（不等于）、LIKE。LIKE 是一个功能强大的运算符，它可以用来查找包含特定字符串的数据，可以用星号“*”来匹配任何可能的字符，“*”可以出现在指定字符的前面或者后面等，也可以是逻辑运算符，例如：AND（与）、OR（或）、NOT（非）等。

(4) GROUP BY 子句

GROUP BY 子句指定当前查询是汇总查询，即不是根据对每行产生一个查询结果，而是对相似的行进行分组，然后再对每组产生一个汇总查询结果。

(5) HAVING 子句

HAVING 子句告诉 SQL 只对由 GROUP BY 所得到的某些行组的结果进行过滤，选择出满足分组条件的分组。

(6) ORDER BY 子句

ORDER BY 子句确定是否将查询出的结果按一列或多列中的数据进行排序，缺省时是不排序的。

(7) INTO 子句

INTO 子句确定是否将查询出的结果存入一张新的关系表中，缺省时只将查询出的结果显示在屏幕上。这是非标准 SELECT 语句中的子句，但目前绝大多数实际应用的 SQL 数据库系统的 SQL 语言提供了这一选项。

以上一节我们所建立的 11-6 学生信息表、11-7 课程信息表以及 11-8 学生成绩表这 3 个表为例，讲解如何使用 SELECT 查询语句进行数据库查询。

【例11-1】 在学生信息表中，查找性别为“男”的同学的姓名。

```
SELECT 姓名
FROM 学生信息表
WHERE 性别=男
```

【例11-2】 查询课程号为“02”的科目的考生学号及成绩。

```
SELECT 学号,成绩
FROM 学生成绩表
WHERE 课程号=02
```

【例11-3】 查询姓名为“孙立”的学生的数学成绩。

```
SELECT 成绩
FROM 学生信息表,课程信息表,成绩信息表
WHERE 学生信息表.姓名="孙立"
    AND 课程信息表.课程名="数学"
    AND 学生信息表.学号=课程信息表.学号
    AND 课程信息表.课程号=成绩信息表.课程号
```

【例11-4】 查询考试成绩小于 60 分的学生学号及课程号，并按学号排序。

```
SELECT 学号,课程号
```

```
FROM 学生成绩表
WHERE 成绩 < 60
ORDER BY 学号 ASC
```

另外 SQL 语言还提供了 6 种处理函数，帮助完成不同的处理工作。这 6 种处理函数是：

- SUM()：用于计算表中某一列的总和。
- AVG()：用于计算表中某一列的平均值。
- MIN()：用于选择表中某一列的最小值。
- MAX()：用于选择表中某一列的最大值。
- COUNT()：用于计算表中某一列中值的个数。
- COUNT(*)：用于计算某张表的行数。

【例11-5】 统计参加课程号为“01”的考试的考生人数。

```
SELECT COUNT(姓名)
FROM 学生成绩表
WHERE 课程号=01
```

【例11-6】 统计年龄最大的学生姓名。

```
SELECT 姓名
FROM 学生信息表
WHERE 年龄>=MAX(年龄)
```

至此，有关 SELECT 查询语句所有的基本使用方法就介绍完了，作为总结，下面给出 SELECT 查询处理的工作过程描述：

- 形成 FROM 子句所指定的目标表（表的乘积）。如果 FROM 子句仅指定一个表。则此表为目标表。
- 如果有 WHERE 子句，则将其搜索条件用于目标表中的每一行，保留使搜索条件为“TRUE”的行，剔除使搜索条件为“FALSE”或“NULL”的行。若 WHERE 子句中包含另一个 SELECT 查询语句，先去完成另一个 SELECT 查询语句操作，然后将其查询结果用到本次查询的搜索条件中，进行检测。
- 若有 GROUP BY 子句，则对目标表中所保留的行进行分组，以使得每组中的各行具有相同的分组列的列值。
- 若有 HAVING 子句，则将其搜索条件作用于每个行组，保留那些使搜索条件为“TRUE”的组，剔除使搜索条件为“FALSE”或“NULL”的行。若 HAVING 子句中包含另一个 SELECT 查询语句，先去完成另一个 SEKECT 查询语句操作，然后将其查询结果用到本次查询的 HAVING 子句搜索条件中，进行检测。
- 对于所保留的每一行(或行组)，计算由 SELECT 子句所指定的每项值，对于简单的列指定，取当前行(或行组)中该列的值；对于列函数，若有 GROUP BY 子句，则用当前行组作为其参数；否则，用所有行作为其参数；剔除 SELECT 子句中没有指定的各列。
- 若 SELECT 子句后选择了 DISTINCT，则将所生成结果中的所有内容重复的行剔除掉。
- 若有 ORDER BY 子句，则按相应的排序设置对结果进行排序。

11.2 Visual Basic 6.0 可视化数据管理器

可视化数据管理器（Visual Data Manager）是 Visual Basic 6.0 提供的数据库管理工具。它为 Visual Basic 6.0 提供了一种创建和修改数据库的交互方式，并可以安全地进行管理和实验 SQL 语句。

11.2.1 【可视数据管理器】窗口

【可视化数据管理器】可以通过选择【外接程序】/【可视化数据管理器】菜单命令，打开【可视化数据管理器】窗口，如图 11-1 所示。

图11-1 【可视化数据管理器】窗口

如图 11-1 所示，【可视化数据管理器】窗口主要由标题栏、菜单栏和工具栏 3 部分组成。

【可视化数据管理器】的菜单栏如图 11-2 所示，它含有【文件】、【实用程序】、【窗口】、【帮助】4 个一级菜单。

图11-2 菜单栏

【可视化数据管理器】的工具栏如图 11-3 所示。

图11-3 工具栏

利用菜单栏和工具栏，可以实现以下主要功能：

- 创建数据库与表。
- 表的修改（添加、删除字段与索引）。
- 所有对象的属性浏览。
- 演示新的数据绑定列表与数据绑定组合框控件。
- 所有被支持的数据类型的导入与导出。
- 直接的 SQL 语句执行，支持 SQL 支持的所有函数。
- 查询生成器，可以帮助不熟悉 SQL 的用户使用各种查询语句。
- 窗体生成器，可以快速生成表的编辑窗体。
- 将表的结构以及数据复制到相同或者不同的数据库中。

- **关系/参照完整性地创建与修改。**

如果执行【文件】/【新建】/【Microsoft Access】/【Version 7.0】菜单命令，如图 11-4 所示，将会出现如图 11-5 所示的【选择要创建的 Microsoft Access 数据库】对话框。

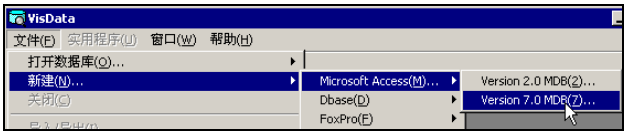


图11-4　新建一个 Microsoft Access 数据库

图11-5　【选择要创建的 Microsoft Access 数据库】对话框

选择要保存数据库的文件夹，并在【文件名】文本编辑框中输入数据库文件名（如“职工信息”），然后单击 保存(S) **命令按钮，将出现如图 11-6 所示的【数据库窗口】和【SQL 语句】窗口。**

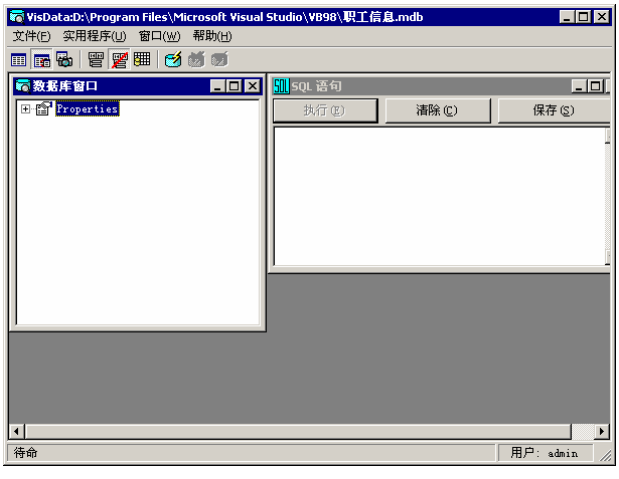


图11-6　【数据库窗口】和【SQL 语句】窗口

下面几节将讲述【数据库窗口】和【SQL 语句】窗口的一些经常使用的方法。

11.2.2　用【可视化数据管理器】创建数据库

在这里主要以创建“职工信息库”为例，讲解如何用【数据库窗口】创建一个新表。其具体步骤如下：

1. 如图 11-6 所示，用鼠标右键单击“ Properties ”，将出现如图 11-7 所示的弹出式菜单。
2. 在弹出式菜单中，选择【新建表】命令，将出现【表结构】对话框，如图 11-8 所示。

图11-7　弹出式菜单

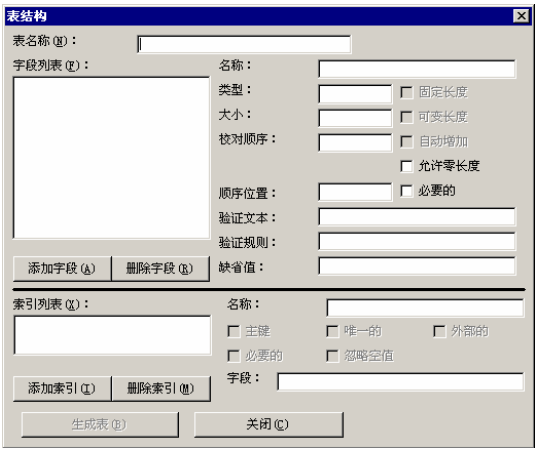


图11-8　【表结构】对话框

3. 在【表名称】文本编辑框中，输入“ 职工信息表 ”。
4. 在【表结构】对话框中，单击 添加字段(A) 命令按钮，将出现如图 11-9 所示的【添加字段】对话框。

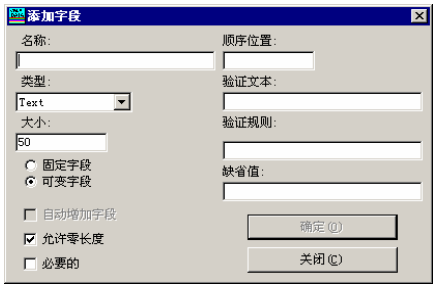


图11-9　【添加字段】对话框

5. 在【添加字段】对话框中，添加名为“ 职工编号 ”字段。在【名称】文本编辑框中输入“ 职工编号 ”，【类型】组合框中，选择“ Long ”，选中【自动增加字段】复选框。如图 11-10 所示。

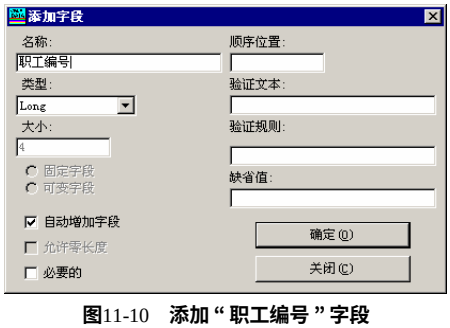


图11-10　添加“ 职工编号 ”字段

6. 单击 确定(O) 命令按钮，完成“ 职工编号 ”字段的添加。同时，在【表结构】对话框中，出现“ 职工编号 ”字段的相关信息，如图 11-11 所示。

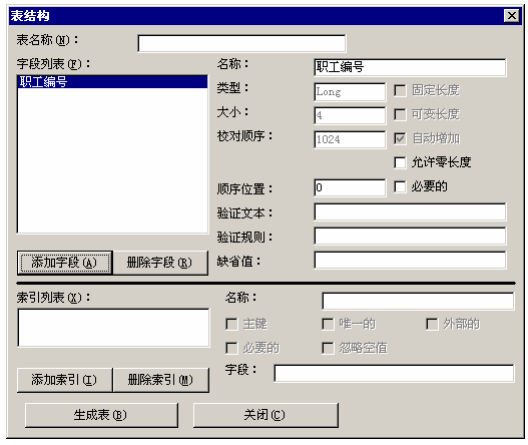


图11-11　添加“职工编号”字段后的【表结构】对话框

7. **按照 5、6 步的方法，为职工信息表添加其他字段，其属性如表 11-13 所示。**

表 11-13　　【职工信息表】的结构

字段名	字段类型	字段大小
职工编号	Long	**默认**
姓名	Text	20
性别	Text	2
年龄	Integer	3
政治面貌	Text	4
职务	Text	30
特长	Text	50

