



普通高等教育“十一五”国家级规划教材

董华松 编著

Visual Basic 程序设计综合实践

21世纪计算机科学与技术实践型教程

丛书主编 陈明

北京大学出版社



北京大学出版社
PEKING UNIVERSITY PRESS

内 容 简 介

Visual Basic 作为程序设计的入门语言,具有易学易懂的特点。Windows 是当今主流操作系统,而 Visual Basic 作为一种支持可视化程序设计的语言,也是开发 Windows 应用程序最简单易学的语言之一。

本书是一本全面、系统地介绍 Visual Basic 程序设计的教程。全书共分 11 章,介绍了 Visual Basic 程序开发环境、数据类型和表达式、常用内部函数、程序控制结构、常用控件、数组、过程、键盘和鼠标事件过程、菜单与对话框程序设计、图形程序设计、文件处理和数据库程序设计等内容。

本书在编排上由简及繁、由浅入深和循序渐进,力求通俗易懂、简捷实用。全书内容全面,实例丰富。书中所有例题均在 Visual Basic 6.0 上运行通过。

本书可以作为高等院校非计算机专业计算机程序设计课程的教材和计算机专业可视化程序设计教材,也可以作为全国计算机等级考试(二级 Visual Basic)的培训教材和参考书,还可供各类程序设计培训班学员和 Visual Basic 语言自学者参考。

图书在版编目(CIP)数据

Visual Basic 程序设计/闵联营,董华松主编. —北京:北京大学出版社,2006.1

(21 世纪全国应用型本科计算机系列实用规划教材)

ISBN 7-301-10503-7

I. V… II. ①闵…②董… III. BASIC 语言—程序设计—高等学校—教材 IV. TP312

中国版本图书馆 CIP 数据核字(2006)第 003438 号

书 名: Visual Basic 程序设计

著作责任者: 闵联营 董华松 主编

责任编辑: 周 欢

标准书号: ISBN 7-301-10503-7/TP • 0876

出版者: 北京大学出版社

地 址: 北京市海淀区成府路 205 号 100871

网 址: <http://cbs.pku.edu.cn> <http://www.pup6.com>

电 话: 邮购部 62752015 发行部 62750672 编辑部 62750667

电子信箱: pup_6@163.com

排版者: 北京东方人华北大彩印中心 电话: 62754190

印刷者:

发 行 者: 北京大学出版社

经 销 者: 新华书店

787 毫米×1092 毫米 16 开本 16 印张 384 千字

2006 年 1 月第 1 版 2006 年 1 月第 1 次印刷

定 价: 22.00 元

21 世纪全国应用型本科计算机系列实用规划教材

联合编写学校名单（按拼音顺序排名）

- | | |
|------------|--------------|
| 1 安徽财经大学 | 23 杭州师范学院 |
| 2 安徽工业大学 | 24 合肥工业大学 |
| 3 安阳师范学院 | 25 合肥学院 |
| 4 北华大学 | 26 河北经贸大学 |
| 5 北京化工大学 | 27 河南科技学院 |
| 6 北京建筑工程学院 | 28 黑龙江八一农垦大学 |
| 7 北京理工大学 | 29 黑龙江科技学院 |
| 8 渤海大学 | 30 湖南大学 |
| 9 长春大学 | 31 湖北经济学院 |
| 10 长春工业大学 | 32 孝感学院 |
| 11 长春理工大学 | 33 湖州师范学院 |
| 12 长春税务学院 | 34 华北科技学院 |
| 13 滁州学院 | 35 华南师范大学 |
| 14 楚雄师范学院 | 36 华中农业大学 |
| 15 东北电力大学 | 37 华中师范大学 |
| 16 福建工程学院 | 38 华北水利水电学院 |
| 17 福建师范大学 | 39 淮北煤炭师范学院 |
| 18 广西财经学院 | 40 黄石理工学院 |
| 19 桂林工学院 | 41 吉林农业大学 |
| 20 哈尔滨理工大学 | 42 集美大学 |
| 21 海南大学 | 43 江汉大学 |
| 22 韩山师范学院 | 44 江苏科技大学 |

- | | | | |
|----|----------|----|------------|
| 45 | 内蒙古大学 | 64 | 苏州大学 |
| 46 | 南昌工程学院 | 65 | 台州学院 |
| 47 | 南京航空航天大学 | 66 | 太原理工大学 |
| 48 | 南开大学 | 67 | 太原师范学院 |
| 49 | 南阳理工学院 | 68 | 唐山师范学院 |
| 50 | 宁波工程学院 | 69 | 同济大学 |
| 51 | 平顶山学院 | 70 | 皖西学院 |
| 52 | 青岛理工大学 | 71 | 武汉大学 |
| 53 | 青岛科技大学 | 72 | 武汉科技学院 |
| 54 | 青海民族学院 | 73 | 武汉理工大学 |
| 55 | 曲阜师范大学 | 74 | 武夷学院 |
| 56 | 山西大学 | 75 | 忻州师范学院 |
| 57 | 山西广播电视大学 | 76 | 新疆石油学院 |
| 58 | 陕西理工学院 | 77 | 许昌学院 |
| 59 | 上海第二工业大学 | 78 | 玉溪师范学院 |
| 60 | 上海海事大学 | 79 | 浙江工业大学之江学院 |
| 61 | 沈阳大学 | 80 | 衢州广播电视大学 |
| 62 | 沈阳化工学院 | 81 | 中国农业大学 |
| 63 | 石家庄铁道学院 | 82 | 中国石油大学 |

本系列教材编写目的和教学服务

《21 世纪全国应用型本科计算机系列实用规划教材》在全国的各位编写老师的共同辛勤努力下，在编委会主任刘瑞挺教授和其他编审委员会成员的悉心指导下，经过北京大学出版社第六事业部各位编辑的刻苦努力，终于与师生们见面了。

教材编写目的

目前，我国高等教育正迎来一个前所未有的发展机遇期。高等教育的发展已进入到一个新的阶段。高等本科院校也逐渐演变成“研究型、学术型”和“应用型、就业型”两类。

作为知识传承载体的教材，在高等院校的发展过程中起着至关重要的作用。但目前教材建设却远远滞后于应用型人才培养的步伐，许多院校一直沿用偏重于研究型的教材，应用型教材比较缺乏，这势必影响应用型人才的培养。

为顺应高等教育普及化迅速发展的趋势，配合高等院校的教学改革和教材建设，坚持“因材施教”的教学原则，注重理论联系实际，全面促进高等院校教材建设，进一步提高我国高校教材的质量，北京大学出版社大力推出高校“应用型本科”各专业相关教材。本系列教材不仅讲解基础理论技术，更突出工程实际应用，注重**技术与应用**的结合。

本套计算机系列教材的编写思想主要如下：

(1) 要符合学校、学科的计算机课程设置要求。以高等教育的培养目标为依据，注重教材的科学性、实用性、通用性，尽量满足同类专业院校的需求。

(2) 要定位明确。要准确定位教材在人才培养过程中的地位和作用，正确处理系列教材与系列课程、读者层次的关系，面向就业，突出应用。

(3) 合理选材和编排。教材内容应处理好传统内容与现代内容的关系，大力补充新知识、新技术、新工艺、新成果。根据教学内容、学时、教学大纲的要求，制定模块化编写体例，突出重点、难点。

(4) 体现建设“立体化”精品教材的宗旨。提倡为主干课程配套电子教案、学习指导、习题解答、课程设计、毕业设计等教学配套用书。

教学服务

1. 提供教学资源下载。本系列大部分教材中涉及到的实例（习题）的原始图片和其他素材或者是源代码、原始数据等文件，都可以在相关网站上下载。每本教材都配有 PPT 电子教案，老师可随时在网络上下下载并可修改为适合自己教学的 PPT。

2. 提供多媒体课件和教师培训。针对某些重点课程，我们配套有相应的多媒体课件，对大批量使用本套教材的学校，我们会免费提供多媒体课件。另外，我们还将免费提供教师培训名额，不定期组织老师进行培训。

3. 欢迎互动。欢迎使用本系列教材的老师和同学提出意见和建议，有建设性的将给予奖励；同时有教材或者专著出版要求的老师，请与我们联系。

北京大学出版社第六事业部(<http://www.pup6.com>)

信息技术的应用化教育

(代丛书序)

刘瑞挺/文

北京大学出版社第六事业部组编了一套《21 世纪全国应用型本科计算机系列实用规划教材》。为了做好这项工作，他们制订了详细的编写目的、丛书特色、内容要求和风格规范。在内容上强调面向应用、任务驱动、注重实例分析、培养能力；在风格上力求文字精练、脉络清晰、图表丰富、版式明快。

一、组编过程

2004 年 10 月，该部开始策划教材丛书，派出编辑分别深入各地高校，了解教学第一线的情况，物色合适的作者。2005 年 4 月 16 日在北京大学召开了“《21 世纪全国应用型本科计算机系列实用规划教材》研讨会”。来自全国 73 所院校的 102 位教师汇聚一堂，共同商讨应用型本科计算机系列教材建设的思路，并对规划选题进行了分工。2005 年 7 月 21 日在青岛又召开了“《21 世纪全国应用型本科计算机系列实用规划教材》审纲会”。编审委员会成员和 46 个选题的主编、参编，共 100 多位教师参加了会议。审稿会分专业基础课、软件开发与软件工程、硬件与网络技术、计算机应用技术等小组对大纲及部分稿件进行了审定，力争使这套规划教材成为切合当前教学需要的高质量的精品教材。

二、转变观念

为了搞好这套教材，要转变一些重要的观念。

首先，需要转变的观念就是大学及其培养人才的定位。大学并不都是“研究型”的，每个大学生不一定都当科学家。事实上，大多数学校应该是“应用型”的，大学生将直接进入社会基层、生产一线、服务前沿，成为各行各业的实践者和带头人。

其次，应该转变的观念就是教材建设的思路。许多人偏爱于“研究型”的教材，即使写“应用型”教材，也多半是对前者进行删繁就简、避虚就实，这样还不能产生真正“应用型”的教材。因此，以“学科”为中心、追求雄厚“理论基础”的传统应该被以“应用”为导向、追求熟练“实践技能”的思路所取代。

第三，必须转变对计算机技术的认识。20 年前，有人把计算机技术理解为 BASIC 编程；10 年前，有人把 Windows 95 和 Word 称为计算机文化；今天，中小学陆续开出《信息技术》课，有人对此怀疑观望，其实它意义深远。以计算机为核心的信息技术，今后 20 年的发展主题将是在各个领域的应用普及。大学计算机应用型本科的教材建设应该面向信息技术的深入应用，而不是相反，因为信息时代已经不是遥远的未来。