8. **添加完各字段后，单击** 关闭(C) **命令按钮，关闭【添加字段】对话框，回到【表结构】对话框，在其【字段列表】列表框中，列出了所有的字段名。如图 11-12 所示。**

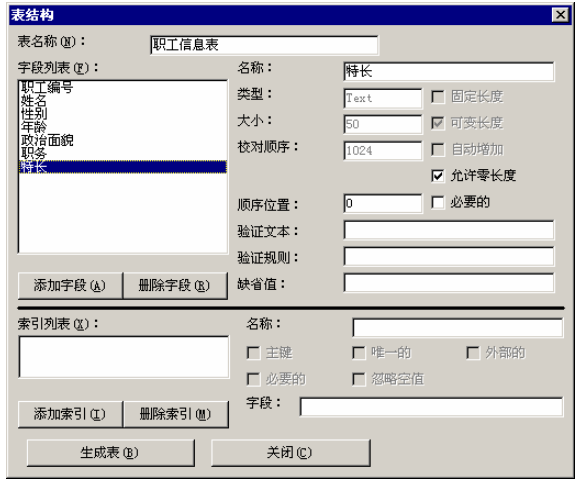


图11-12　添加完字段后的【表结构】对话框

9. **下面就要为此表添加索引。单击** 添加索引(I) **命令按钮，将出现如图 11-13 所示的**

【添加索引】对话框。

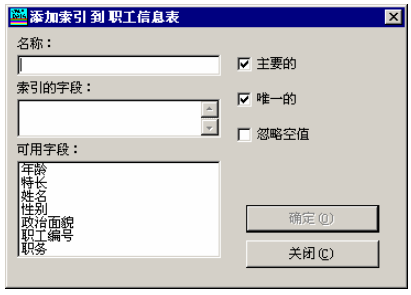


图11-13 【添加索引】对话框

10. 在【添加索引】对话框中，在【名称】文本编辑框中输入“编号”；选中【主要的】和【唯一的】两个复选框；在【可用字段】列表框中选择“职工编号”，此时在【索引的字段】列表框中将出现“职工编号”索引字段，同时 确定(O) 命令按钮变为可用状态，如图 11-14 所示。

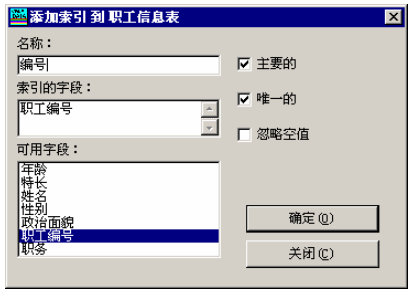


图11-14 添加索引后的【添加索引】对话框

11. 单击 确定(O) 命令按钮，添加索引。
12. 单击 关闭(C) 命令按钮，关闭【添加索引】对话框，回到【表结构】对话框，此时【表结构】对话框中的下部显示了建立索引的相关信息，如图 11-15 所示。

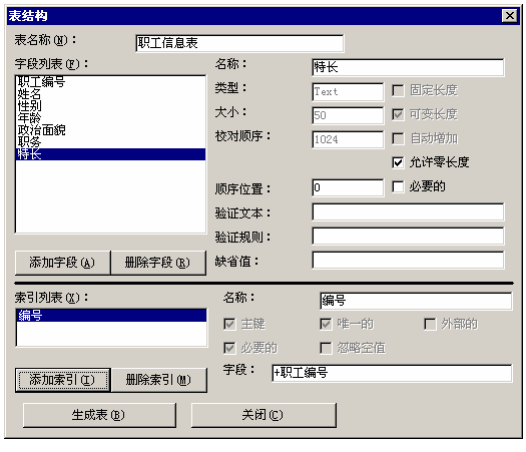


图11-15 建立索引后的【表结构】对话框

13. 单击 生成表(B) 命令按钮，“职工信息表”便形成了，在【数据库窗口】出现了“职工信息表”文件，如图 11-16 所示。
14. 此时建立了“职工信息表”，接下来要在表中增加记录。在【数据库窗口】中右击“职工信息表”，出现一个弹出式菜单，如图 11-17 所示。

图11-16　建立“职工信息表”后的【数据库窗口】

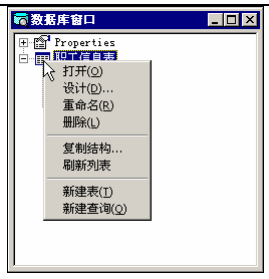


图11-17　弹出式菜单

15. **选择【打开】菜单命令，打开【表数据管理】对话框如图** 11-18 **所示。**

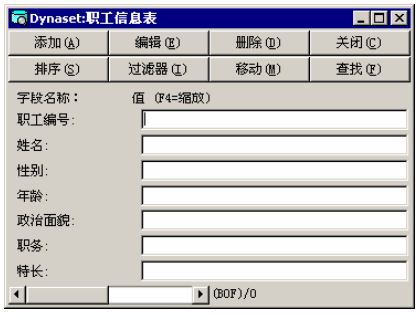


图11-18　【表数据管理】对话框

16. **单击** 添加(A) **命令按钮，显示【添加记录】对话框，如图** 11-19 **所示。**

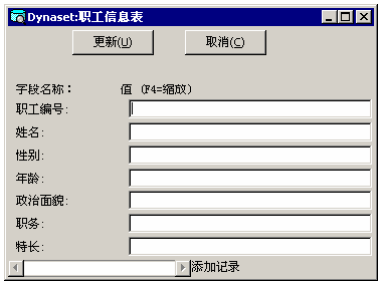


图11-19　【添加记录】对话框

17. **如图** 11-20 **所示，输入第一个职工的相关信息，然后单击** 更新(U) **命令按钮保存数据，完成第一个记录的添加。**

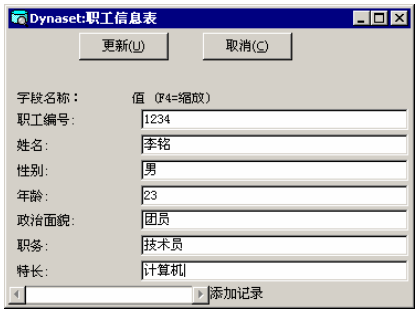


图11-20　添加第一个职工信息

18. **如果需要继续添加，重复第（**17**）步的工作即可。在表中按表** 11-14**，添加所有的记录。**

表 11-14　　职工信息表

职工号	姓名	性别	年龄	政治面貌	职务	特长
1234	李铭	男	23	团员	技术员	计算机
3255	胡兵	男	43	党员	处长	跳舞
3466	赵明	男	28	党员	科长	文艺、棋牌
1236	邓丽	女	20	团员	技术员	文艺、美术
4356	黎平	女	29	党员	会计	篮球
2351	孙灵	男	25	群众	科员	棋牌

到此为止，一个利用 Visual Basic 6.0 的【可视化数据库管理器】初步创建数据库就完成了。

11.2.3　【SQL 语句】窗口

在前面介绍了 SQL 语言，要在 Visual Basic 6.0 中使用，就要使用到【SQL 语句】窗口，如图 11-21 所示。

图11-21　【SQL 语句】窗口

要在某数据库中查询数据，可以在【SQL 语句】窗口中输入 SQL 语句，然后单击 执行(E) 命令按钮，即可显示查询结果。单击 保存(S) 命令按钮，即可将查询结果以数据库表的形式保存在数据库中。

例如，要在“ BIBLIO ”数据库中查询“ Publishers ”表中的 PubID<10 的“ PubID ”和“ Name ”两个字段。其具体步骤如下。

1. 执行【文件】/【打开数据库】/【Microsoft Access】菜单命令，打开如图 11-22 所示的【打开 Microsoft Access 数据库】对话框。

图11-22　【打开 Microsoft Access 数据库】对话框

2. 选择“ BIBLIO ”，打开“ BIBLIO ”数据库。

3. 在【SQL 语句】窗口中输入以下语句：

```
SELECT PubID,Name
FROM Publishers
Where PubID<10
```

4. 单击 执行(E) 命令按钮，将看到如图 11-23 所示的消息框。

图11-23　消息框

5. 因为“SQL 传递查询”只有在客户/服务器开发中使用，所以单击 否(N) 命令按钮即可。此时就会出现查询结果，如图 11-24 所示。

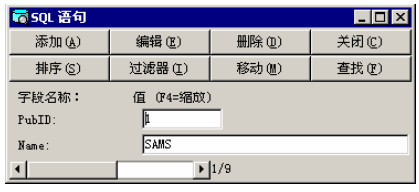


图11-24　查询结果

如果在单击 执行(E) 命令按钮之前，单击工具栏上的 Data 控件，使用此方法来显示查询结果如图 11-25 所示。

图11-25　使用 Data 控件的方法显示查询结果

如果在单击 执行(E) 命令按钮之前，单击工具栏上的 DBGrid 控件，使用此方法来显示查询结果如图 11-26 所示。

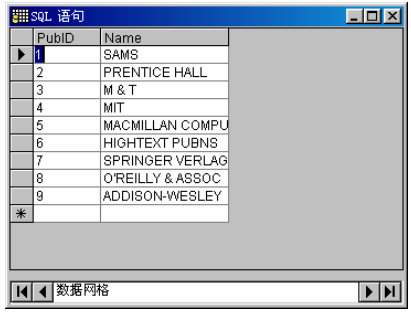


图11-26　使用 DBGrid 控件的方法显示查询结果

6. 关闭查询结果窗口，在【SQL 语句】窗口中，单击 保存(S) 命令按钮，将看到如图 11-27 所示的【保存查询】对话框。

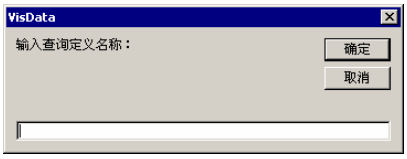


图11-27 【保存查询】对话框

为查询结果起一个名字，然后单击 确定 命令按钮，在出现如图 11-23 所示的消息框之后单击 否(N) 命令按钮，将看到数据库窗口里多了一个查询。

到此，这个查询就已经储存在数据库中了。在任何时候都可以像使用一个普通的表格一样来使用它。

另外，可以不直接在【SQL 语句】窗口中输入 SQL 语句，可以使用【查询生成器】来创建一个可以作为基本查询使用的 SQL 语句。

11.2.4 利用数据管理器管理数据库

前几节介绍了怎么来创建一个数据库，但是在数据库投入使用之后，其主要工作就是怎么来管理好一个数据库。同样可以使用可视化数据库管理器管理数据库。

11.2.4.1 【数据库窗口】管理数据库

当用 Visual Basic 6.0 打开某一数据库时，在【数据库窗口】中将出现各表的字段、索引及属性。在【数据库窗口】中选取字段和索引，即可查看表中字段和索引的结构。如图 11-28 所示。

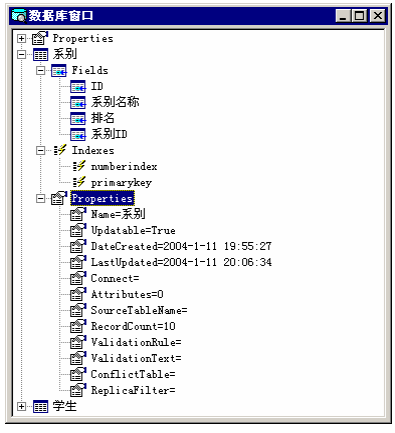


图11-28 查看数据库的相关信息

在这里，还可以增加、删除和重命名字段，也可以修改字段的某些属性。

在【数据库窗口】中用鼠标右键单击一个表名，会出现如图 11-17 所示的弹出式菜单。在弹出式的菜单中选择相关命令既可实现相关操作，例如：选择【重命名】命令，则表明变为可修改状态，输入新的名字，按下回车键即可；选择【删除】命令，将显示一个如图 11-29 所示的【确认】对话框，单击 是(Y) 命令按钮，即可将这个表从数据库中永久删除。

图11-29 【确认】对话框

11.2.4.2 【表结构】窗口管理数据库

表中的字段创建完毕后，可以在【表结构】窗口中设置或修改字段及索引的许多属性。在如图 11-17 所示的弹出式对话框中，选择【设计】命令即可打开【表结构】窗口。

要修改字段的属性，在【字段列表】列表框中选择字段名，在【字段列表】列表框的右边便显示了该字段的各属性，如图 11-12 所示。字段中可修改的属性以可修改文本框或检查框的形式显示，其他不可修改的属性则以不能修改的方式显示出来。

用同样的方法可修改索引的属性。

11.2.4.3 【表数据管理】窗口管理数据库

在【数据库窗口】中双击想要修改的表名，则会出现【表数据管理】窗口，如图 11-30 所示。

对话框上部有两排命令按钮，可以对表进行相关管理。在命令按钮下面的文本框中显示的是表中各字段在该记录中的相应内容。在最下面有一个滚动条，后面显示“ 1/10 ”，表示共有 10 个记录，当前显示的是第一个记录。

利用窗口中的 8 个命令按钮可以对字段进行如下操作。

- 添加：对于 添加(A) 命令按钮的功能，在前面一节已经介绍过，在此就不再重复讲解。
- 编辑：在【表数据管理】窗口的文本框中显示的内容均不可修改。若要修改记录的内容，推动滚动条，或者单击滚动条两边的前进▶或后退◀按钮，将要修改的记录定为当前记录。然后单击 编辑(E) 命令按钮，则出现如图 11-31 所示的【编辑记录】对话框。在此文本框中的字段内容均为可修改的，将字段内容改为所希望的内容，然后单击 更新(U) 命令按钮，就可完成编辑操作。

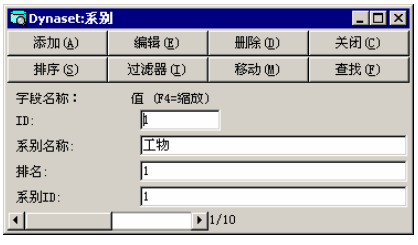


图11-30　【表数据管理】窗口

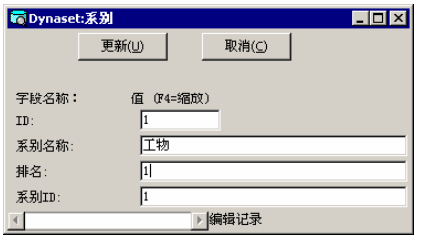


图11-31　【编辑记录】对话框

- 删除：在【表数据管理】窗口中，将所要删除的记录定为当前记录，单击 删除(D) 命令按钮，会出现如图 11-32 所示的【确认】对话框。单击 是(Y) 命令按钮即可删除该记录，【表数据管理】窗口中将自动显示下一条记录的内容。

图11-32　【确认】对话框

- 排序：当记录比较凌乱时，使用排序可以使当前表井然有序。单击 排序(S) 命令按钮，将出现如图 11-33 所示的【排序列】对话框，输入作为排序标准的字段名，然后再单击 确定 命令按钮，这时表已经按照要求的排序条件自动排好序。
- 过滤器：有时候表中的记录太多，但只需要操作其中很少一部分记录，这时就可以使用【过滤器】精简显示出来的记录集。单击 过滤器(I) 命令按钮，会出现如图 11-34 所示的【输入过滤器表达式】对话框。输入要求操作的记录所满足的条件表达

式，单击 确定 命令按钮，即可显示对应的记录。

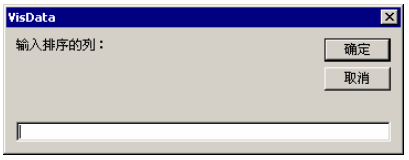


图11-33 【排序列】对话框

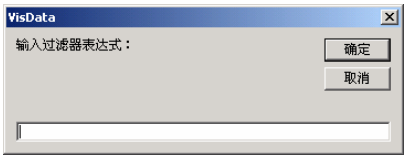


图11-34 【输入过滤器表达式】对话框

- 移动：如果经过排序不能达到，自己想要的记录序列，则可通过移动单个记录来调整。将要移动的记录跳到当前记录，单击 移动(M) 命令按钮，出现如图 11-35 所示的【输入移动的行数】对话框，输入所要移动的行数，单击 确定 命令按钮，即可将记录按指定的步数向前或向后移动。

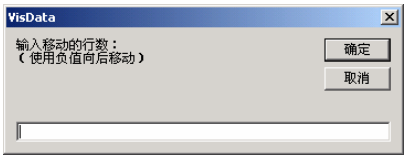


图11-35 【输入移动的行数】对话框

- 查找：当记录比较多时，要在记录中查看某个记录所在位置，可以使用查找功能。单击 查找(F) 命令按钮，将出现如图 11-36 所示的【查找记录】对话框。在【字段】列表框中选择一字段名，在【运算符】列表框中选择运算符，在【值或表达式】文本编辑框中输入适当的数值或表达式，在下面的一组单选按钮中选择一种查找方式。然后单击 确定(O) 命令按钮，即可找到符合条件的记录。如果该记录不是所要找的记录，可以选择【查找下一个】单选按钮继续查找，直到找到为止。

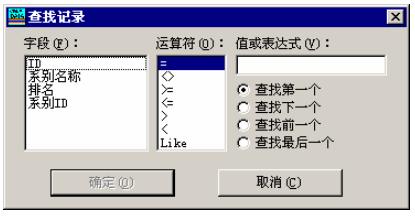


图11-36 【查找记录】对话框

11.3 使用控件访问数据库

Visual Basic 6.0 除了采用上述方式来操作数据库之外，还提供了几个可以访问数据库的控件，如数据控件（Data）、列表控件（DBList）、数据组合框控件（DBCombo）以及数据表格控件（DBGrid）。

11.3.1 数据控件（Data）

使用数据控件访问数据库，这是 Visual Basic 6.0 中连接数据库的最简单的方法。如果是简单的数据库应用，可以使用 Data 控件来执行大部分数据的访问操作，而根本不用编写代码，从而大大简化了数据库的编程。与 Data 控件相捆绑的控件自动显示来自当前记录的一个或多个字段的数据。此外也可以把 Data 控件和 Visual Basic 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，而且在运行中不可以更改。从工具栏中选择 Data 控件，将其放置在窗体上，其外观如图 11-37 所示。

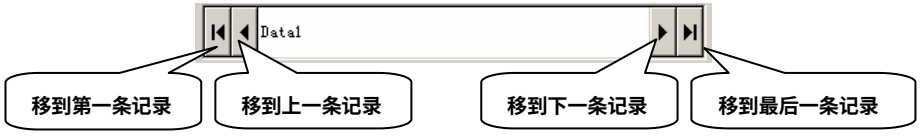


图11-37　Data 控件

11.3.1.1 Data 控件的主要属性

前面的章节已经介绍了一些基本控件的属性，Data 控件的很多属性与其他基本控件的一样，但是 Data 控件主要是用于数据库连接的，因此它还具有一些重要的特殊属性。

- Connect 属性：Connect 属性用来指定 Data 控件连接的数据库类型，Visual Basic 6.0 支持的数据库类型众多，如 MicrosoftAccess、Excel 及 Foxpro 等。其中默认的数据库为 Access。
- DatabaseName 属性：当 Data 控件连接 Access 时，该属性用于设置或返回 Data 控件所使用的数据库的名称。
- Record-Source 属性：Record-Source 属性用于指定 Data 控件所要操作的一个表或一个查询。一个数据库中可能有多个表和查询，在设置了 Record-Source 属性后，在 Record-Source 属性的下拉列表中会出现所选数据库中的所有表及查询，可以从中选择一个。
- RecordsetType 属性：该属性用来设置记录集的类型。它包含 3 种记录集类型，分别是 Table（表型集）、Dynaset（动态集）和 Snapshot（快照集）。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，是最灵活的一种记录集类型。Snapshot 类型的记录集只能静态显示数据（只读），其灵活性最低，但所需的系统资源最少。
- Exclusive 属性：该属性的功能是决定 Data 控件所连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。
- Recordset 属性：记录集，代表“Record-Source”所选中的表或查询。
- 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。
- BOFAction 属性：该属性用来决定当记录移动超出起点时，Data 控件要执行的操作。0 代表定位到第一个记录；1 代表移过记录集的开始位置，定位到一个无效位置，且触发 Data 控件 Validata 事件。
- EOFAction 属性：该属性用来决定当记录移动超出结束时，Data 控件要执行的操作。0 代表重定位到最后一个记录；1 代表移过记录集的结束位置，定位到一个无效位置，且触发 Data 控件 Validate 事件；2 代表代表移过记录集的结束位置，并自动增加一条新记录。

11.3.1.2 Data 控件的方法

Data 控件常用的方法有：Refresh 方法、UpdateRecord 方法、UpdateControls 方法及

Cancelupdate 方法。

- Refresh 方法

当运行时修改了“ Record-Source ”属性后，需要调用该方法刷新记录集。该方法的语法如下：

```
Data1.Refresh。
```

- UpdateRecord 方法

将被绑定在 Data 控件上的控件的数据写入数据库中，即当修改了数据后调用该方法确定修改。

- UpdateControls 方法

将被绑定在 Data 控件上的控件修改后的数据恢复为原始值。

- Cancelupdate 方法

将数据库中的数据重新读到被绑定在 Data 控件上的控件中。即当修改了数据后调用该方法放弃修改。

11.3.1.3 Data 控件的事件

Data 控件的事件主要包括 Validate 事件和 Reposition 事件。

- Validata 事件

当移动记录集记录指针时发生的事件。例如，将记录集记录指针从 A 移动到记录 B 产生 Validate 事件时，记录指针仍在记录 A 上。

```
Sub Object_Validate (Action As integer, Save As integer)
```

其中：

Object：是一个对象表达式，用于确定 Validate 事件是由哪一个对象产生的。

Action：一个整数值，指出如何产生了该事件，如移动、增加或查询等。

Save：一个布尔表达式，表示是否保存已修改的数据。当修改了绑定在数据控件上的数据，又没有调用 UpdateRecord 方法，则移动指针时，Save=True。如果在事件中令 Save=False，则放弃修改。

例如：

```
Sub Data1_Validate(Action As integer,Save As integer)
    Dim Msg
    If Save then
        Msg= MsgBox("Dada changed,Save?",vbYesNo)
        If Msg = vbNo then
            Save = False
        End if
    End if
End Sub
```

- Reposition 事件

当移动记录集指针到当前位置时发生的事件。使用 Reposition 事件可以在指针到达记录位置前进行基于当前记录的数据操作，例如，将记录集记录指针从 A 移动到记录 B，当记录指针移动到 B 上时，已产生 Reposition 事件。它可以改变其他对象的属性或使用其他对象的方法来设置窗体和界面。

Reposition 事件的语法如下：

```
Private Sub Object_Reposition()
```

11.3.2　数据绑定控件

Data 控件能够操作一个数据库文件，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，还需要使用其他控件来显示数据。这就需要将文本框等控件与 Data 控件连接起来，使之成为 Data 控件的数据绑定控件。Visual Basic 6.0 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、图片框等都可以作为数据绑定控件。另外 Visual Basic 6.0 还提供了专门的数据绑定控件，例如，DBList(数据列表框)、DBCombo(数据组合框)、DBGrid(数据网格)等。

可以说，Data 控件是 Visual Basic 6.0 和数据库之间的桥梁，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 Visual Basic 6.0 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：