三、信息技能

以计算机为核心的信息技术，从一开始就与应用紧密结合。例如，ENIAC 用于弹道计算，ARPANET 用于资源共享以及核战争时的可靠通信。即使是非常抽象的图灵机模型，也

与第二次世界大战时图灵博士破译纳粹密码的工作相关。

今天的信息技术有三个重要的特点：

第一，信息技术是计算机与通信技术融合的辉煌成果。长期以来，计算机技术和通信技术并行不悖地独立发展。20 世纪后半叶，两者相互渗透，产生了程控电话、数据通信、网络技术、高清晰电视，世界各国构建了全球的、宽带的、网站密布的信息高速公路，出现了无处不在的手机通信和移动办公系统、随身听、数码摄录相机、家庭影院、智能控制系统，还有越来越多的嵌入式系统。人们的工作方式和生活方式都发生了质的飞跃。

第二，信息技术与各行各业紧密结合。我国的职业门类有：农林牧渔、交通运输、生化与制药、地矿与测绘、材料与能源、土建水利、制造、电气信息、环保与安全、轻纺与食品、财经、医药卫生、旅游、公共事业、文化教育、艺术设计传媒、公安、法律，这些门类都需要信息技术。

第三，在发展初期，以计算机为核心的信息技术是一项专门技术，只有专家才需要它、才能掌握它，在专家与平民之间有很深的“信息鸿沟”。今天，信息技术已经不再是只有专家才需要、才能掌握的专门技能，而是普通人都需要、也都能掌握的基本信息技能。但是，“信息鸿沟”也迁移到普通人之间。具有信息优势的学生能良性循环，强者更强。

有了这样广阔的应用信息背景，再造计算机应用型本科的课程体系就有了基础。

四、能力结构

关于应用型计算机人才的能力结构，我们不用“宫殿”模型，而用“雄鹰”模型。前者是建筑学模型，适合描述学科；后者是生物学模型，适合描述人才。“雄鹰”模型包括主体、两翼、头部、尾部等，它有可成长性。

首先，数据是信息技术的主体，数据技术是基本功。通常，数据包括文字、公式、表格、图形、图像、动画、声音、视频等等。因此，你不仅要会录入文章、绘制图表，还应该会采集音乐、编辑视频。大家面对的是多媒体数据，应该能收集它、整理它，数据经过整理就成为有用的信息。

其次，信息技术的两翼是数据库技术和网络技术。为了管理好、使用好数据，就必然用到数据库技术，数据库技术是一切信息管理的基石。为了分享数据和信息，就需要网络技术。有了上述数据主体技术和两个“翅膀”，你应该可以起飞了。

但是能飞多高，能飞多远，还应该编程技术、智能技术、安全技术的支持。这相当于头尾各部分的作用。编程将使大家的信息技能游刃有余。人工智能使你飞得更远，安全技术能让你飞得更稳。

有人可能会责难我们，难道大学本科生还需要学习办公软件的技能吗？他们认为这是让人“笑掉大牙”的事。其实，办公软件是最重要的提高生产效率的应用软件，很容易使用，但各人使用效率的高低则十分悬殊。我们设想，今后大学生在入学前先学会计算机的基本操作，我们再开一门高级办公技术的课，通过严格的行业及个人行为规范，对学生进行应用化训练，养成正确的职业习惯。将来工作时能提高效率、改善质量、降低成本。这决不是贻笑大方的事。

五、初步规划

应用型本科教材的规划是一个长期的战略任务，不是短期的战术行为。因此，目前的规划教材不可能一步到位，还会保留一些传统的基础课。例如，数字电路与逻辑设计、微机原理及接口技术、单片机原理及应用等。即使是纯硬件专业的学生，如何学这些传统硬件课都值得商榷，更何况公共基础课。

我们将分门别类逐步建设好应用型本科的重点课程和教材：

- (1) 基础类教材：信息技术导论，计算机应用基础，高级办公技术，数据与操作，密码与安全，实用数据结构，实用离散数学……；
- (2) 数据库类教材：数据库原理与应用，信息系统集成，数据采掘与知识发现……；
- (3) 网络类教材：计算机网络，因特网技术，网络管理与安全，网站与网页设计……；
- (4) 编程类教材：面向对象程序设计，C++程序设计，Java 程序设计……；
- (5) 提高类教材：软件工程原理及应用，人工智能原理及应用……。

新教材要体现教育观念的转变，系统地研究普通高校教学改革的需求，优先开发其中教学急需、改革方案明确、适用范围较广的教材。注重规划教材的科学性、实用性、易学性，尽量满足同类专业院校的需求。教材内容应处理好传统与现代的关系，补充新知识、新技术、新工艺、新成果。

我相信北京大学出版社在全国各地高校教师的积极支持下，精心设计，严格把关，一定能够建设一批符合应用型人才培养目标的、适应计算机应用型人才培养模式的系列精品教材，而且能建设一体化设计、多种媒体有机结合的立体化教材，为各门课程配套电子教案、学习指导、习题解答、课程设计等辅导资料。让我们共同努力吧！

刘瑞挺教授 曾任中国计算机学会教育培训委员会副主任、教育部理科计算机科学教学指导委员会委员、全国计算机等级考试委员会委员。目前担任的社会职务有：全国高等院校计算机基础教育研究会副会长、全国计算机应用技术证书考试委员会副主任、北京市计算机教育培训中心副理事长。

《21 世纪全国应用型本科计算机系列实用规划教材》

专家编审委员会

(按姓氏笔画排名)

主 任 刘瑞挺

副主任 胡昌振 段禅伦

崔广才 谢红薇

委 员 叶俊民 杨 璐 陈天煌 范冰冰

陈仲民 胡 明 秦 锋 龚声蓉

《21 世纪全国应用型本科计算机系列实用规划教材》

分系列专家编审委员会名单

(按姓氏笔画排名)

计算机应用技术——

主 任 胡昌振

副主任 杨 璐 龚声蓉

委 员 云 敏 马秀峰 李 明 肖淑芬 周松林

杨长生 钟 声 赵忠孝 高 巍

软件开发与软件工程——

主 任 谢红薇

副主任 叶俊民 陈天煌

委 员 王建国 孙 辉 吕海莲 李福亮 何朝阳

张世明 陈佛敏 贺 华 赵绪辉 徐庆生

徐 辉

硬件与网络技术——

主 任 崔广才

副主任 范冰冰 胡 明

委 员 龙冬云 冯嘉礼 曲朝阳 汤 惟 张有谊

董春游 程小辉

专业基础课——

主 任 段禅伦

副主任 陈仲民 秦 锋

委 员 王昆仑 王 虹 仇 汶 田敬军 刘克成

朴春慧 吴晓光 苏守宝 陈付贵 咎风彪

谭水木 魏仕民

前言

Visual Basic 是近年来得到迅速推广和应用的一种可视化的计算机高级编程语言，它适于在 Windows 平台上开发应用程序。Visual Basic 因其功能强大、容易掌握，受到广大初学者和程序开发人员的喜爱。

目前不少高校开设了 Visual Basic 程序设计课程，许多学生通过短短的数十小时的学习，就能用它编写一些相对简单的 Visual Basic 应用程序，为今后的深入应用打下很好的基础。事实表明，Visual Basic 作为大学生（尤其是非计算机专业的大学生）学习程序设计的第二种语言是比较合适的。为此，我们在近几年教学实践的基础上编写了这本书。

本书介绍了 Visual Basic 程序设计的基础知识，主要包括界面设计和程序代码的设计。

全书共分 11 章。第 1 章是 Visual Basic 的概述，介绍 Visual Basic 的集成开发环境、Visual Basic 开发应用程序的一般开发过程等内容；第 2 章介绍 Visual Basic 程序基本的语法单位和语法规则，包括 Visual Basic 的数据类型、常量和变量、运算符和表达式、常用内部函数等；第 3 章主要讨论 Visual Basic 组成程序流程控制的三种基本结构及相关语句；第 4 章集中地介绍 Visual Basic 工具箱中的常用控件的基本属性和方法；第 5 章讨论数组的定义和使用方法；第 6 章介绍过程的有关概念和使用方法；第 7 章具体探讨键盘和鼠标事件过程；第 8 章介绍如何使用 Visual Basic 提供的菜单技术、工具栏技术、对话框技术、多重文档技术设计应用程序界面；第 9 章介绍 Visual Basic 基本的图形处理技术；第 10 章介绍 Visual Basic 对数据文件的组织处理方法，以及用于文件操作的有关控件；第 11 章介绍数据库的初步概念以及 Visual Basic 中对数据库进行处理的基本方法。

本书针对初学者的特点，在体系结构和内容上注意了由简及繁、由浅入深、循序渐进、深入浅出以及理论与实践密切结合。全书内容全面，实例丰富。书中所有例题均在 Visual Basic 6.0 上运行通过。

本书由武汉理工大学闵联营老师和中国石油大学董华松老师主编。第 1、2 章由江汉大学刘全老师编写，第 3、8 章由董华松老师编写，第 4、5 章由浙江工业大学之江学院张惠老师编写，第 6、7 和 9 章由闵联营老师编写，第 10 和 11 章分别由长春大学宋雅娟老师和华北科技学院任占营老师编写。最后由闵联营老师统编全书。

由于作者水平有限，书中难免有不足之处，恳请读者批评指正。

编者
2005 年 9 月

目 录

第 1 章 Visual Basic 概述.....	1	2.1.5 对象型数据.....	22
1.1 Visual Basic 的特点和版本	1	2.1.6 可变类型数据.....	23
1.1.1 Visual Basic 的特点	1	2.2 常量和变量.....	23
1.1.2 Visual Basic 6.0 版本简介	2	2.2.1 常量.....	23
1.2 Visual Basic 的启动与退出	3	2.2.2 变量.....	24
1.3 Visual Basic 的集成开发环境	4	2.3 运算符与表达式.....	26
1.3.1 标题栏	5	2.3.1 算术运算符与算术表达式	26
1.3.2 菜单栏	5	2.3.2 字符串运算符与字符串 表达式.....	27
1.3.3 工具栏	6	2.3.3 关系运算符与关系表达式	28
1.3.4 工具箱	7	2.3.4 逻辑运算符与逻辑表达式	28
1.3.5 窗口	8	2.3.5 表达式的运算顺序	29
1.4 可视化编程的基本概念.....	10	2.4 小结	30
1.4.1 对象	10	2.5 习题	30
1.4.2 属性	10	第 3 章 Visual Basic 程序控制结构.....	32
1.4.3 事件	11	3.1 顺序结构	32
1.4.4 方法	11	3.1.1 赋值语句.....	32
1.5 创建窗体	12	3.1.2 数据输入和输出	33
1.5.1 窗体的属性	12	3.1.3 注释、暂停与程序结束语句 ...	35
1.5.2 窗体的方法	14	3.1.4 应用举例.....	36
1.5.3 窗体的事件	15	3.2 选择结构	36
1.6 用 Visual Basic 开发应用程序	15	3.2.1 单行结构条件语句.....	37
1.6.1 用 Visual Basic 开发应用 程序的一般步骤.....	15	3.2.2 块结构条件语句.....	38
1.6.2 编写一个简单的 Visual Basic 应用程序	17	3.2.3 多分支选择语句.....	40
1.7 小结	18	3.2.4 应用举例.....	42
1.8 习题	19	3.3 循环结构	43
第 2 章 Visual Basic 程序设计基础	21	3.3.1 For...Next 循环结构.....	43
2.1 数据类型	21	3.3.2 Do...Loop 循环结构.....	45
2.1.1 数值型数据	21	3.3.3 While...Wend 循环结构.....	48
2.1.2 字符串型数据.....	22	3.3.4 循环的嵌套.....	49
2.1.3 布尔型数据	22	3.3.5 应用举例.....	50
2.1.4 日期型数据	22	3.4 小结	51
		3.5 习题	53