- 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。

- 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，用户可以从中选择一个或多个。

下面将通过一个实例来介绍绑定控件的基本使用方法。

【例11-7】绑定控件的基本使用方法。

其具体操作步骤如下。

1. 新建一个 EXE 工程。此时在一般情况下，工具箱中并没有 DBList（数据列表框）、DBCombo（数据组合框）、DBGrid（数据网格）这些绑定控件。
2. 执行【工程】/【部件】菜单命令，打开如图 11-38 所示的【部件】对话框

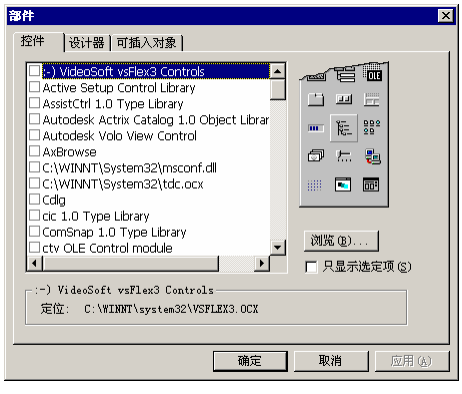


图11-38　【部件】对话框

3. 选择【控件】选项卡，找到“Microsoft Data Bound List Contol 6.0”(可以同时添加 DBList(数据列表框)、DBCombo(数据组合框)两个控件）和“Microsoft Data Bound Grid Contol 5.0”(用来添加 DBGrid(数据网格)控件）两项，选择左边的复选框，然后单击 确定 命令按钮，此时在工具箱内就出现了 DBList(数据列表

框)、DBCombo(数据组合框)、DBGrid(数据网格)3 个控件，如图 11-39 所示。

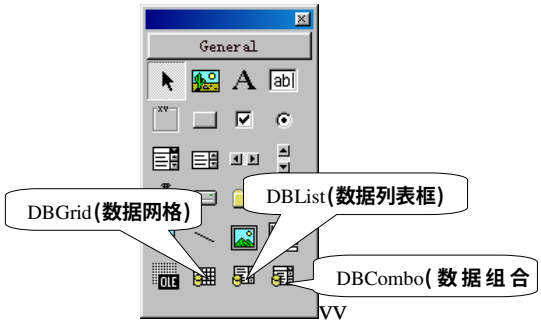


图11-39　添加控件后的工具箱

4. 在窗体上添加如图 11-40 所示的控件。

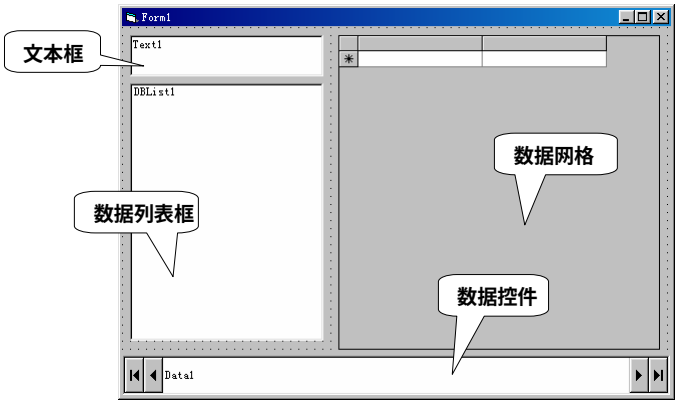


图11-40　控件布置图

5. 按照表 11-15 所示，为窗体设置属性。

表 11-15　　控件属性设置表

控件名称	控件属性	控件属性值	说明
Data1	DatabaseName	…/vb98/Biblio.mdb	把 Biblio 数据库中的 Publishers 表引入
	RecordSource	Publishers	
	Caption	Publisher	
Text1	text	空	
	DataSource	Data1	把 Text1 控件和 Data1 指定表中的 PubID 字段绑定
	DataField	PubID	
DBList1	DataSource	Data1	把 DBList1 控件和 Data1 指定表中的 PubID 字段绑定
	DataField	PubID	
	RowSource	Data1	设定 DBList1 中显示的是表中的 Company Name 字段
	ListField	Company Name	
DBGrid1	DataSource	Data1	

6. 运行程序，界面如图 11-41 所示。

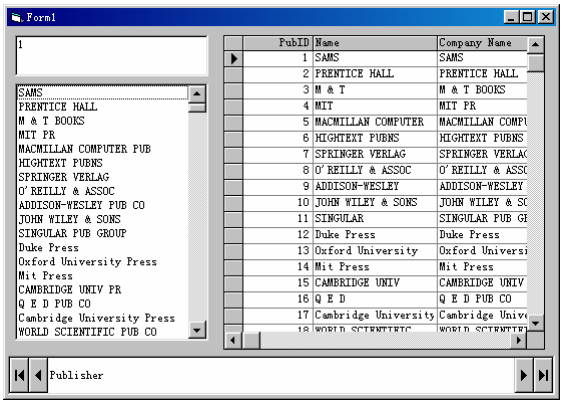


图11-41　程序运行界面

11.3.3　数据库记录的操作

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 Recordset 对象的属性和方法编写事件过程来实现。

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。

11.3.3.1 Recordset 对象的属性

在操作数据库时，经常要使用 Recordset 对象的一些属性来判断记录的位置。

- BOF 属性：当 BOF 属性的值为“True”时，表明当前位置位于第一个记录之前。
- EOF 属性：当记录集记录指针指向最后一条记录时返回“True”。
- AbsloutePosition 属性：返回当前记录集记录指针，第一条记录为 0，是只读属性。
- Bookmark 属性：String 类型，返回或设置当前记录集记录指针的书签，是可读写属性。每一条记录都有自己惟一的书签，它与记录在记录集中的顺序无关。将 Bookmark 属性存放到变量中，后面可以通过将该变量赋值给 Bookmark 属性，并返回到这个记录。
- NoMatch 属性：当使用 Find 方法查询时如果未找到则返回 True。NoMatch 属性常与 BookMark 属性同时使用。

11.3.3.2 Recordset 对象的方法

使用 Recordset 对象的方法可以实现记录指针的移动、记录的查找以及记录集的编辑等操作。

(1)　记录指针的移动方法

可以使用 Data 控件的箭头按钮来遍览记录，也可以使用 Data 控件的 Move 方法来实现记录指针的移动操作。下面列出了 Data 控件的 5 种 Move 方法：

- MoveFirst 方法：将记录集指针移动到第一条记录上。
- MoveLast 方法：将记录集指针移动到最后一条记录上。
- MovePrevious 方法：将记录集指针移动到前一条记录上。
- MoveNext 方法：将记录集指针移动到下一条记录上。
- Move(n)方法：将记录集指针向前或向后移动 *n* 个记录。

使用这些方法的格式相同，例如使用 MoveFirst 方法的格式为：

```
Data1.Recordset.MoveFirst
```

应该注意的是，如果记录已经处于 BOF 或 EOF 位置时，使用 MovePrevious 方法或 MoveNext 方法就会造成非法程序的错误。因此，我们在编写代码时要首先检查记录指针的位置，当记录已经达到 BOF 或 EOF 位置时，必须先进行处理。

若到达 BOF 位置时，执行 MovePrevious 方法时，将指针移动到第一条记录上，例如：

```
Private Sub Command1_Click()
    Data1.Recordset.MovePrevious
    If Data1.Recordset.BOF Then
        Data1.Recordset. MoveLast
    End If
End Sub
```

若到达 EOF 位置时，执行 MoveNext 方法时，将指针移动到第一条记录上，例如：

```
Private Sub Command2_Click()
    Data1.Recordset.MoveNext
    If Data1.Recordset.EOF Then
        Data1.Recordset. MoveFirst
    End If
End Sub
```

(2) 记录的查找

使用记录的一些查找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。下面列出了了 Data 控件的 4 个查找方法。

- FindFirst 方法：在记录集中查询符合条件的第一条记录。
- FindLast 方法：在记录集中查询符合条件的最后一条记录。
- FindPrevious 方法：在记录集中查询符合条件的前一条记录。
- FindNext：在记录集中查询符合条件的下一条记录。

这 4 种查找方法的语法格式与移动方法的格式相同，例如：

查找记录集中姓名为吴慧的第一条记录

```
Data1.Recordset.FindFirst "姓名='吴慧'"
```

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'"
```

如果条件的部分常数来自于变量，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.FindFirst"姓名：" & "，" & text1.Text & "，"
```

在调用查找方法时，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 Data1.Recordset.NoMatch Then
    MsgBox "记录不存在",64,"提示"
```

```
End lf
```

(3) 编辑记录集

还可以使用一些编辑方法对记录进行编辑，例如：

- AddNew 方法：将当前记录指向缓冲区，从而可以向记录集增加一条新记录。添加记录后将焦点移到第一个字段，用户可以马上开始新记录的输入。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。
- Update 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。
- Delete 方法：从记录集中将当前记录删除（在删除后常使用 MoveNext 方法移动指针）。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。

【例11-8】使用 AddNew 方法，添一个【增加】命令按钮。

```
Private Sub Command1_Click()
    Data1.Recordset.AddNew
    Data1.Recordset.Update
End Sub
```

因为只有在进行了 AddNew 方法后才可以输入数据，所以一般要在窗口的初始化时就增加一条新记录。

```
Private Sub Form_Initialize()
    Data1.Recordset.AddNew
End Sub
```

但是每当调用 AddNew 方法时，它就将输入的记录存入数据库中，而当关闭窗口时，刚输入的记录并没有保存到数据库中，那么在关闭窗口之前对 DATA 控件进行一次刷新就可以将数据存入数据库中了。

```
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    Data1.Refresh
End Sub
```

【例11-9】使用 Delete 方法，添加一个【删除】命令按钮。

```
Private Sub Command2_Click()
    Data1.Recordset.Delete
End Sub
```

11.4　数据库应用程序设计

前面已经介绍了如何创建数据库应用程序的各部分，下面将通过创建一个应用程序实例来进一步了解 Visual Basic 6.0 数据库程序设计。

在本例中将数据控件设置为不可见，仅使用 Data 控件连接数据库。本例通过按钮和程序代码来浏览和查找记录，运行时的界面如图 11-42 所示。

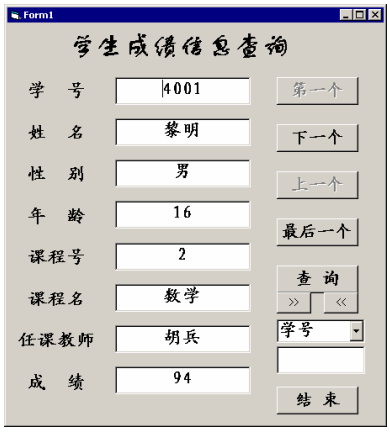


图11-42 程序运行界面

程序实现的主要功能为浏览并查询如表 11-5 所示的学生成绩信息表中的各记录。

11.4.1 建立数据库模型

在 11.4 中介绍过，在表 11-5 中有很多冗余数据，因此可以将此表分解成 11-6 学生信息表、11-7 课程信息表以及 11-8 学生成绩表 3 个表，然后在连接成表 11-5 形式。这里，可以使用 SQL 窗口创建学生成绩信息表 11-5。其具体操作步骤如下。

1. 利用 Visual Basic 6.0 可视化数据库管理器，按照 11.2.1 节及 11.2.2 节创建“职工信息表”的方法，在“学生信息数据库”中创建学生信息表、课程信息表及学生成绩表 3 个表。
2. 在【SQL 语句】窗口中输入以下语句：