第4章 Visual Basic 常用内部控件	56	5.5.2 控件数组的使用	95
4.1 概述	56	5.6 小结	97
4.2 命令按钮	57	5.7 习题	98
4.2.1 命令按钮的常用属性	57	第6章 过程	102
4.2.2 命令按钮的常用事件	58	6.1 过程的定义	102
4.3 文本控件	59	6.1.1 Sub 过程	102
4.3.1 标签	59	6.1.2 Function 过程	104
4.3.2 文本框	60	6.2 过程的调用	105
4.4 单选按钮、复选框和框架	63	6.2.1 调用 Sub 事件过程	106
4.4.1 单选按钮	63	6.2.2 调用 Sub 通用过程	106
4.4.2 复选框	65	6.2.3 调用 Function 过程	107
4.4.3 框架	67	6.3 参数的传递	108
4.5 列表框和组合框	68	6.3.1 形参和实参	108
4.5.1 列表框	68	6.3.2 参数按值传递和按地址 传递	109
4.5.2 组合框	72	6.3.3 数组参数	110
4.6 图像框和图片框	74	6.3.4 可选参数与可变参数	112
4.6.1 图片框	74	6.3.5 对象参数	113
4.6.2 图像框	76	6.4 递归过程	114
4.7 滚动条	77	6.5 变量的作用域与生存期	117
4.7.1 滚动条的常用属性	77	6.5.1 变量的作用域	117
4.7.2 滚动条的常用事件	78	6.5.2 变量的生存期	120
4.8 定时器	79	6.6 小结	121
4.9 小结	80	6.7 习题	122
4.10 习题	81	第7章 鼠标和键盘事件过程	126
第5章 数组	85	7.1 鼠标	126
5.1 数组的基本概念	85	7.1.1 鼠标事件	126
5.1.1 数组与数组元素	85	7.1.2 鼠标光标形状	128
5.1.2 数组的维数	85	7.2 键盘	129
5.2 数组的定义	86	7.2.1 KeyDown 和 KeyUP 事件	129
5.2.1 静态数组的定义	86	7.2.2 KeyPress 事件	130
5.2.2 动态数组的定义	86	7.3 拖放	132
5.3 数组的基本操作	87	7.3.1 与拖放有关的属性、事件 和方法	132
5.3.1 数组元素的输入与输出	87	7.3.2 OLE 拖放	135
5.3.2 数组元素的复制	89	7.4 小结	137
5.3.3 保留动态数组的内容	90	7.5 习题	137
5.4 数组应用举例	91		
5.5 控件数组	94		
5.5.1 控件数组的建立	94		

第 8 章 界面设计	142	9.4.3 填充颜色属性和填充 样式属性	185
8.1 菜单的设计	142	9.4.4 自动重画属性	186
8.1.1 下拉式菜单	142	9.4.5 Paint 事件	187
8.1.2 弹出式菜单	144	9.5 小结	188
8.1.3 菜单事件与菜单命令	145	9.6 习题	189
8.2 对话框的设计	147	第 10 章 文件	191
8.2.1 自定义对话框	147	10.1 文件的基本概念	191
8.2.2 通用对话框	148	10.1.1 文件的类型	191
8.2.3 通用对话框控件的使用	153	10.1.2 文件访问函数和语句	191
8.3 状态栏的设计	154	10.2 顺序文件	192
8.4 工具栏的设计	157	10.2.1 顺序文件的打开与关闭	192
8.4.1 使用手工方式制作工具栏	157	10.2.2 顺序文件的读写	193
8.4.2 使用工具栏控件制作 工具栏	158	10.3 随机文件	195
8.4.3 文档编辑器的实现	161	10.3.1 定义数据类型和变量声明 ...	195
8.5 多文档界面设计	164	10.3.2 随机文件的打开与关闭	195
8.5.1 多文档界面(MDI)	164	10.3.3 随机文件的读写操作	196
8.5.2 建立多文档界面	165	10.4 文件系统控件	196
8.5.3 创建 MDI 应用程序的菜单 ...	166	10.4.1 驱动器列表框控件	196
8.6 Visual Basic 的工程结构	167	10.4.2 目录列表框控件	197
8.7 小结	167	10.4.3 文件列表框控件	198
8.8 习题	168	10.5 文件系统对象模型	200
第 9 章 Visual Basic 图形设计	171	10.5.1 文件系统对象模型概述	200
9.1 图形设计基础	171	10.5.2 管理驱动器	201
9.1.1 坐标系统	171	10.5.3 管理文件夹	202
9.1.2 颜色	173	10.5.4 管理文件	203
9.2 图形控件	176	10.6 文件应用举例	205
9.2.1 Shape 控件	176	10.7 小结	207
9.2.2 Line 控件	177	10.8 习题	207
9.3 绘图方法	178	第 11 章 数据库程序设计	209
9.3.1 画点方法	178	11.1 数据库和 SQL 语言基础	209
9.3.2 画直线、矩形方法(Line)	180	11.1.1 数据库简介	209
9.3.3 画圆方法(Circle)	181	11.1.2 结构化查询语言	210
9.3.4 PaintPicture 方法	183	11.2 可视化数据管理器	212
9.4 与绘图有关的常用属性和事件	185	11.2.1 启动可视化数据管理器	212
9.4.1 清除图形方法	185	11.2.2 新建数据库	213
9.4.2 线宽属性和线型属性	185	11.2.3 打开数据库	214

11.2.4	添加数据表.....	214	11.4	使用 ADO 访问数据	225
11.2.5	数据的增加、删除和修改	218	11.4.1	ADO 对象模型	225
11.2.6	数据的查询.....	219	11.4.2	ADODC 控件.....	226
11.2.7	数据窗体设计器.....	221	11.5	应用示例.....	229
11.3	数据控件和数据绑定控件.....	221	11.6	小结	232
11.3.1	数据控件	222	11.7	习题	232
11.3.2	Recordset 对象的属性 和方法	222	参考答案		234
11.3.3	数据绑定控件.....	223	参考文献		239

第 1 章 Visual Basic 概述

教学提示: Visual Basic 语言是 Microsoft Windows 操作系统环境下的程序开发工具, 是以 BASIC 语言为基础、以事件驱动作为运行机制的可视化程序设计语言。迄今为止, Visual Basic 经历了几次升级, 它的功能也日益强大和完善。1998 年 Microsoft 公司又推出了 Visual Basic 6.0 版本, 本书就是以 Visual Basic 6.0 为背景, 详细介绍利用 Visual Basic 进行程序设计的方法。

教学要求: 通过本章的学习, 了解 Visual Basic 的特点, 熟悉 Visual Basic 的集成开发环境, 掌握建立一个 Visual Basic 应用程序的方法。

1.1 Visual Basic 的特点和版本

1.1.1 Visual Basic 的特点

Visual Basic 是一种面向对象的程序设计语言, 与传统的程序设计语言相比, Visual Basic 不仅简单易学, 而且功能比较强大, 在许多方面有了重大的改进和突破。在这里将介绍 Visual Basic 的几个最基本的特点。

1. 可视化

用传统的高级语言编写程序时, 对界面的设计和算法的实现, 常常需要编写大量的代码来完成。并且只有在该程序运行时, 才能看到用户的界面效果, 如果不满意, 又需要回到设计阶段重新设计, 这样的程序调试方式影响了编程效率。然而, Visual Basic 则不一样, 它为用户编写应用程序提供了可视化的集成开发环境, 使用户不需要编写大量代码去描述界面元素的外观和位置, 只要把系统预先建立的元素对象放在窗口中, 并对该元素的属性进行设置, 就可以快速地构造出美观实用的用户界面。即运行后的实际效果与预先“画”出的界面是一样的。因此, 用户可以方便地设计美观、友好的界面, 这样就大大简化了界面设计, 同时也提高了编程效率。

2. 事件驱动

传统的编程方式是面向过程的, 设计人员必须将要处理的事物编写出一个完整的程序, 计算机按照程序的流程运行, 直到程序的结束语句为止。在这种编程方式下, 程序设计人员必须十分周全地考虑程序运行过程中的每一个细节, 否则稍有不慎就有可能造成不可预见的错误。然而 Visual Basic 改变了这种编程方式, 它采用面向对象的程序设计方法, 即通过事件来执行对象的操作。每一个对象都能响应多个不同的事件。每个事件都能驱动一段

程序代码。例如，命令按钮是一个对象，当用户单击该按钮时，将产生一个“单击”事件，而在产生该事件时，将执行一段设计人员预先设计好的程序，用来实现指定的功能。

用 Visual Basic 设计大型应用程序时，不必建立具有统一控制的、包罗万象的大程序，只需要编写若干个微小的子程序，即事件过程。这些过程分别面向不同的对象。由用户操作引发某个事件来驱动完成某种特定的功能，或由来自系统的消息触发来执行指定的操作，这样使得程序编制工作变得相对简单，提高了编程效率。

3. 交互式

传统高级语言编程一般都要经过 3 个步骤，即编码、编译和测试代码，其中每一步还需要调用专门的处理程序，而 Visual Basic 与传统的高级语言不同，它将这 3 个步骤的操作都集中在它的集成开发环境内统一处理，使得 3 个步骤之间不再有明显的界限，大大方便了设计人员的使用。

在大多数语言中，如果设计人员在编写代码时发生错误，则只有在该程序编译时，错误才会被编译器捕获，此时设计人员必须查找并改正错误，然后再一次进行编译，对于每一个发现的错误都要重复这样的过程。而 Visual Basic 则不同，它采用交互式的在线检测方式，即在设计人员输入代码时，便对其进行解释，即时捕获并突出显示其语法或拼写错误，使设计人员能及时发现错误并改正错误。

4. 可扩充性

Visual Basic 是一种高度可扩充的语言，它为用户提供了多种途径来扩充其功能，主要表现为支持其他软件开发商为扩充其功能而开发的外部控件，只要这些外部控件的文件扩展名为.ocx 就可以加入到 Visual Basic 系统中，以扩充其功能。Visual Basic 还提供了 OLE (Object Linking and Embedding, 对象的链接与嵌入)功能。利用这一功能在 Visual Basic 的应用程序中，可以使用其他 Windows 应用程序对象的某些功能，例如，用户在建立一个 Visual Basic 应用程序时，可以使用 Microsoft Excel 建立一个计算表格。

另外 Visual Basic 还支持动态交换和动态链接技术，通过动态数据交换(DDE)的编程技术，Visual Basic 开发的应用程序能与其他 Windows 应用程序之间建立数据通信，通过动态链接库技术，在 Visual Basic 程序中可方便地调用用 C 语言或汇编语言编写的函数，也可调用 Windows 的应用程序接口(API)函数，利用这些 API 函数，可大大增强 Visual Basic 的编程能力，并可实现一些用 Visual Basic 语言本身不能实现的一些特殊功能。

1.1.2 Visual Basic 6.0 版本简介

在 Visual Basic 6.0 中提供了 3 种版本，即学习版、专业版和企业版。这 3 种不同的版本分别可以满足不同的设计人员对开发的需要，用户可以根据需要选择不同的版本。

1. 学习版

Visual Basic 6.0 的基础版本，该版本主要是为初学者开发的，用它可以开发基于 Windows 9x/NT 的应用程序，该版本包含所有内部控件。

2. 专业版

主要是为专业人员创建基于客户机/服务器应用程序而设计的，该版本包括学习版的全部功能以及 ActiveX 控件、Internet 控件、Crystal Report Writer 控件等开发工具。

3. 企业版

该版本的用户主要是专业软件开发人员，包括了专业版的全部功能，同时具有自动化管理、数据库、管理工具和 Microsoft Visual Source soft 面向工程版的控制系统等。该版本主要用于创建更高级的分布式、高性能的客户机/服务器或 Internet 上的应用程序。

1.2 Visual Basic 的启动与退出

Visual Basic 作为 Visual Studio 6.0 套装软件中的成员，可以和 Visual Studio 6.0 一起安装，也可以单独安装，单独安装的 Visual Basic 6.0 中文版包括 4 张光盘，其中 2 张为 MSDN (联机帮助)。

1. 启动 Visual Basic 6.0

Visual Basic 6.0 的启动方式主要有 4 种。

(1) 单击 Windows 桌面左下角的【开始】按钮，执行【开始】→【程序】→Visual Basic 6.0 菜单操作。

(2) 建立启动 Visual Basic 6.0 的快捷方式，通过快捷方式图标启动 Visual Basic 6.0。

(3) 使用【开始】菜单中的【运行】命令，在【打开】栏内输入“C:\Program Files\Microsoft Visual Studio\VB98\VB6.EXE”，单击【确定】按钮，即可启动 Visual Basic 6.0。

在成功启动 Visual Basic 6.0 之后，屏幕上会显示一个【新建工程】对话框，如图 1.1 所示。



图 1.1 【新建工程】对话框

【新建工程】对话框中有 3 个标签，单击它们可打开相应的选项卡。

① 【新建】选项卡：创建新的 Visual Basic 6.0 应用程序工程。该选项卡中有若干个工程类型，在这里对这几种类型进行说明。

- 标准 EXE：用来创建一个标准的 EXE 文件。
- ActiveX EXE：用来创建一个 ActiveX 可执行文件。
- ActiveX DLL：用于创建一个与 ActiveX EXE 功能相同的 DLL 文件。
- ActiveX 控件：用来创建一个 ActiveX 控件。
- Visual Basic 应用程序向导：用于帮助用户建立应用程序框架，使用户可以快速建立一个具有基本功能的应用程序。
- 数据工程：用于建立一个数据工程。
- IIS 应用程序：IIS 应用程序是一种 Visual Basic 应用程序，它组合了 HTML 技术和动态的、基于浏览器的应用程序的编译 Visual Basic 代码技术。IIS 应用程序位于 Web 服务器上，从浏览器接收请求，并运行与请求相关的代码，然后向浏览器发出响应请求。
- 外接程序：用于建立自定义的 Visual Basic 的 IDE 外接程序。
- ActiveX 文档 EXE 和 ActiveX 文档 DLL：ActiveX 文档相当于可以在支持超链接环境下运行的 Visual Basic 程序。
- DHTML 应用程序：与 IIS 应用程序相似，只是在客户端的浏览器上，解释与响应浏览器上终端的用户操作。
- Visual Basic 企业版控件：该选择不是用来建立应用程序，而是用来在工具箱中加入企业版控件图标。

② 【现存】选项卡：选择和打开现有的工程，继续进行编辑、修改和调试。

③ 【最新】选项卡：列出最近使用过的工程。

如果不希望 Visual Basic 每次启动时都出现该对话框，可以选择该对话框下方的【不再显示这个对话框】复选框，在这种情况下，集成开发环境每次启动时，会自动创建一个类型为“标准 EXE”的工程。

2. Visual Basic 的退出

退出 Visual Basic 的方法主要有 3 种。

- (1) 在 Visual Basic 窗口中，选择【文件】菜单中的【退出】命令。
- (2) 通过组合键 Alt+Q 或 Alt+F4 也可退出 Visual Basic 集成环境。
- (3) 单击 Visual Basic 窗口右上角的关闭按钮。

无论采用何种方式退出 Visual Basic，系统都会在退出集成开发环境时，检查目前打开的工程是否被修改。如果用户对工程做了修改，则系统会提示用户保存其最新版本。

1.3 Visual Basic 的集成开发环境

和大多数开发工具一样，Visual Basic 也提供了一个集成开发环境，在这样一个工作平台

上,用户可以完成应用程序的设计、编辑、编译及调试等工作,因此,熟练掌握 Visual Basic 集成开发环境是学习 Visual Basic 的第一步。

如前所述, Visual Basic 被启动后,用户在对话框中选择一个要建立的工程类型,单击【打开】按钮,就进入了 Visual Basic 的集成开发环境(如图 1.2 所示)。下面的 Visual Basic 集成开发环境界面是由以下几部分构成的。

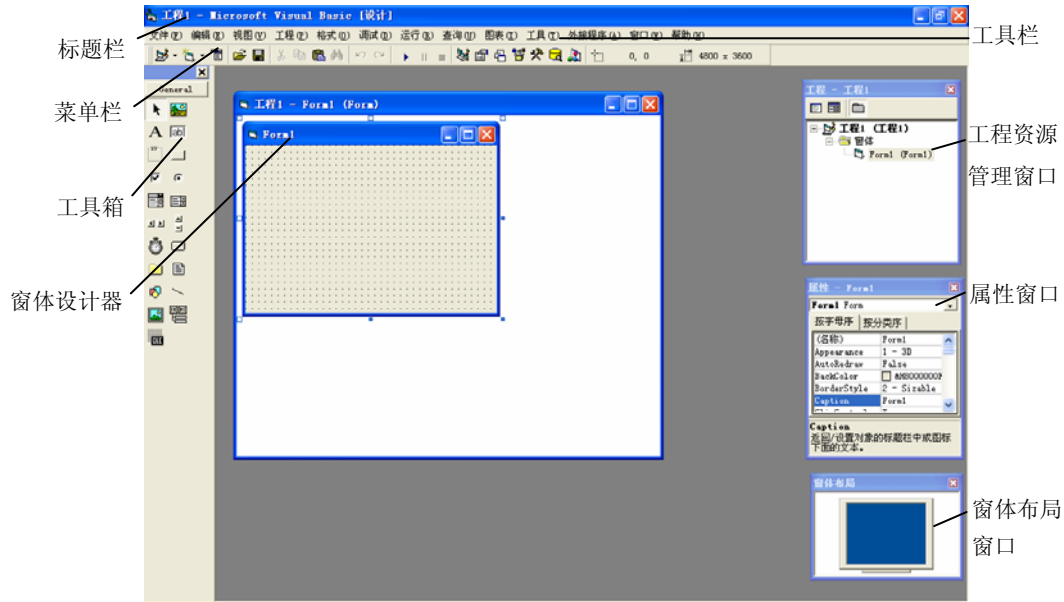


图 1.2 Visual Basic 集成开发环境

1.3.1 标题栏

标题栏是 Visual Basic 集成环境窗口顶部的水平条,在标题栏中显示了当前操作的工程名称以及 Visual Basic 的工作模式,在 Visual Basic 中有 3 种工作模式。

(1) 设计模式:在该模式下可进行用户界面的设计和代码的编写。进入设计模式时,在标题栏中显示“设计”字样。

(2) 运行模式:在该模式下可运行 Visual Basic 应用程序,但不可编辑代码,也不可编辑界面。进入运行模式时,在标题栏中显示“运行”字样。

(3) 中断模式:在该模式下可暂时中断应用程序的执行,而且可编辑代码,但不可编辑用户界面。进入中断模式时,在标题栏中显示 break 字样。

在标题栏中除了显示工程的名称和工作模式之外,在标题栏的最左端还有窗口控制菜单框,在标题栏的最右边还有最大化按钮、最小化按钮和关闭按钮。

1.3.2 菜单栏

Visual Basic 集成环境窗口的第二行就是菜单栏,使用菜单栏中的菜单就可以访问 Visual Basic 中的所有功能。

在菜单栏上共有 13 个菜单(即文件、编辑、视图、工程、格式、调试、运行、查询、图表、工具、外接程序、窗口和帮助),每个菜单的功能如表 1-1 所示。

表 1-1 Visual Basic 菜单的功能一览表

菜 单 名	功 能
文件	包括文件的打开、删除、保存和加入窗体以及生成执行文件等功能
编辑	提供各种编辑功能
视图	提供显示或隐藏各种视图
工程	包括将窗体、模块加入当前工程等功能
格式	对界面设计的辅助控制，如控件对齐方式、间距的设置等
调试	提供对程序代码进行调试的各种方法
运行	执行、中断和停止程序
查询	实现与数据库有关的查询
图表	实现与图表有关的操作
工具	主要包括三方面的功能：对集成开发环境进行定制、向程序代码中添加过程、激活应用程序的菜单编辑器
外接程序	主要包括两方面的功能：Visual Basic 环境下的数据库管理器、外部程序管理器
窗口	设置 Visual Basic 子窗口在主窗口中的排列方式
帮助	提供 Visual Basic 的联机帮助