```
select 课程信息表.课程号,课程名,任课教师
    ,学生信息表.学号,姓名,性别,年龄,成绩
from 课程信息表,学生信息表,学生成绩表
where 课程信息表.课程号=学生成绩表.课程号 and
    学生信息表.学号=学生成绩表.学号
```

3. 然后参照 11.2.3 节的方法，创建一个查询表，表命名为“总查询表”。其结构如表 11-5 所示。

这样就建立了一个数据库模型，此时在 Visual Basic 6.0【可视化数据库管理器】中，显示结果如图 11-43。

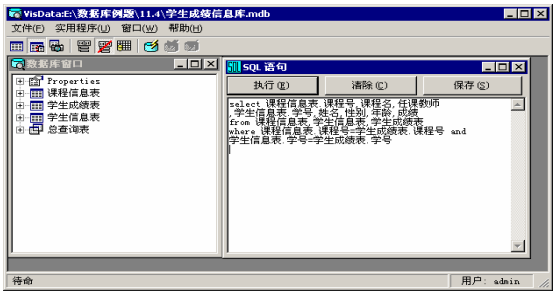


图11-43 创建查询后的【可视化数据库管理器】窗口

11.4.2　设计数据库应用程序

在创建了一个数据库模型之后，接下来就要创建数据库应用程序，其具体操作步骤如下。

1. **按照图** 11-44 **所示，创建应用程序界面。**

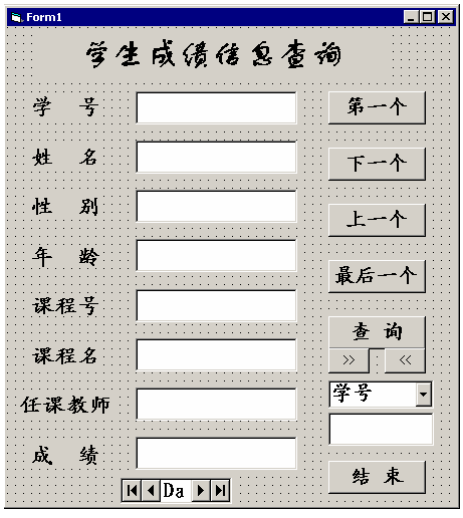


图11-44　**应用程序界面**

2. **依照表** 11-16（**只包含在界面上不能显示的部分属性**），**设置个控件的其他属性。**

表 11-16　　**控件部分属性设置**

控件名称	控件属性	控件属性值	说明
Data1	DataSource	学生成绩信息库	把学生信息数据库中的“总查询”表引入
	RecordSource	总查询表	
	Vasible	Flase	使 Data1 控件不可见
Text1	DataSource	Data1	把 Text1 控件和 Data1 指定表中的“学号”字段绑定
	DataField	学号	
Text2	DataSource	Data1	把 Text2 控件和 Data1 指定表中的“学号”字段绑定
	DataField	姓名	
Text3	DataSource	Data1	把 Text3 控件和 Data1 指定表中的“学号”字段绑定
	DataField	性别	
Text4	DataSource	Data1	把 Text4 控件和 Data1 指定表中的“学号”字段绑定
	DataField	年龄	
Text5	DataSource	Data1	把 Text5 控件和 Data1 指定表中的“学号”字段绑定
	DataField	课程号	
Text6	DataSource	Data1	把 Text6 控件和 Data1 指定表中的“学号”字段绑定
	DataField	课程名	
Text7	DataSource	Data1	把 Text7 控件和 Data1 指定表中的“学号”字段绑定
	DataField	任课教师	

续表

控件名称	控件属性	控件属性值	说明
Text8	DataSource	Data1	把 Text8 控件和 Data1 指定表中的“学号”字段绑定
	DataField	成绩	
ComboBox1	Text	学号	
	List	学号 姓名 课程号 课程名	

3. **在设计完界面后，就要为应用程序编写代码，其代码如下所示：**

```
Dim msg
Private Sub Command1_Click() '“第一个”按钮代码
    Data1.Recordset.MoveFirst   '将指针移到第一个记录上
    Command1.Enabled = False    '将“第一个”按钮设为不可用
    Command3.Enabled = False    '将“上一个”按钮设为不可用
    If Command2.Enabled = False Then Command2.Enabled = True
                                '将“下一个”按钮设为不可用
    If Command4.Enabled = False Then Command4.Enabled = True
                                '将“最后一个”按钮设为不可用
End Sub

Private Sub Command2_Click() '“下一个”按钮代码
    Data1.Recordset.MoveNext    '将指针移到下一个记录上
    If Data1.Recordset.EOF Then     '判断指针是否达到最后一个记录
        Data1.Recordset.MoveLast
        MsgBopx "已经达到最后一个记录！", 65, "提示"
        Command2.Enabled = False
        Command4.Enabled = False
    Else
        If Command1.Enabled = False Then Command1.Enabled = True
        If Command3.Enabled = False Then Command3.Enabled = True
    End If
End Sub

Private Sub Command3_Click() '“上一个”按钮代码
    Data1.Recordset.MovePrevious    '将指针移到上一个记录上
    If Data1.Recordset.BOF Then     '判断指针是否达到第一个记录
        Data1.Recordset.MoveFirst
        MsgBox "已经达到第一个记录！", 65, "提示"
```

```
            Command1.Enabled = False
            Command3.Enabled = False
        Else
            If Command2.Enabled = False Then Command2.Enabled = True
            If Command4.Enabled = False Then Command4.Enabled = True
        End If
    End Sub

    Private Sub Command4_Click() '“最后一个”按钮代码
        Data1.Recordset.MoveLast    '将指针移到最后一个记录上
        Command2.Enabled = False
        Command4.Enabled = False
        If Command1.Enabled = False Then Command1.Enabled = True
        If Command3.Enabled = False Then Command3.Enabled = True
    End Sub

    Private Sub Command5_Click() '“查询”按钮代码
        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
            msg = "姓名=" & "'" & TextFind.Text & "'"
            Data1.Recordset.FindFirst "姓名=" & "'" & TextFind.Text & "'"
        ElseIf CobFind.Text = "学号" Then
            msg = "学号 Like" & "'" & TextFind.Text & "'"
            Data1.Recordset.FindFirst "学号 Like" & "'" _
            & TextFind.Text & "'"
        ElseIf CobFind.Text = "课程名" Then
            msg = "课程名=" & "'" & TextFind.Text & "'"
            Data1.Recordset.FindFirst "课程名=" & "'" & TextFind.Text & "'"
        ElseIf CobFind.Text = "课程号" Then
            msg = "课程号 Like" & "'" & TextFind.Text & "'"
            Data1.Recordset.FindFirst "课程号 Like" & "'" _
            & TextFind.Text & "'"
        End If
        If Data1.Recordset.NoMatch Then
            MsgBox "记录不存在！", 64, "提示"
        End If
    End Sub
```

```
Private Sub Command6_Click() '“>>”查找符合条件的下一个记录
    Data1.Recordset.FindNext msg
End Sub

Private Sub Command7_Click() '“<<”查找符合条件的上一个记录
    Data1.Recordset.FindPrevious msg
End Sub

Private Sub Command8_Click() '“关闭”按钮代码
    End
End Sub

Private Sub Form_Load()
    Command1.Enabled = False
    Command3.Enabled = False
End Sub
```

到此，数据库应用程序就设计完毕，我们可以保存和运行应用程序了。

11.5 小结

通过本章的学习，我们对数据库的基本知识有了一定的了解，能够应用 Visual Basic 6.0 **【可视化数据库管理器】创建简单的数据库，能够应用** Data **控件及其他一些数据绑定控件，创建简单的数据库应用程序。**

在本章我们主要学习了以下内容：

- **数据库的基本组成。**
- **数据库的设计原则及步骤。**
- **数据库设计标准语言** SQL**。**
- Visual Basic 6.0 **可视化数据管理器的使用方法。**
- **数据控件及数据绑定控件**
- **数据库应用程序的设计方法。**

11.6 习题

一、 选择题

1. **关系数据库中的数据集合用表来表示，表示它的基本组成单元。一个数据库（ ）个表组成。**

   A. 1 **个** B. 2 **个** C. **多个** D. **一个或多个**

2. **表的每一行就是一个（ ）；表中的每一列称作一个（ ），描述了它所含有的数据。**

   A. **录 字段** B. **索引 字段**

   C. **段 记录** D. **记录 索引**

3. **一个好的数据库设计方案应不包括（ ）。**

A. 能用最少的时间定位特定记录。
B. 以最有效的方式存放数据，以节省存储空间。
C. 能使数据更新以尽量复杂的方式进行。
D. 在包含程序所需的新功能时应由足够的灵活性。

4. 数据组织的基本原则就是要使（　）。
A. 数据库更复杂
B. 数据库更简单
C. 数据库易于维护
D. 数据库更全面化

5. 以下（　）不是 SQL 语言的主要特点。
A. SQL 是一种交互式查询语言
B. SQL 不能嵌套在其他语言中
C. SQL 是一种数据库管理语言
D. SQL 是一种分布式数据库语言

6. Data 控件的很多属性，（　）属性用于指定 Data 控件所要操作的一个表或一个查询。
A. DatabaseName 属性
B. RecordsetType 属性
C. Exclusive 属性
D. RecordSource 属性

7. Data 控件的很多方法，当运行时修改了 Record-Source 属性后，需要调用（　）刷新记录集。
A. Refresh 方法
B. UpdateRecord 方法
C. UpdateControls 方法
D. Cancelupdate 方法

8. 在操作数据库时，经常要使用 Recordset 对象一些属性来判断记录的位置，EOF 属性的作用是（　）。
A. 表明当前位置位于第一个记录之前。
B. 当记录集记录指针指向最后一条记录时返回 True。
C. 返回当前记录集记录指针
D. 返回或设置当前记录集记录指针的书签

9. 使用记录的一些查找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；（　）在记录集中查询符合条件的前一条记录。
A. FindFirst 方法
B. FindLast 方法
C. FindPrevious 方法
D. FindNext 方法

二、 填空题

1. 一个数据库由一个或多个表组成，表的每一行就是一个_____，表中的每一列称作一个_____。

2. 每个表都应该有一个_____，它是记录的惟一标识符。
3. 子表（Child Table）是一个所有条目共享存放在另一个表中的公用信息表。而存放公用信息的表，则称为_____。
4. 对数据库进行规范化处理后，有时会产生把数据从一个表移到另一个表的需要，这可以通过在表间建立_____来实现。
5. 关键数据分为_____和_____。其中_____是对数据表内一个记录进行惟一表示的信息，而_____是把一个记录与另外某个数据表中的关键字联系起来的信息。
6. 【可视化数据管理器】窗口主要有_____、_____和_____3 部分组成。
7. Data 控件的很多属性，_____属性用来指定 Data 控件连接的数据库类型。
8. _____控件是 Visual Basic 6.0 和数据库之间的桥梁，而_____控件则把 Data 控件和用户界面联系起来，两者构成了 Visual Basic 6.0 开发数据库的主体。
9. Data 控件的_____方法，将记录集指针移动到最后一条记录上。
10. 在调用_____方法添加新记录后，必须调用_____方法来保存新添加的记录，否则所添加的记录无效。

三、 编程题

1. 预先定义一个小型销售公司关系数据库的内容，该数据库所包含的关系表有：

   • OFFICE 表。该表的列有 OFFICE、CITY、REGION，MGR、TARGET 和 SALES；分别表示销售处编号，销售处所在的城市名，销售处所在的地区名，销售处经理编号，销售处的目标和销售处的销售额。

   • SAlESREPS 表：该表有 EMPL_NUM、NAME、AGE、REP_OFFICE、TITLE、HIRE_DATE、MANAGER、QUOTA 和 SALES，分别表示销售人员编号、销售人员姓名、销售人员年龄、销售人员所在的销售处编号、销售处名称、雇用日期、销售人员负责人编号、销售人员定额和销售人员的销售额。

   • ORDERS 表：该表的列有 ORDER_NUM、ORDER_DATE、CUST、REP、MFR、PRODUCT、QTY 和 AMOUT，分别表示订单编号、订单日期、顾客编号、销售处编号、制造商编号、产品型号、产品数量和订单金额。

   根据定义的一个小型销售公司关系数据库的内容，用 SQL 语句完成以下几题的查询。

   (1) 列出销售额超过 6000 元的销售人员的姓名，销售目标和超过销售目标的销售额。
   (2) 查出 1999 年最后一个季度的订单情况。
   (3) 查出名称是以 ABC 开头的产品订单情况。
   (4) 列出所有的销售处,按区域名字母顺序排列。
   (5) 列出每个销售人员以及他们工作的城市和区域的情况。
   (6) 查出有多少销售员的销售额超过了其目标额，以及他们的销售额总和。

2. 使用 Visual Basic 6.0 可视化数据库管理器创建一个数据库，其中包括“学号”、“姓名”、“班级”和“成绩”4 个字段。编写一个应用程序，使用数据库控件，建立与存放学生成绩的数据库连接，程序执行后输出班级号为 004 的班级的所有学生成绩一览表。

第12章　Visual Basic 6.0 程序调试与维护

使用任何一种计算机语言，出错是在所难免的，即使是编程高手，也同样会出错，而且程序越大，代码越复杂，错误越是容易出现。有些错误是可以避免的，而有些错误是不可避免。另外有些错误对程序的运行，影响不大，而有些错误对程序的运行是致命的。因此，有效的调试手段和完善的错误处理，对于每个编程人员来说，都是必须的。学会这个技能，将是本章所要讲的主要内容。

本章学习目标

- Visual Basic 6.0 的 3 种工作模式。
- 错误的分类。
- 编译错误处理。
- 实时错误处理。
- 逻辑错误处理。

12.1　Visual Basic 6.0 的 3 种工作模式

编写程序时，出现了错误是一件很正常的事，但是出现了错误，必须找出错误出现的原因，以便排除错误。为了及时发现错误，有必要先知道程序是在何种模式下工作。Visual Basic 6.0 为用户提供了设计、运行、中断 3 种工作模式，以方便用户进行程序的维护和发现错误。在 Visual Basic 6.0 主界面的工具栏中，有 3 个快捷按钮允许用户在这 3 种模式之间切换，如图 12-1 所示。

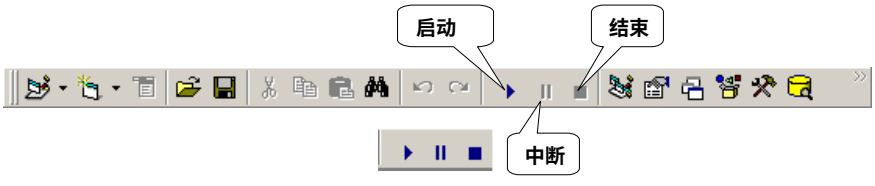


图12-1　工具栏上的快捷按钮

程序处于设计模式下时，可以进行设计工作，完成窗体的设计和程序代码的编写。在设计阶段，只有启动按钮可用，其余两个按钮为灰色，表示不可用，如图 12-2 所示。在窗体和代码设计完毕后，单击按钮，程序便进入运行模式下。在运行模式下，只能查看程序运行的结果以及程序代码，但不能修改程序代码。在程序运行阶段，中断按钮和结束按钮可用，而启动按钮不可用，如图 12-3 所示。如果单击结束按钮，则应用程序回到设计模式下；如果单击中断按钮，则程序进入中断模式下。在中断模式下，应用程序暂时的被停止，可以在程序暂停的时间里调试程序，并修改程序。在程序中断阶段，运启按钮和结束按钮可用，而中断按钮不可用，如图 12-4 所示。单击启动按钮，则程序继续运行；单击结束按钮，则结束程序。

图12-2　设计阶段的工具栏按钮

图12-3　运行阶段的工具栏按钮

图12-4　中断阶段的工具栏按钮

12.2 错误的分类

编写程序时，错误可以说是千差万别，各不相同。有些错误是由于执行了非法的操作所造成的，而有些错误是由于逻辑上的错误所造成的；有些错误是很容易被发现，而有些错误很隐秘。在 Visual Basic 6.0 中，错误被分为编译错误、实时错误、逻辑错误 3 大类。

编译错误是主要是由于没有按语法要求来编写代码所造成，例如：将变量或关键字写错了，漏写一些标点符号，或者是少写了配对语句等都会造成这类错误产生。这类错误一般出现在程序的设计或编译阶段，并且很容易被监测到。例如，在某个事件中，添加了如下语句：