1.3.3 工具栏

工具栏提供了在编辑环境下快速访问常用命令的方法，当光标指向工具栏上的按钮时，会显示工具按钮的名称及功能，单击工具栏上的按钮，将执行该按钮所对应的功能。

Visual Basic 6.0 中提供了 4 种工具栏：【标准】工具栏、【调试】工具栏、【编辑】工具栏和【窗体编辑器】工具栏。

启动 Visual Basic 后，系统默认主窗口中只显示【标准】工具栏，其他工具栏可以通过【视图】菜单中的【工具栏】命令打开和关闭。

图 1.3 所示的是【标准】工具栏，它列出了 Visual Basic 应用程序中最常用的命令。

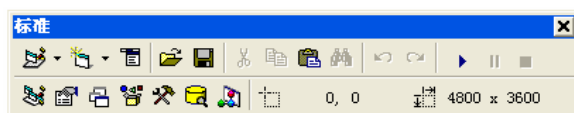


图 1.3 【标准】工具栏

图 1.4 所示的是【调试】工具栏，它用于调试程序，主要功能包括程序的运行、暂停和停止等。可以通过【视图】菜单中的【工具栏】命令打开和关闭。

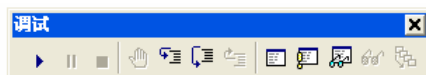


图 1.4 【调试】工具栏

图 1.5 所示的是【编辑】工具栏，它用于对用户编写的程序或用户建立的各种对象进行编辑工作。可以通过【视图】菜单中的【工具栏】命令打开和关闭。

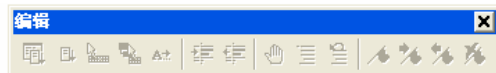


图 1.5 【编辑】工具栏

图 1.6 所示的【窗体编辑器】工具栏，它用于对控件的大小、对齐方式等的设置。可以通过【视图】菜单中的【工具栏】命令打开和关闭。

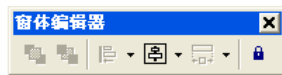


图 1.6 【窗体编辑器】工具栏

1.3.4 工具箱

控件是构成 Visual Basic 应用程序和用户界面的基本组成部分。工具箱中列出了 Visual Basic 的常用控件，不同的图标代表不同的控件类型，因此，可以说工具箱是控件的选用区。将控件放置到窗体的表面有两种操作方法：一种是先单击控件工具箱上的某个控件，然后使用鼠标拖动的方法将该控件在窗体表面上画出来；另一种是双击控件工具箱上的某一个控件，该控件对象就会自动出现在窗体中央，其大小是默认的。图 1.7 列出了所有 Visual Basic 内部控件。需要说明的是，图中左上角的箭头不是控件，单击它可以把鼠标指针由其他形状变为箭头形状。



图 1.7 Visual Basic 的工具箱

工具箱中除了有 Visual Basic 内部控件，还可以添加 Active 控件(又称为外部控件)，向工具箱中添加控件的步骤如下。

(1) 将鼠标指向控件窗口的任意位置，单击右键，显示快捷菜单，然后单击【部件】选项，此时，屏幕上显示如图 1.8 所示的【部件】对话框，在列表框中显示出可以使用的外部控件列表。

(2) 选中需要添加到工具箱的控件，然后单击【确定】按钮，选定的控件就会出现在工具箱中。

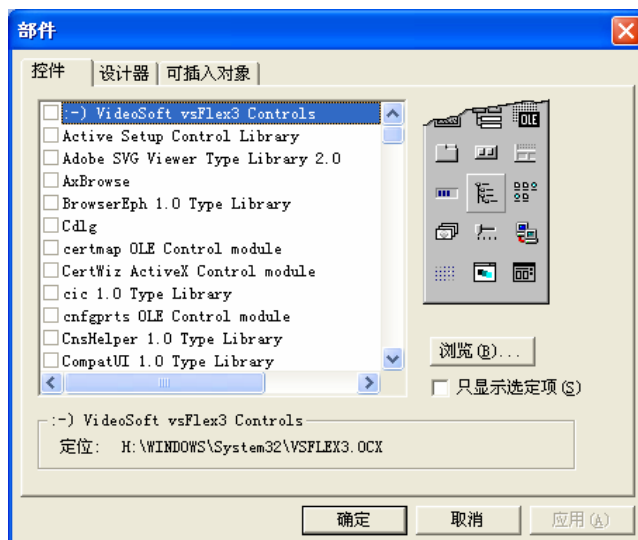


图 1.8 【部件】对话框

1.3.5 窗口

除了图 1.2 所示的 Visual Basic 集成开发环境窗口以外，Visual Basic 还有一些其他的窗口，它们分别是窗体设计器窗口、工程资源管理器窗口、属性窗口、代码编辑器窗口、窗体布局窗口和立即窗口。

1. 窗体设计器窗口

窗体设计器窗口简称窗体(Form)，它是 Visual Basic 最基本的对象，提供与用户交互的窗口，每当 Visual Basic 启动时，系统会打开一个窗体，默认名称为 Form1，用户也可以通过属性窗口的 Name(名称)属性设置来改变窗体名称。如果应用程序需要多个窗体，可以选择【工程】→【添加窗体】菜单创建新窗体。

在程序设计时，程序员可根据程序界面的要求，从工具箱中选择所需要的内部控件，并在窗体中画出来，以形成程序运行时的用户界面窗口。

2. 工程资源管理窗口

在 Visual Basic 集成开发环境窗口右侧的第一个窗口就是工程资源管理窗口，在这个窗口中，包含有建立应用程序所需要的文件清单，在工程资源管理窗口中的文件可以分为以下几类。

(1) 工程文件和工程组文件(文件的扩展名为.vbp 和.vbg)

工程是一个用来建立、保存和管理应用程序中的各种相关信息的管理系统，同时也是应用程序的集合体，每个工程对应一个工程文件，它的扩展名为.vbp。

当一个程序包括两个以上的工程时，这些工程就构成了一个工程组，工程组的扩展名为.vbg。

使用【文件】菜单中的【新建工程】命令可以建立一个工程，使用【打开工程】命令可以打开一个已有的工程，而使用【添加工程】命令则可以添加一个工程。

(2) 窗体文件(文件的扩展名为.frm)

每一个窗体对应一个窗体文件，窗体及其控件和其他信息都存放在该窗体文件中，一个应用程序可以有多个窗体(最多可达 255 个)，它的扩展名为.frm。执行【工程】菜单中的【添加窗体】命令可以添加一个窗体，而执行【工程】菜单中的【移除 form】命令可以删除当前窗体。

(3) 标准模块文件(文件的扩展名为.bas)

标准模块文件是纯代码文件，由程序代码组成，主要用来声明全局变量和定义一些通用的过程。它不属于任何窗体，可以被不同的窗体程序调用。标准模块通过【工程】菜单中的【添加模块】命令建立。

(4) 类模块文件(文件的扩展名为.cls)

类模块用来定义和保存用户根据程序设计的需要建立的类代码，每一类都用一个文件保存，它的文件扩展名为.cls。通过【工程】菜单中的【添加类模块】命令可以建立类模块。

(5) 资源文件(文件的扩展名为.res)

资源文件是纯代码文件，其中可以存放文本、图片、声音等多种资源文件。

除以上几类文件以外，在工程资源管理窗口的顶部还有 3 个按钮。

- 【查看代码】按钮：用来显示相应文件的代码。
- 【查看对象】按钮：用来显示相应的窗体。
- 【切换文件夹】按钮：用来显示各类文件所在的文件夹。

3. 属性窗口

属性窗口用于设定对象的属性，通过【视图】菜单中的【属性窗口】命令，即可打开属性窗口。该窗口由对象选择框、属性显示排列方式、属性列表框及当前属性的解释框 4 部分组成。

(1) 对象选择框

位于属性窗口的上方，在这一栏的右侧有一个下拉按钮，单击它便会显示对象列表。在对象列表中选择一个对象，该对象的属性就会显示在下方的属性列表框中。

(2) 属性显示排列方式

它有两个标签：【按字母序】和【按分类序】，因此可以按字母方式排列对象属性和按对象属性分类排列对象属性。

(3) 属性列表框

位于属性窗口的中间，分为左、右两栏，左边显示的是属性的名称，右边显示的是属性值。设置属性值可以使用以下 3 种方法。

- 通过键盘直接输入属性值。
- 单击属性值右边的下拉箭头，在列出的选项中选择。
- 单击属性值右边的选择按钮，打开一个对话框来设置属性。

(4) 属性解释框

位于属性窗口的底部，每选中一种对象属性，在其下方的属性解释框中就会显示该属性的名称及功能。

4. 代码编辑窗口

用来输入应用程序代码的编辑器，应用程序的每个窗体或代码模块都有一个单独的代码编辑窗口。用户可通过工程资源管理窗口自由地在窗体设计窗口或代码编辑窗口之间进行切换，双击某一个对象也可进入代码编辑窗口。如图 1.9 所示的是代码编辑窗口。

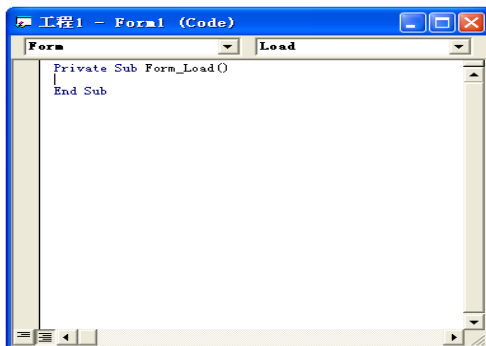


图 1.9 代码编辑窗口

5. 窗体布局窗口

在窗体布局窗口中可以使用表示屏幕小图像来布置应用程序中各窗体的位置，设计时可以使用鼠标把窗体拖动到一个新的位置，运行时窗体就会定位到新位置上。

6. 立即窗口

立即窗口是为调试应用程序提供的，用户可以直接在该窗口中利用 **Print** 方法显示表达式的值。

1.4 可视化编程的基本概念

1.4.1 对象

对象(Object)是代码和数据的集合，也是对具有某些相同特性的具体事物的抽象，对象在现实生活中是很常见的，比如人、电话等都是对象。

在 Visual Basic 中对象分为两类，一类是由系统设计好的，称为预定义对象，用户可以直接使用或对其进行操作，比如窗体及各种内部控件等都是 Visual Basic 预定义对象；另一类由用户定义的，称为用户对象，程序员可以定义和建立用户对象。

从可视化编程的角度来看，对象是一个具有某种属性和方法并能响应外部事件的实体。一个对象建立以后，操作就是通过与对象有关的属性、事件和方法来描述的。

1.4.2 属性

对象的属性用来表示对象的状态以及控制对象的外观和行为。可以通过改变对象的属性值来改变对象的特征。比如可以改变对象的颜色、大小属性来改变该对象的外观特征。

对象的属性有两个基本特点。

- 不同的对象有不同的属性，比如，标签控件具有标题属性，而文本框控件则不具备这个属性。
- 同一个对象有多个不同的属性，比如命令按钮，它不仅具有 Caption(标题)属性，还具有其他的属性，如 Name(名称)、Color(颜色)、Visible(可见性)等。