```
InputBox("请输入数据","数据输入")
```

然后按回车键换行，这时便会弹出如图 12-5 所示的提示框，提示用户出现编译错误。

再如，在程序中如果使用了一个未被定义过的变量或者是在使用 if 语句时少写了配对语句 End if，也会产生编译错误，但这种编译错误在程序的编写阶段并不会被监测到，只有在程序的编译阶段才会被监测到。如果运行程序，便会立即弹出如图 12-6 或 12-7 所示的【编译错误提示】对话框。

图12-5 【语法错误提示】对话框

图12-6 【编译错误提示】对话框

图12-7 【语法错误提示】对话框

实时错误一般在运行的过程中才会出现，主要是由于执行了不能执行的操作而引起的。例如，在进行除法运算时，除数变为了 0；在写文件时，磁盘已满等都会引起实时错误。在编写文件处理程序时，实时错误最容易出现。例如，在某应用程序中，用以下代码来打开某个文件：

```
Private Sub Command1_Click()
    CommonDialog1.ShowOpen
    Open CommonDialog1.FileName For Input As #1
End Sub
```

在程序运行时，单击按钮弹出【打开】对话框，这时如果单击【打开】对话框的 取消 按钮，便会出现实时错误，弹出如图 12-8 所示的【实时错误】提示对话框。对于实时错误，用户可以通过编写相应的错误处理程序便可以排除。

图12-8 【运行错误提示】对话框

逻辑错误是最难被发现的错误。如果一个应用程序本身没有编译错误，并且在运行过程中也没有出现实时错误，但运行后所得到的结果就是不正确，通常这种情况都是由于逻辑错误所

造成的。这类错误的排除最为麻烦，需要积累一定的经验，并还要对运行的结果进行分析才能够被发现。

12.3　编译错误处理

编译错误一般是由于不正确的语法结构所引起的，这类错误是可以避免的。Visual Basic 6.0 为用户提供的自动语法检查功能，可以很容易地捕捉到这类错误。单击【工具】/【选项】菜单打开【选项】对话框，如图 12-9 所示，单击【编辑器】按钮，让【选项】对话框切换到【编辑器】设置窗口，如图 12-9 所示。在这个窗口中，可以看到【自动语法检测】栏默认被选中，因此一旦在【代码】窗口遇到语法错误，Visual Basic 6.0 便会自动弹出【编译错误提示】对话框，并且会以醒目的样式显示错误的代码行。如果错误出现在设计阶段时，便会以红色字体显示代码行，并弹出相应的【错误提示】对话框，如图 12-10 所示。如果错误出现在编译阶段时，便会以蓝色光条显示错误所出现的位置，并弹出相应的【错误提示】对话框，如图 12-11 所示。

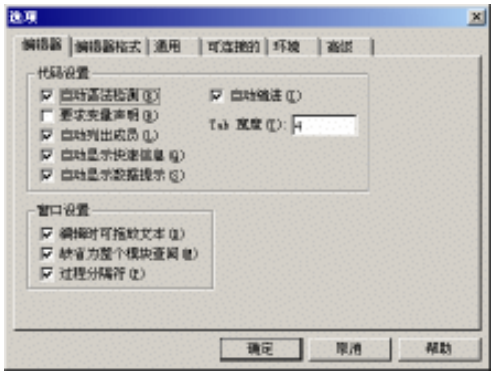


图12-9　【选项】对话框

为了能够及时发现编译错误，在编些程序时，要求变量必须被声明之后才能够被使用，这样就能够有效的发现编译错误。为了强调变量必须被声明，可以在程序的开始部分添加如下语句：

```
Option Explicit
```

在图 12-9 所示的【选项】对话框中，如果将【要求变量声明】选中，则 Visual Basic 6.0 会自动的在程序的开始添加 Option Explicit 语句。

编译错误被监测到后，只需要按语法要求去修改相应的错误代码，编译错误便会被排除掉，因此编译错误是 3 种错误中最容易被排除的错误。

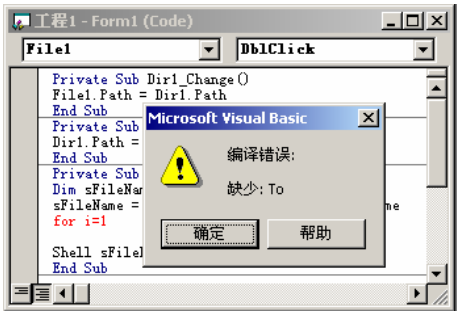


图12-10　错误提示

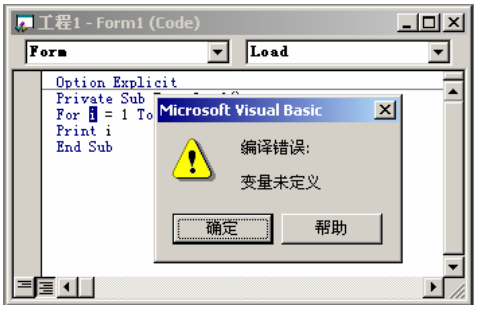


图12-11　错误提示

12.4 实时错误处理

实时错误是出现在程序运行的过程中，如果程序不能处理这类错误，就会使程序被意外的终止，甚至会导致死机。因此在设计程序之前，必须将可能出现的错误都考虑进去，然后编写相应的错误处理程序来捕获实时错误并执行相应的错误处理操作。

在【例 10-11】中，使用了一个文件的打开的程序，程序代码如下：

```
Private Sub mnuFileOpen_Click()
    Dim fName As String
    Dim text As String
    Dim textbuff As String
    '设置文件过滤器
    CommonDialog1.Filter = "文本文件(*.txt)|*.txt"
    '显示“打开”对话框
    CommonDialog1.ShowOpen
    fName = CommonDialog1.FileName
    If fName <> "" Then
        '打开顺序文件
        Open fName For Input As #1
        '读取顺序文件中的内容，并将它显示到文本框中
            Do While Not EOF(1)
                Line Input #1, text
                textbuff = textbuff + text
                Text1.text = textbuff
            Loop
        Close #1
    End If
End Sub
```

该程序能够被顺利执行的前提是在打开文件时，不出现错误，但实际上在打开文件时，也会产生一些错误，比如说，如果要打开软盘中的文件，但软驱中没有磁盘时，便会出现打开文件出错，这时程序便会被终止，并弹出如图 12-12 所示的【实时错误提示】对话框。为了让程序能够顺利的执行，必须在代码中加入实时错误处理程序。

图12-12 【实时错误提示】对话框

【例12-1】 为【例 10-11】添加实时错误处理程序。

1. 打开【例 10-11】所建的过程。

2. **打击【文件】/【打开】菜单，打开【代码】窗口，并在【打开】菜单的 Click 事件中新增如下带底纹的代码：**

```
Private Sub mnuFileOpen_Click()
    Dim fName As String
    Dim text As String
    Dim textbuff As String
    '捕获实时错误，并进入实时错误处理程序
    On Error GoTo errorprocedure
    '设置文件过滤器
    CommonDialog1.Filter = "文本文件(*.txt)|*.txt"
    '显示"打开"对话框
    CommonDialog1.ShowOpen
    fName = CommonDialog1.FileName
    If fName <> "" Then
        '打开顺序文件
        Open fName For Input As #1
        '读取顺序文件中的内容，并将它显示到文本框中
            Do While Not EOF(1)
                Line Input #1, text
                textbuff = textbuff + text
                Text1.text = textbuff
            Loop
        '关闭文件
        Close #1
    End If
    '实时错误处理程序
    errorprocedure:
    Dim retvalue As Integer
    '根据错误的种类，进行不同的操作
    If Err.Number = 53 Then
       retvalue = MsgBox(fName + "文件不存在", _
       vbOKCancel + vbExclamation, "错误")
              If retvalue = vbOK Then
                  '退出错误处理程序，并重新执行产生错误的语句
                  Resume
              Else
                  '退出错误处理程序，并执行下一条语句
                  Resume Next
              End If
    Else
```

```
            MsgBox "未知错误发生!", vbCritical, "错误"
            Exit Sub
        End If
    End Sub
```

3. 运行程序，如果文件打开出错，便不再弹出图 12-12 所示的【实时错误提示】对话框，并且程序还会继续进行。

从【例 12-1】中，可以看出，实时错误的排除，一般要经历以下几个过程：

- 实时错误的捕获

实时错误的捕获是通过 On Error 语句来实现的，语法结构如下：

```
On Error GoTo  行号或行标号
```

如果 On Error 语句一旦捕获到了实时错误，便会立即跳转到错误处理程序。例如，在【例 12-1】中，实时错误捕获语句

```
On Error GoTo errorprocedure
```

其中“ errorprocedure ”是用来标示错误处理程序所在的位置。如果程序一旦出现打开文件错误，则会直接跳转到错误处理程序。

- 进入实时错误的处理程序

程序在运行的过程中，如果一旦产生实时错误，便会暂停剩下代码的执行，而直接进入错误处理程序。在错误处理程序，可根据错误的不同而执行不同的错误处理。例如，在【例 12-1】中，在错误处理程序中，使用 if 条件语句将实时错误分为了两类，并且所执行的处理过程也不一样。

- 错误处理完毕，退出错误处理程序

实时错误处理完毕之后，便要退出实时错误处理程序，并恢复程序的运行。退出实时错误处理程序是由 Resume 语句来完成的，其语法结构有以下 3 种形式：

Resume 0 或 Resume：结束实时错误处理程序，并从产生错误的语句开始恢复运行。

Rexume Next：结束实时错误处理程序，并从紧随产生错误的语句的下一个语句恢复运行。

Resume line：其中参数 line 是行标签或行号，是用来指定从第几行开始恢复运行，参数 lin 所指定的行必须和错误处理程序处于同一个过程中。

例如，在【例 12-1】中，如果产生错误的原因可以更正，使用 Resume 语句不断的执行产生错误的语句，直到错误被更正；如果产生错误的原因不可以更正，使用 Resume Next 语句跳过产生错误的语句，直接执行程序的下一个语句。

Resume 语句只能使用在错误处理程序之中，不能在其他任何地方使用。

实时错误虽然也可以被 Visual Basic 6.0 监测到，但用户在编写程序之前，还是要周密的考虑，并且还需要用户积累一定的编程经验。

12.5 逻辑错误处理

逻辑错误是 3 类错误中最为头痛的，并且 Visual Basic 6.0 不能监测到这种错误。这类错误需要用户不断的调试程序，然后分析调试的结果，才能发现错误产生的原因。在调试程序之

前，我们有必要先了解一下 Visual Basic 6.0 提供的【调试】工具栏和【调试】菜单。

12.5.1　【调试】工具栏和【调试】菜单

为了方便调试程序，Visual Basic 6.0 提供了强大的【调试】工具栏，如图 12-13 所示。使用这些调试工具，可以深入到应用程序的内部，详细的分析程序运行时的过程，这对于逻辑错误的发现是非常有用的。【调试】工具栏一般是不显示在工具栏中的，需要自己去打开它。在主界面上单击【视图】/【工具栏】/【调试】菜单，便可以将【调试】工具栏显示到主界面上。【调试】工具栏各个按钮的说明见表 12-1。

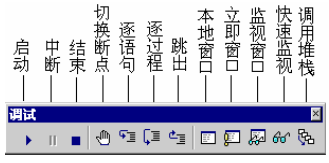


图12-13　【调试】工具栏

表 12-1　　【调试】工具栏按钮说明

按钮名称	说明
开始/继续	开始运行程序或继续运行程序
暂停	暂停应用程序的执行
结束	结束应用程序
切换断点	设置或清除断点
逐语句	逐行的执行程序
逐过程	逐过程执行程序
跳出	跳出某个过程
本地窗口	显示【本地】窗口
立即窗口	显示【立即】窗口
监视窗口	显示【调试】窗口
快速监视	显示【快速调试】窗口
调用堆栈	显示【调用堆栈】窗口

除了提供调试工具栏之外，Visual Basic 6.0 还对应的提供了调试菜单，在主界面上单击【调试】菜单，便可以看到所有的【调试】菜单，如图 12-14，并且每个菜单项与相应的【调试】工具栏按钮对应。

图12-14　【调试】菜单

12.5.2 使用断点

如果怀疑某条或某几条语句是产生逻辑错误的原因时，那么用设置断点的方法来发现逻辑错误，不失为一种可靠的方法。断点是程序代码中的一个标志位，程序运行到断点位置，便会停下来，并进入中断模式下。

(1) 断点的设置与清除

断点的设置和取消一般是在程序的设计阶段来完成的，断点的设置可以按以下步骤来完成：

- 在【代码】窗口将光标移到待设置断点的代码行。
- 在代码行的左侧的“边界指示区”上单击鼠标左键或直接按 F9 键，此时该代码行会以反白样式显示，并且在“边界指示区”上会出现一个实小圆点●，如图 12-15 所示。

断点设置完毕后，如果想清除某个断点，可按以下步骤进行：

- 将光标移动到断点所在的代码行。
- 在代码行的左侧的“边界指示区”上单击鼠标左键或直接按 F9 键，此时“边界指示区”上的实小圆点●就会消失，表示断点已被清除。

程序设置断点之后，程序运行到断点所在的代码行时，便会停下来，在该代码行的“边界指示区”显示一个黄色的小箭头，并将该代码行置为黄色，以表示程序暂停的位置，如图 12-16 所示。

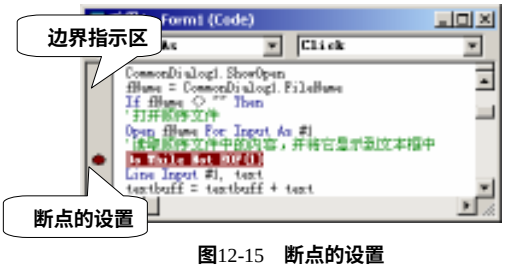


图12-15 断点的设置

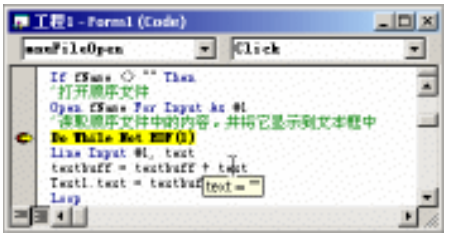


图12-16 程序进入中断状态

(2) 使用 Stop 语句设置断点

设置断点的另外的一种方法就是在代码中加入一条 Stop 语句，应用程序在运行的过程中，一旦遇到 Stop 语句，就会将 Stop 语句所在行看作是断点的位置，如图 12-17 所示，并暂停应用程序，进入中断状态。Stop 语句所设置的断点和直接设置的断点虽然实现的功能是一样，但它们之间还是存在着一定的差别。直接设置的断点，随着应用程序的关闭而消失，而 Stop 语句是作为代码的一行而加入程序中的，因此他不会随着程序的关闭而消失。

程序进入中断状态之后，用户可以在程序暂停的时间里观察某个变量或属性值或者是修改程序代码，如果一旦确认程序可以继续进行时，单击工具栏中的 按钮或直接按 F5 键，程序便会继续运行。

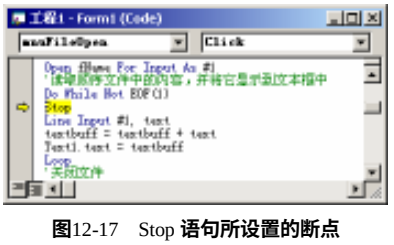


图12-17 Stop 语句所设置的断点

12.5.3　控制程序运行

当只是知道逻辑错误出现的大致范围时，如果还采用设置断点的方法来调试程序则比较麻烦，这时可以采用控制程序运行的方式来调试程序，让程序运行特定范围的语句或按特定的方式来运行程序。程序代码设计完毕后，直接单击程序按钮，则程序直接运行完毕，是看不到程序的运行过程的。如果想看到程序运行的过程，这时就必须让程序按特定方式运行。

- 逐语句的运行程序

逐语句运行程序就是让程序一次只运行一条语句，程序运行完一条语句之后，便进入中断状态，并将待运行的语句移到下一条语句，如图 12-18 所示，这样就可以此在程序执行完一条语句后，检查一下程序运行的情况，以便能够及时的发现错误。

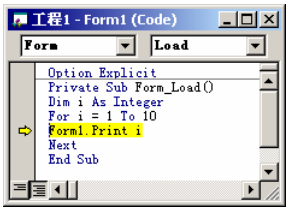


图12-18　逐语句运行程序

要想让程序逐语句的运行，则在准备运行程序或继续运行程序时，单击【调试】/【逐语句】菜单或单击【调试】工具栏上的按钮便可以让程序按逐语句的运行方式运行。

- 逐过程的运行程序

逐过程与逐语句类似，都是单步执行，即每次只执行一次操作。逐过程运行是将一个过程当作一次操作，程序每次只运行一个过程，然后便进入中断状态。如果怀疑逻辑错误出现在某个过程中时，用逐过程的方式运行程序是最佳的选择。

要想让程序逐过程的运行，则在准备运行程序或继续运行程序时，单击【调试】/【逐语句】菜单或单击【调试】工具栏上的按钮便可以让程序按逐过程的运行方式运行。

如果程序代码中没有过程的调用，则逐语句运行和逐过程运行两种运行方式是一样的；如果程序中存在着过程的调用，则两种运行方式是有差别的。

【例12-2】建立一个在窗体上显示 1～10 之间整数的程序，然后分别采用逐语句和逐过程运行程序，看看两种方式有什么区别。

1. 新建一个工程，并将窗体的“AutoRedraw”属性设为 True。
2. 双击窗体，为窗体添加 Load 事件，并在代码窗口添加如下代码：

```
Option Explicit
Private Sub Form_Load()
    Dim i As Integer
    Form1.AutoRedraw = True
    i = 10
    '调用显示数字的子过程
    ShowNum i
End Sub