对象的属性可通过以下两种方法设置。

- (1) 在程序设计阶段，可通过属性窗口设置对象的属性值。
- (2) 在程序运行阶段，由程序代码设置对象的属性值，其格式为：

[<对象名>.]<属性名>=<属性值>

例如：要将一个名为 Command1 的命令按钮的 Caption 属性值设置为“取消”，则可用以下语句：

```
Command1.Caption="取消"
```

其中 Command1 为对象名，Caption 为属性，字符串“取消”就是设定的属性值。

在 Visual Basic 中大部分属性在设计和运行时都可设置，但也有少部分属性只能在设计阶段或者运行阶段才能设置。

1.4.3 事件

对象的事件是指由系统预先设置好的，能够被对象所识别和响应的动作。比如 Click(单击)事件、KeyPress(键盘按下)事件及 Load(装入)事件等。

在这些事件中，有的事件是由用户操作引发的，如用户单击某一个窗体，将产生窗体的单击事件；有的事件是由系统的消息触发的，如某一个窗体装入时，它将自动产生该窗体的 Load 事件。

为了响应某个事件以完成某个功能，设计人员必须针对这个事件编制一段程序代码，这样的一段程序代码就称为事件过程。

在 Visual Basic 中系统会自动给出事件过程的结构，作为设计人员只需要将程序代码放入事件过程中。事件过程的一般格式如下：

```
Private Sub 对象名_事件名()  
    事件过程程序代码  
End Sub
```

其中，对象名指的是该对象的 Name(名称)属性，事件名是由 Visual Basic 预先定义好的赋给该对象的事件。

1.4.4 方法

对象的方法是对象要执行的动作。每种方法可使对象完成某个特定的功能。如窗体对象的 Print(打印)方法、Cls(清除)方法等。同一个对象有多种方法，不同的对象有不同的方法，并且其方法只能在编程中调用。它的调用格式为：

[<对象名>.]方法名 [(参数)]

如：Form1.Print "欢迎您！"。

其中，Form1 为窗体对象名，Print 就是 Form1 对象的方法名，此方法的功能是在窗体 Form1 中显示“欢迎您！”。

1.5 创建窗体

窗体像一块“画布”，它是所有控件的容器。在设计程序时，窗体是程序设计人员的工作平台；在程序运行阶段，窗体则是用户界面的窗口。窗体是 Visual Basic 中的一个重要对象，具有自己的属性、事件和方法。

1.5.1 窗体的属性

1. Name 属性

Name(名称)属性用来定义窗体的名称，每个窗体都必须有一个 Name 属性，以便在代码中用这个名称引用窗体，它与对象的标题(Caption)属性不一样。在工程中添加第一个窗体对象时，系统会给窗体 Name 属性的值自动赋一个默认名称：Form1。而以后在同一个工程中添加其他的窗体时，它的默认名称则在后面的数字逐次加 1 来表示，如 Form2、Form3…。为了提高程序的可维护性，便于识别窗体的功能，一般给窗体对象名设置一个有实际意义的名称，并建议以 frm 为前缀。需要用户注意的是：Name 属性是只读属性，即只能在属性窗口中设置，不能在程序代码中设置。

2. Caption 属性

Caption(标题)属性用来定义窗体的标题栏上的文字，默认值与默认的对象名相同，该属性既可在属性窗口中设置，也可以在程序代码中设置，在代码中设置该属性的语法格式如下：

```
[<窗体名称>.] Caption [=字符串]
```

如：Form1.Caption="登录用户"。

3. BackColor 和 ForeColor 属性

BackColor(背景颜色)属性用来设置窗体的背景颜色，而 ForeColor(前景颜色)属性则用来设置窗体的前景颜色。

每种颜色是用一个十六进制常量来表示的。在程序设计阶段，一般不用颜色常量来设置，而是通过调色板直观地设置。其操作方法为：在属性窗口中，用鼠标双击 BackColor 或 ForeColor 属性，从弹出的调色板中单击调色板中的某个色块，即可把这种颜色设置为窗体的背景色或前景色。

4. BorderStyle 属性

该属性用来确定窗体的边框样式，取值为 0~5 之间的整数，如表 1-2 所示。

表 1-2 Borderstyle 属性值的设置

属 性 值	功 能
0-None	无边框窗体，它不含控制菜单框、标题栏及窗体控制按钮
1-Fixedsingle	固定单边框窗体，运行时不能改变窗体的大小
2-Sizable	标准窗体形式，该窗体具有 Windows 标准窗体形式
3-FixedDialog	固定对话框窗体形式，窗体的大小不能改变
4-FixedToolWindow	固定工具窗体形式，只有一个关闭按钮，并且窗体的大小不能改变
5-SizableToolWindow	可变大小工具窗体形式，只有一个关闭按钮，并且窗体的大小能改变

BorderStyle 属性是只读属性，即只能在属性窗口设置，不能在程序运行期间通过代码设置。

5. Enabled 属性

该属性决定窗口对象是否有效，其默认值为 True。该属性的值为 True 时，表示允许用户操作窗口，并对操作做出响应；该属性的值为 False 时，表示禁止用户操作窗体，即窗体呈灰色。用代码设置该属性时，一般格式为：

```
[<对象名>.]Enabled [=Boolean]
```

6. Visible 属性

该属性决定对象是否可见，其默认值为 True。该属性的值为 True 时，程序运行时窗体可见；该属性的值为 False 时，程序运行时该窗体隐藏，即不可见。用代码设置该属性的一般格式为：

```
[<对象名>.]Visible [=Boolean]
```

7. Windowstate 属性

该属性决定窗体的当前状态是还原、最小化还是最大化。其属性取值如表 1-3 所示。用代码设置该属性时，一般格式为：

```
[<对象名>.] Windowstate [=设置值]
```

表 1-3 Windowstate 属性值的设置

属 性 值	功 能
0-Normal	默认值，运行时的大小与设计相同
1-Minmized	最小化状态，显示一个示意图标
2-Maxmized	最大化状态，窗体充满整个屏幕

8. MaxButton 和 MinButton 属性

这两个属性分别决定窗体标题栏上的最大化与最小化按钮是否可用，它们的系统默认值为 True。当其值为 True 时可用，其值为 False 时不可用。并且这两个属性只有在运行期间起作用。在设计阶段这两个属性的设置不起作用。

另外 `MaxButton` 和 `MinButton` 属性只能在属性窗口中设置，不能在程序运行期间用代码设置。

9. Left 和 Top 属性

`Left` 属性的值决定了窗体外框的左边缘与屏幕显示区左边缘之间的距离。`Top` 属性的值决定了窗体外框上边缘与屏幕显示区上边缘之间的距离。

这两个属性主要用来控制窗体的位置，其默认单位为 `twip`。用代码设置该属性的一般格式为：

```
[<对象名>.]Top [=数值]
[<对象名>.]Left [=数值]
```

10. Width 和 Height 属性

`Width` 属性的值决定窗体的宽度，`Height` 属性值决定窗体的高度，它们的默认单位为 `twip`。用代码设置它们的属性的格式如下：

```
[<对象名>.] Height [=数值]
[<对象名>.] Width [=数值]
```

11. Font 属性

该属性用来设置窗体上的字体的格式、大小及字体等。

12. Picture 属性

该属性用于在窗体中显示一个图形。窗体的 `Picture` 属性一般只在设计阶段通过属性窗口设置。其操作方法是在属性窗口选择该属性，单击其右端的【...】按钮，在弹出的【加载图片】对话框中选择一个图形文件，该图片即可显示在窗体上，该属性可以显示扩展名为 `bmp`、`wmf`、`gif`、`ico`、`jpg` 等多种格式的图形文件。

1.5.2 窗体的方法

1. Cls 方法

`Cls` 方法用来清除由 `Print` 方法在窗体或图片框中显示的文本或者使用绘图方法在窗体或图片框中显示的图形。其语法为：

```
[<窗体名>.]Cls
```

需要说明的是，在设计时在窗体中使用 `Picture` 属性设置的背景图形不受 `Cls` 方法的影响。

2. Print 方法

`Print` 方法用于在窗口中显示信息。其语法为：

```
[<窗体名>.]Print [<表达式列表>1, .....]
```

其中，表达式列表是一个或几个表达式，可以是数值表达式或字符串表达式。如果是数值表达式，则输出表达式的值；如果是字符串表达式，则原样输出；如果省略表达式列表，则输出一个空行。

3. Show 方法

Show 方法用于显示窗体。如果窗体被遮住，通过调用 Show 方法可以将其移到屏幕的顶端。其语法为：

```
[<窗体名>.]Show[Style]
```

其中，参数 style 为 0 时，窗体是无模式的；style 为 1 时，窗体是有模式的。当用 Show 方法显示无模式窗体时，在 Show 方法之后遇到的代码将要执行；而当用 Show 方法显示有模式窗体时，其后的代码要到该窗体被隐藏或卸载时才能被执行。

4. Hide 方法

Hide 方法用于隐藏窗体，其语法为：

```
[<窗体名>.]Hide
```

窗体隐藏后，窗体不可见，但未从内存中清除。

1.5.3 窗体的事件

事件的作用在于能够对用户的行为做出响应，与窗体有关的事件较多，在这里仅对最常用的几个事件进行介绍。

1. Click(单击)事件

程序运行后，当鼠标在窗口内的某一个位置单击时，触发该事件。

2. Load(装入)事件

Load 事件是在窗体被装载时，由系统触发。通常可以在 Load 事件过程中编写窗体启动代码，用来对窗体或窗体内的控件进行初始化。

3. Activate(活动)和 Deactivate(非活动)事件

当一个窗体变为活动窗体时触发 Activate 事件，而在一个窗体变为非活动窗口前触发 Deactivate 事件。

1.6 用 Visual Basic 开发应用程序

1.6.1 用 Visual Basic 开发应用程序的一般步骤

Visual Basic 的最大的特点就是在 Visual Basic 环境下，可以以最快的速度 and 效率开发具有良好用户界面的应用程序。用 Visual Basic 开发应用程序的一般步骤如下。

1. 创建一个新的工程

单击【文件】菜单中的【新建工程】命令后，将弹出一个新建工程对话框，单击该对话框中的【标准 EXE】图标，然后单击【确定】按钮，用户就建立了一个默认名称为“工

程 1”的工程，在此工程中会有一个默认名为 Form1 的窗体。

2. 在 Visual Basic 应用程序界面上添加控件

用 Visual Basic 开发环境编写应用程序，编程人员不需要编写大量的代码来描述用户界面的外观和位置，只需要根据应用需要在窗体上添加控件。在这个过程中，必须掌握添加选择控件的基本操作。

(1) 向窗体中添加控件

向窗体中添加控件的方法通常有两种。

方法 1:

- ① 双击工具箱中要使用的控件，该控件就会出现在窗体的正中央。
- ② 将控件移到适当位置并调整其大小。

方法 2:

- ① 用鼠标单击所需控件。
- ② 移动鼠标到窗体上适当的位置，此时光标变为十字形，然后拖动鼠标，即可以在窗体上画出一个控件。

(2) 对多个控件的选择

对多个控件进行操作前，必须选择需要操作的控件，选择控件通常有两种方法。

方法①：按住 **Shift** 键不松开，再用鼠标单击每一个要选择的控件。

方法②：用鼠标在窗体上拖出一个虚框，被框中的控件被选中。

此外，为了创建整齐美观的用户界面，往往需要窗体中同类控件的大小相等，间距相同，可以选择这些控件，然后使用【格式】菜单中的各种命令实现上述要求，注意所有的对齐都是以基准控件为对齐基准来进行的。

3. 设置窗体和控件的属性

建立界面以后，可以根据应用的需要对窗体及每个控件设置属性。

4. 编写事件驱动代码

事件驱动程序是 Visual Basic 应用程序的核心部分，编写事件驱动过程的步骤如下。

- (1) 在窗体设计界面中选中某个控件，然后双击，即进入代码窗口。
- (2) 在代码窗口中，有两个下拉列表框，左边的为对象列表框，右边的为过程列表框。
- (3) 用户根据需要在对象列表框中选择对象，在过程列表框中，选择该对象的事件。
- (4) 在代码编辑窗口的事件过程中，编写事件代码。

5. 程序的调试与运行

运行程序有 3 种方式。

- (1) 单击工具栏上的【启动】按钮，可运行程序。
- (2) 单击【运行】菜单中的【启动】命令。
- (3) 按 **F5** 键。

程序的逻辑比较复杂时，通常不能保证所编写的程序运行时不出现错误，而需要反复多次的调试。用户可以按 **F8** 键或使用【调试】菜单下的【逐语句】命令，进入程序的调试

模式，在该模式下，可以将代码单行运行，分析出现的问题，进而解决问题。

6. 保存

在窗体和代码设计好以后，为了防止出现死机等故障而造成的应用程序的丢失，需要及时对应用程序进行保存。

保存窗体文件及其他 Visual Basic 文件的步骤如下。

(1) 单击【文件】菜单中的【保存工程】命令，打开【文件另存为】对话框。

(2) 在【文件另存为】对话框中，指定保存文件的目录及窗体文件名，单击【保存】按钮。

(3) 最后在【工程另存为】对话框中，指定保存工程文件的目录及工程名，单击【保存】按钮。

1.6.2 编写一个简单的 Visual Basic 应用程序

下面将通过一个例子来说明如何在 Visual Basic 环境下设计一个应用程序。

【例 1.1】 设计一个 Visual Basic 应用程序，在用户界面上单击【显示】按钮后，窗体上显示“Visual Basic 欢迎您”，单击【退出】按钮时，结束整个应用程序，其步骤如下。

1. 新建工程

选择【文件】菜单的【新建工程】命令，在弹出的【新建工程】对话框中选中【标准 EXE】图标，然后单击【确定】按钮，建立一个默认名称为“工程 1”的工程，在此工程中有一个默认名为 Form1 的窗体。

2. 在窗体 Form1 中添加控件

单击工具箱中的命令按钮图标，鼠标指针变成“+”，在窗体中拖动鼠标指针至合适的位置，画出一个命令按钮 Command1，用同样的方法画出命令按钮 Command2。

再单击工具箱中的标签图标，鼠标指针变成“+”，在窗体中拖动鼠标指针至合适的位置，画出一个标签 Label1。此时界面如图 1.10 所示。



图 1.10 例 1.1 的设计界面

3. 设置控件的属性

对象建立好后, 可通过属性窗口为其设置属性, 各个对象属性值的设置如表 1-4 所示。

表 1-4 对象的属性设置

对 象	属 性	设 置 值
Command1	Caption	显示
Command2	Caption	退出
Label1	Caption	Visual Basic 欢迎您
	Visible	Flase
	Alignment	2-center

4. 编写事件代码

首先双击【显示】按钮, 自动出现代码编辑窗口, 在 Command1_Click() 过程中编写下列事件过程。

```
Private Sub Command1_click( )  
    Lable .Visible =True  
End sub
```

用同样的方法, 双击【退出】按钮, 在 Command2_Click() 过程中加入语句 End。

5. 运行程序

选择【运行】菜单下的“启动”命令, 即可运行。

在用户界面下, 单击【显示】按钮, 将显示“Visual Basic 欢迎您”, 单击【退出】按钮, 退出应用程序。

6. 保存工程

首先选择【文件】菜单中的【保存工程】命令, 在打开的【文件另存为】对话框中, 选择保存窗体文件的路径及窗体文件名 form1 以后, 单击【保存】按钮, 窗体文件就被保存了。

然后在弹出的【工程另存为】对话框中, 选择保存工程的路径及工程名, 单击【保存】按钮保存工程。

1.7 小 结

Visual Basic 是 Microsoft 公司 1991 年推出的, 它提供了开发 Microsoft Windows 应用程序的最迅速、最简捷的方法。它不但是专业人员得心应手的开发工具, 而且易于被非专业人员掌握使用。本章主要介绍了 Visual Basic 的主要特点和集成开发环境, 可视化编程的基本概念, 窗体的有关属性、事件和方法以及开发 Visual Basic 应用程序的一般步骤等内容。

Visual Basic 提供了一个集成开发环境，在这样一个工作平台上，用户可以完成应用程序的设计、编辑、编译及调试等工作。Visual Basic 6.0 的集成开发环境与 Windows 环境下的许多应用程序相似，同样有标题栏、菜单栏和工具栏等，除此之外，它还有工具箱、窗体设计器窗口、资源管理器窗口、属性窗口、窗体布局窗口和立即窗口等。

使用 Visual Basic 开发应用程序的一般步骤为：创建一个新的工程→在 Visual Basic 应用程序界面上添加控件→设置窗体和控件的属性→编写事件驱动代码→调试与运行程序→保存工程。

1.8 习 题

一、选择题

1. 在下列选择项中，()不是 Visual Basic 可能的状态。
A) 设计状态 B) 运行状态 C) 工程状态 D) 中断状态
2. 在程序设计中，通过()窗口可以设置窗体或控件的属性。
A) 窗体布局窗口 B) 属性窗口 C) 代码窗口 D) 窗体设计窗口
3. 窗体 Form1 的名称属性值为 Myform，它的 Load 事件过程名是()。
A) Form1_Load() B) Form_Load() C) Me_Load D) Myform_Load()
4. 与 C 程序设计语言相比，Visual Basic 最突出的特点是()。
A) 结构化程序设计 B) 程序开发环境
C) 事件驱动编程机制 D) 程序调试技术
5. 要改变一个窗体的标题内容，需重新设置的窗体属性是()。
A) Name B) Caption C) Borderstyle D) Enable
6. 启动 Visual Basic 时，在新建的 EXE 工程中，不会在工具箱中出现的控件是()。
A) 命令按钮 B) ActiveX 控件 C) 标签框 D) 文本框
7. 下列叙述中正确的是()。
A) 只有窗体才是 Visual Basic 的对象 B) 只有控件才是 Visual Basic 的对象
C) 窗体和控件都不是 Visual Basic 的对象 D) 窗体和控件都是 Visual Basic 的对象
8. “打开便携式电脑”，其中“电脑”、“便携式”、“打开”依次是下面的哪一项？
()
A) 对象、属性、方法 B) 属性、对象、方法
C) 属性、方法、对象 D) 对象、方法、属性

二、填空题

1. 工程文件的扩展名是_____，窗体文件的扩展名是_____，标准模块文件的扩展名是_____。
2. 可以通过_____菜单中的_____命令退出 Visual Basic。

3. 程序运行后,当单击窗体本身(不是窗体上的控件)时,将触发窗体的_____事件。
4. 在_____窗口中可以查看工程的文件结构。
5. 若要改变窗体运行时在窗口中的位置,可在_____窗口中设置。

三、编程题

编写一个应用程序。要求设置窗体的标题是“第一个 Visual Basic 程序”,程序执行时,单击窗体,在窗体中显示以下红色文字:

这是我编写的第一个 Visual Basic 应用程序。

第 2 章 Visual Basic 程序设计基础

教学提示：要使用 Visual Basic 编写程序，就必须掌握 Visual Basic 程序设计语言的语法规则。本章将主要介绍 Visual Basic 的基本数据类型、常量、变量、运算符和表达式，以及 Visual Basic 的常用内部函数，包括数学函数、字符串函数、转换函数、日期与时间函数等内容。

教学目标：掌握 Visual Basic 的基本数据类型、常量、变量、运算符和表达式以及 Visual Basic 的常用内部函数的使用方法。

2.1 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，为了合理地组织数据，Visual Basic 将数据分成不同的数据类型，每种数据类型占用的存储空间不同，能表示数值的范围也不同，用户在设计程序时，应该根据应用的需要，选择合适的数据类型描述数据并组织程序。Visual Basic 提供了系统定义的数据类型，即基本数据类型，同时也允许用户根据应用的需要定义自己的数据类型，即用户自定义数据类型，在这里仅介绍基本数据类型，它们主要是数值型、字符串型、逻辑型、日期型、可变类型和对象型等。

2.1.1 数值型数据

数值型数据可分为以下几类。

1. integer(整型)

整型数是不带小数点和指数符号的数，每个整型数据占用 2 个字节(16 位)的存储空间，其类型声明符为“%”，其取值范围为-32768~32768 之间的整数。

2. long(长整型)

长整型数也是不带小数点和指数符号的数，每个长整型的数据占用 4 个字节(32 位)的存储空间，其类型声明符为“&”，其取值范围为-2147483648~2147483647 之间的整数。

3. single(单精度浮点型)

单精度浮点型数是带小数的数值，通常以指数形式(科学记数法)来表示，以“E”或“e”表示指数部分。每个单精度浮点型数据占 4 个字节的存储空间，可以精确到 7 位，其正数的取值范围为 $1.401298E-45 \sim 3.402823E+38$ ，其负数的取值范围为 $-3.402823E+38 \sim -1.401298E-45$ ，其类型声明符为“!”。

4. double(双精度浮点型)