Public Sub ShowNum(Num As Integer)
```

```
    Dim j As Integer
    For j = 1 To Num
        Form1.Print j
    Next
End Sub
```

3. 在代码窗口，将光标移到“ShowNum i”代码行，然后按 F9 键，在此代码行设置一个断点，如图 12-19。
4. 单击【视图】/【工具栏】/【调试】菜单，将【调试】工具栏显示到主界面上。
5. 单击 按钮运行程序，程序会在“ShowNum i”代码行暂停下来，如图 12-20 所示。

图12-19　设置断点

图12-20　程序暂停在断点处

6. 单击【调试】工具栏上的逐语句 按钮，则程序停在 Public Sub ShowNum（Num As Integer）代码行，如图 12-21 所示。
7. 单击 按钮停止程序，然后重新运行程序，同样程序会在“ShowNum i”代码行暂停下来。
8. 单击调试工具栏的逐过程 按钮，则程序停在“ShowNum i”的下一条语句“End Sub”上，如图 12-22 所示。

图12-21　逐语句运行程序

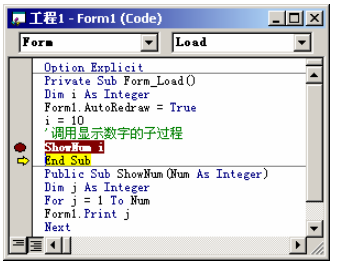


图12-22　逐过程运行程序

如果当前语句包含过程的调用，采用逐语句运行则进入被调用的过程里面，并将下一条运行语句设在所调用的过程中，而采用逐过程运行则直接将调用的过程运行完毕，并将主程序中的下一条设为下一条运行语句。例如，在【例 12-2】中，采用两种方式继续运行程序，两种方式运行的过程明显不同，如图 12-21、12-22 所示。

12.5.4　使用窗口

设置断点和控制程序运行这两种调试的方法，只能将出现逻辑错误的代码行找出来，但不能将出错的具体原因给找出来。如果分析逻辑错误产生的具体原因，例如，是某个变量引起的，还是某个属性引起的，这时就要用到与出错有关的 3 个窗口【本地窗口】、【立即窗口】、【监视窗口】，利用这 3 个窗口，可以实时地监测到各个变量是如何变化的，这样更有利于分析逻辑错误产生的具体原因，但这 3 个窗口在程序处于中断模式下时，才会发挥作用。

- 【本地窗口】

【本地窗口】是用来显示当前过程中所有变量的值，它只显示当前过程中可用的变量，如果过程发生改变，则立即窗口所显示的变量也会跟着改变。在中断模式下，单击【视图】/【本地窗口】菜单或单击【调试】工具栏的【本地窗口】按钮，便可以调出【本地窗口】，如图 12-23 所示。

在中断模式下时，通过【本地窗口】，不仅可以查看当前过程和窗体中所使用变量，而且还可以查看当前窗体的有关属性，在图 12-23 所示所示的【本地窗口】中，单击 Me 前面的加号，便可以看到当前窗体的所有属性及其属性值。另外，还可以在【本地窗口】中改变某个变量值或属性值。例如，在图 12-23 所示的【本地窗口】中，先让蓝色光条停在变量 i 这一栏，然后单击数字“ 10 ”，这时变量的值便处于可更改的状态，如图 12-24 所示，如果输入“ 15 ”，则变量 i 的值变为 15。

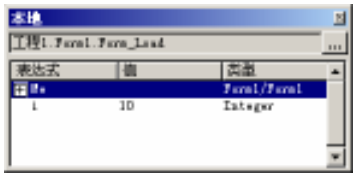


图12-23　【本地窗口】

图12-24　更改变量值

- 【立即窗口】

【立即窗口】可以用来检查某个变量或属性的值，也可以用来给变量或属性赋值。在中断模式下，单击【视图】/【立即窗口】菜单或单击【调试】工具栏的【立即窗口】按钮，便可以调出【立即窗口】，如图 12-25 所示。

图12-25　立即窗口

【立即窗口】只是为用户提供一个命令窗口，用户要想查看某个变量值或某个属性值，还必须在【立即窗口】输入相应的变量名或属性值，并在变量名或属性名前加一个问号（？），输入完毕后，按回车键换行，这时所输入的变量的值或属性值便会显示在下一行，如图 12-26 所示。除了可以查看变量值或属性值之外，【立即窗口】还可以用来更改变量值或属性值，如图 12-27 所示。

图12-26　用【立即窗口】来查看变量或属性值

图12-27　用【立即窗口】来设置变量或属性值

【立即窗口】除了可以在中断模式下可以调用之外，还可以在程序中通过代码来调用【立即窗口】，并将特定的变量值或属性值显示在【立即窗口】中。

【例12-3】在不中断程序的前提下，将【例 12-2】中的变量 i 的值显示到【立即窗口】中。

1. 打开【例 12-2】所建的工程。

2. 在窗体资源资源管理器中单击查看代码按钮，打开【代码】窗口，然后在 Form 的 Load 事件中增加如下带底纹的代码：

```
Private Sub Form_Load()
    Dim i As Integer
    Form1.AutoRedraw = True
    i = 10
    '调用显示数字的子过程
    ShowNum i
    Debug.Print "i="; i
End Sub
```

3. 运行程序，【立即窗口】便会直接显示出来，并在【立即窗口】中显示变量 i 的值，如图 12-28 所示。

在程序中，使用 Debug.Print 语句，便可以将数据直接显示到立即窗口中，而不必先中断程序。具体的语法结构如下：

```
Debug.Print [Items][;]
```

例如，在【例 12-3】中，便是通过语句

```
Debug.Print "i="; i
```

将变量 i 的值直接显示到【立即窗口】的。

要想在【立即窗口】中输入命令，则程序必须处于中断模式下，否则不能对【立即窗口】做任何修改。

- 【监视窗口】

监视窗口是用来监视某一个变量或表达式的，一旦程序进入中断模式时，则在监视窗口中显示所监视对象的值，如图 12-29 所示。

图12-28 直接将变量值显示到【立即窗口】

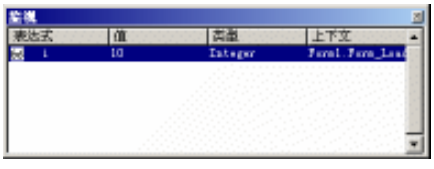


图12-29 【监视窗口】

【例12-4】让【监视窗口】监视【例 12-2】中变量“i”的运行情况。

1. 打开【例 12-2】所建的工程。
2. 单击【调试】/【添加监视】菜单，打开【添加监视】对话框，如图 12-30 所示。
3. 在【表达式】文本框中输入变量“i”，如图 12-30 所示，将变量“i”设为监视对象。
4. 单击【过程】列表框右端的箭头，打开下拉列表，然后单击 Form_Load 列表项，选中该列表项，如图 12-30 所示。
5. 在【监视类型】容器框中选中单选项【当监视值改变时中断】，如图 12-30 所示。
6. 单击对话框的确定按钮，将变量“i”设为监视对象，这时在窗体上会弹出如图 12-31 所示的【监视窗口】。

图12-30　【添加监视】对话框

7. 按 F5 运行程序，程序运行到“ShowNum i”代码行时，便暂停下来，处于中断模式下，并弹出如图 12-32 所示的【监视窗口】。

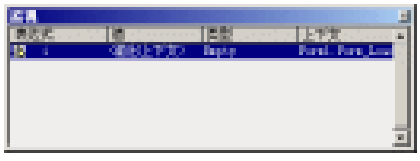


图12-31　【监视窗口】

图12-32　【监视窗口】

要让【监视窗口】来监视某个变量或表达式，必须先将该变量或表达式设为监测对象，具体步骤如下：

1. 单击【调试】/【添加监视】菜单，打开【添加监视】对话框，如图 12-30 所示。
2. 在【表达式】文本框中输入表达式，表达式可以一个变量、属性或其他合法的表达式。
3. 在【上下文】容器框中设置监视范围，包括【工程】和【模块】的设置。
4. 在【监视类型】中设置监视类型。若选中【监视表达式】单选项，则程序不会自动进入中断模式，只监视表达式但不显示其值；若选中【当监视值为真时中断】，则监视到表达式为真时程序会自动进入中断模式，并在【监视窗口】中显示监视表达式的值；若选中【当监视值改变时中断】，则监视表达式的值一旦发生改变，程序便会自动进入中断模式，并在【监视窗口】中显示监视表达式的值。
5. 单击 确定 按钮，完成监视表达式的添加。

12.6　小结

错误是任何编程人员都会遇到的问题，本章主要介绍了如何排除和预防各种错误，让自己编写的应用程序尽可能地完美，当然要排除错误，编程人员除了要认真仔细之外，还要有一定的编程经验。经验是排除任何错误的最佳武器。

本章主要介绍了以下内容：

- 程序工作的 3 种模式及其各自的特点。
- 错误的种类及其产生的原因。
- 编译错误排除方法。
- 实时错误预防的方法。
- 逻辑错误发现的方法。

12.7 习题

一、填空题

1. 在 Visual Basic 6.0 中，程序共有_____、_____、_____3 种工作模式。
2. 在 Visual Basic 6.0 种，错误有_____、_____、_____3 种类型。其中_____最容易被监测到，_____最难以被发现。
3. 实时错误的捕获，可以通过_____语句来显示，退出实时错误处理程序可以用_____语句来完成。

二、问答题

1. 如何设置和取消断点？
2. 逐语句执行和逐过程执行是如何进行的？它们之间有什么区别？
3. 【立即窗口】、【本地窗口】、【监视窗口】各有什么功能？
4. 如何设置监视表达式？

附录 常用资料

F.1 附表一 标准控件常用属性

属性名	功能	说明
Container	返回或设置指定控件的容器（即该控件的父控件）	该控件在设计时不可用，只能通过代码来设置
DragIcon	返回或设置拖放控件时指针的样式（图标）	所加载的文件必须是扩张名为 ico 的文件
DragMode	返回或设置拖放的操作方式	有两个取值：0（vbManual）或 1（vbAutomatic）。取 0 时，手动方式；取 1 时，自动方式。默认值 0
FontName	返回或设置字体名字	Visual Basic6.0 中可用的字体取决于系统的配置、显示设备、打印设备
FontSize	返回或设置字体的大小	
FontTransparent	返回或设置是否以透明的方式显示文本	有两个取值：True 或 False，默认值 True
HasDC	返回或设置是否将显示设备上下文（hDC）分配给控件	有两个取值：True 或 False。取 True，分配；取 False，不分配。默认值 True
hDC	返回一个句柄	该句柄由 Windows 运行环境提供，用于说明窗口绘图环境。语法结构如下： 对象名.hDC
HelpContexID	返回或设置一个与控件关联的上下文编号	用于为应用程序提供上下文有关的帮助
hWnd	返回或设置控件或窗体的句柄	语法结构如下： 对象名.hWnd 该属性常和 hDC 一起配合使用
Index	返回或设置控件数组的编号	只有控件为控件数组中的成员时，该属性才有效
LinkItem	返回或设置传送给接受（客户）端的数据	在动态数据的交换（DDE）时，才有效
LinkMode	返回或设置数据交换的联结模式	
LinkTimeout	设置或返回动态数据交换时等待的响应消息的时间	其设置值以 1/10 秒为单位
MaskColor	返回或设置图片中作为透明的颜色	只有当 UseMaskColor 属性为 True 且 Picture 属性为位图时，才用到该属性
MouseIcon	返回或设置鼠标的样式（图标）	
MousePointer	返回或设置运行时鼠标指针的样式	鼠标的样式有很多种，常用的有：默认样式（0 或 vbDefault）、箭头（vbArrow 或 1）、十字形（vbCrosshair 或 2）等，默认值 0
Name	返回窗体或控件的名字	运行时，该属性为只读，不能更改
OLEDragMode	返回或设置 OLE 拖放操作的模式	有两个取值：0（vbOLEDragManual）或 1（vbOLEDragAutomatic，取 0，手动；取 1，自动。默认值 0
OLEDropMode	返回或设置处理 OLE 操作的方法	

续表

Parent	返回当前控件的父控件（即容器）	语法结构： 对象名.Parent
TabIndex	返回或设置控件的 Tab 键次序	默认情况下，系统按控件添加的先后顺序为其分配 TabIndex 属性值
TabStop	返回或设置是否可以通过使用 Tab 键来获得焦点	有两个取值：True 或 False。默认值 True
Tag	返回或设置控件的标识，该属性取值为字符串	在 Visual Basic6.0 中，除了“名称”属性可以标识控件之外，“Tag”属性也可用来标识控件
ToolTipText	返回或设置一个工具提示，该属性取值为字符串	运行时，当光标移到控件上停留一段时间时，“ToolTipText”属性所设置的字符串便会显示在控件下面的提示框中

F.2 附表二 Visual Basic6.0 常用事件

事件名	响应条件
DragDrop	当控件拖动到指定的对象上并释放鼠标（即完成一个完整的拖放操作）或使用 Drag 方法将其 Action 参数设置为 2 时，该事件被激发
DragOver	拖动控件经过指定对象时，该事件被激发
LinkClose	在动态交换数据结束时，该事件被激发
LinkError	动态交换数据时，数据交换出错，该事件被激发
LinkOpen	动态交换数据被启动时，该事件被激发
OLECompleteDrag	当源控件被放到目标控件中时，该事件被激发
OLEDragDrop	当源控件被放到目标控件中，并由源控件完成放的操作时，该事件被激发。一般的只有当“OLEDropMode”属性设为 1 时，该事件才可被激发
OLEDragOver	当一个控件拖过另外一个控件时，该事件被激发
OLEStarDrag	当控件开始 OLE 拖动时，该事件被激发
Validate	当控件无效或失去焦点时，该事件被激发

F.3 附表三 与文件处理有关的常用函数与语句

函数名或语句名	功能
ChDir 语句	改变当前目录
ChDrive 语句	改变当前驱动器
CurDir 函数	返回当前驱动器的当前目录
Dir 函数	返回与指定的模式、文件名、文件属性或驱动器名称相匹配的文件
MkDir 语句	建立一个新的子目录
RmDir 语句	删除指定的子目录
SetAttr 语句	用于设置文件的属性
GetAttr 语句	返回文件的属性
FileDateTime	返回文件创建或被修改的日期和时间

续表

Seek 语句	设置下一个读/写操作位置
Seek 函数	返回文件当前的读/写操作位置
Loc 函数	返回文件当前的读/写操作位置