双精度浮点型数也是带有小数的数值，通常也以指数形式(科学记数法)表示，以“D”或“d”表示指数部分，双精度浮点型数占 8 个字节的存储空间，可以精确到 15~16 位，其正数的取值范围为 $4.94065645841247\text{E}-324 \sim 1.79769313486232\text{E}+308$ ，其负数的取值范围为 $-1.79769313486232\text{E}+308 \sim -4.94065645841247\text{E}-324$ ，其类型声明符为“#”。

5. currency(货币型)

货币类型是为计算货币而设置的数据类型，它也是一种固定小数点位置的数据类型，占用 8 个字节的存储空间，其小数点左边有 15 位数字，右边有 4 位数字，其取值范围为 $-922337203685477.5808 \sim 922337203685477.5807$ ，其类型声明符为“@”。

6. byte(字节型)

字节型数据用于存储二进制数据，占用 1 个字节的存储空间，其取值范围为 0~255。

2.1.2 字符串型数据

字符串类型数据(string)用来定义一个字符序列，由 ASCII 字符组成，包括标准的 ASCII 字符和扩展 ASCII 字符，其类型声明符为“\$”。在 Visual Basic 中，字符串必须放在双引号内，其中长度为空的字符串称为空串，例如“ABC”、“”(空串)。字符串型数据分为两种。一种为变长字符串，它能够包含字符的个数是可变的，随着对字符串变量赋予新值，它的长度随之发生变化，所占用的存储空间是该字符串长度再加上 10 个字节。另一种为定长字符串，即在程序运行中始终保持其长度不变的字符串，其中每个字符占用一个字节的存储空间，因此定长字符串所占用的存储空间就是该字符串的长度。例如，声明一个长度为 5 的定长字符串变量 A 的方法如下：

```
DIM A AS STRING*5
```

2.1.3 布尔型数据

布尔型数据(boolean)是一个逻辑值，它占用 2 个字节的存储空间，只能取两个值，即 true(真)和 false(假)。

2.1.4 日期型数据

日期型数据(date)用来表示日期，它占用 8 个字节的存储空间，可以表示的日期范围从公元 100 年 1 月 1 日到 9999 年 12 月 31 日，而表示的时间可以从 0:00:00 到 23:59:59。

表示日期的数据必须以“#”括起来，给日期类型变量赋值的基本形式为：

```
Date time = #5/12/2001 1:20 PM #
```

2.1.5 对象型数据

对象型数据(object)用来表示图形或 OLE 对象或其他对象，它占用 4 个字节的存储空间。

2.1.6 可变类型数据

可变类型数据(Variant)是一种通用的、可变的数据类型,除了定长字符串和自定义类型以外,它可以用来表示任何类型的数据。Visual Basic 可根据变量当前的内容,处理声明为可变类型变量和默认声明的变量,可变类型变量还可以在同一程序运行期间放置不同类型的数据,系统会自动完成必要的转换。

例如:

```
Dim MyVariant as Variant '如果不声明 My Variant 变量,下面的变量也是可变类型
My Variant="25"           '存入字符串“25”,变量 MyVariant 为字符串类型的数据
My Variant=My Variant-10  '在进行数值运算时,系统自动将 MyVariant 转换成数值类
                           型的数据,进行运算,并以数值型的数据存储
```

2.2 常量和变量

变量和常量是 Visual Basic 程序中最基本的数据元素,程序中的数据大多是以变量和常量的形式出现的。可以说常量和变量是程序中都存在的基本量。但是在不同的语言中,它们的表现形式是不同的。掌握变量和常量的表达方法是 Visual Basic 语言编程的基础。所谓常量是指在程序执行期间,其值是不能发生变化的数据;而变量的值是可以变化的,它代表计算机内存中指定的存储单元。

2.2.1 常量

常量是指在程序执行期间它的值不会发生变化的数据,在 Visual Basic 中常量分为两种,一种是文字常量,另一种则是符号常量。

1. 文字常量

文字常量分为数值常量、字符串常量、逻辑型常量和日期型常量等。

(1) 数值常量

数值常量又分为整型常量、长整型常量、浮点型常量和货币型常量等。

● 整型和长整型常量

在 Visual Basic 中可以使用十进制、八进制和十六进制来表示整型和长整型常量,十进制数值的表示方法与人们日常的表示方法相同,例如:235。如果在数字的开头加上前缀 &O(字母 O),则表示的是八进制数值,例如:&O315。如果在数字的开头加上前缀 &H,则表示的是十六进制数值,例如:&H418。

需要说明以下两点。

① 如果所表示的数值大小没有超过整型数据的范围(-32768~32768)时,系统会认为它是整型数,在内存中以 2 个字节存放它,如果所表示的数值大小超过了整型数据范围,则系统认为它是长整型数,以 4 个字节存放它。

② 用户还可以根据需求来设置某一个数据为长整型数,其方法是在某一个整型常量的后面加上类型声明符“&”,则该整型数被设置为长整型数,它们数值是相同的,但占用

内存不一样，即整型常量占用 2 个字节，而长整型数则占用 4 个字节。

- 浮点数常量

浮点数常量可分为单精度浮点数和双精度浮点数，浮点数由尾数、指数符号和指数 3 部分组成，用 $mE \pm n$ 或 $mD \pm n$ 来表示。其中 m 为尾数，它用实数表示，指数符号是 E (表示单精度) 或 D (表示双精度)， n 为指数，它必须是整数。

指数符号 E 或 D 的含义是 m 乘上 10 的幂次，例如：

`3.145E+3`

表示的数值分别为 3145。

- 货币型常量

货币型常量是指货币类型数据的常量表示形式。

(2) 字符串型常量

所谓字符串型常量是指用双引号括起来的一串字符(也可以为汉字)，如：

“Visual Basic” “ ” “ab56” “1234” 等。

(3) 逻辑型常量

逻辑型常量很简单，只有两个值，即 `True` 和 `False`。

(4) 日期型常量

日期型常量是指用 `#` 号括起来的一串日期，它可以表示日期、数据，其格式为：

`# mm-dd-yy #`

2. 符号常量

所谓符号常量是用一个被定义的标识符代替数值和字符串。定义符号常量的一般格式为：

`const 新常量名=表达式[, 新常量名=表达式]`

其中符号常量名必须符合 Visual Basic 标识符的定义方式，表达式可以由文字常量、算术表达式及逻辑运算符组成，也可以直接是字符串，但不能使用字符串连接运算符变量及用户定义的函数或内部函数。

例如：

```
const A1=3.14      '定义了一个单精度常量
const B1=2*A1      '定义了一个单精度常量
const C1="abcd"    '定义了一个字符串型常量
```

2.2.2 变量

几乎所有编程语言都要用到变量，变量的实质是指在程序运行过程中，其值可以改变的量。一个变量应该有一个名字，即变量名，在内存中占据一定的存储单元。在该存储单元中存放变量的值称为变量值。

为了通知计算机为变量留出所需要的空间，就需要对变量的类型进行说明，这一过程称为声明，除了对变量进行命名和类型说明以外，还必须规定该变量的使用范围。

在 Visual Basic 中，变量的有效范围是不同的，按照其使用范围可以将变量分为局部变

量、窗体级变量、过程级变量及全局变量，下面将分别介绍变量的命名规则及变量的声明等内容。

1. 变量的命名规则

在 Visual Basic 中，给变量命名时应遵循以下规则。

(1) 变量名只能由字母、数字和下划线组成。

(2) 变量名的第一个字符必须是英文字母。

(3) 变量名的有效长度为 255 个字符，而且名字的有效字符为前 40 个，后面的字符只为了增加易读性，对于区分不同的变量无效。

④ 不能使用 Visual Basic 保留字作为变量名，但可以把保留字嵌入变量名中，例如，不能将变量命名为 DIM(因为 DIM 是 Visual Basic 中的保留字)，但可以命名为 MYDIMS。

在 Visual Basic 中，变量名、过程名、符号常量名、记录类型名和元素名等都遵循上述规则。

在 Visual Basic 中，不区分变量名中字母的大小写，abc、ABC、aBc 指的是同一个名字，为了便于阅读，建议每个变量名开头的第一个字母用大写，例如，可以将某一个变量名定义为 Abc。

2. 变量的声明

在 Visual Basic 中，变量需要声明，声明的过程就是通知系统准备使用哪些变量，该变量需要多大的存储空间，以便系统将其存储到计算机的内存中。变量的声明有两种，即显式声明和隐式声明。

显式声明是用相应的语法声明变量，声明的变量具有所声明的数据类型；隐式声明是不经声明直接使用变量，变量自动具有 Variant 类型。

(1) 变量的显示声明

在变量的显示声明中又分为以下几种方式。

① 用 DIM 语句声明变量。

DIM 语句用于在模块、窗体的过程(子程序)中声明变量，其声明变量的格式为：

```
DIM 变量1 AS 数据类型[, 变量2 AS 数据类型] [, .....]
```

例如：

```
DIM MY2 as string  
DIM MY1 as integer, M3 as string
```

② 用 static 语句声明变量。

static 语句用于在过程中说明静态变量，其一般格式为：

```
static 变量名 as 类型名
```

例如：

```
static A as integer
```

用 static 语句声明的变量称为静态变量。静态变量的特点是第一次执行某个过程时，系统为变量分配了一个存储单元，当退出这个过程，存储单元的值将保留，下次再调用此过

程时，该变量的初值就是上次被保留的值。

例如：

```
private sub command1_dick( )  
    static M1 AS integer  
    print M1  
    M1 = M1 + 2  
End Sub
```

当用户单击 Command1 命令按钮时，该事件过程被调用，此时系统对 M1 赋予默认值 0，然后打印 M1 的值 0，接着执行语句 M1=M1+2，M1 值为 2。由于变量 M1 是静态变量，其值将保留下来，当再次单击 Command1 按钮时，该事件过程又被调用，变量 M1 的初始值不再为 0，而是上次保留的 M1 的值，即 M1 为 2。

③ 用 Public 语句声明变量。

用 Public 声明的变量为全局变量，该变量的作用范围是所有模块的所有过程，声明全局变量的一般格式为：

```
Public 变量名 AS 类型名
```

例如：

```
Public A AS integer
```

(2) 变量的隐性声明

变量的隐性声明方法有以下两种。

① 在 Visual Basic 的程序代码中不对变量进行声明而直接引用，则此变量默认的数据类型为 Variant。

② 可以省略声明语句，而在变量名的尾部加上变量类型说明符来使变量被隐性声明为某种类型的变量。例如：

```
A%=300
```

2.3 运算符与表达式

在程序设计语言中用不同的符号描述不同的运算形式，这些符号称为运算符，而运算的对象就称为操作数，由运算符将操作数接起来的合法的式子构成了表达式。例如：A+5、3*5*6 等都是表达式，单个变量式常量也可以看成是简单的表达式。

Visual Basic 提供了丰富的运算符和表达式。这些运算符和表达式包括算术运算符和算术表达式、字符串运算符和字符串表达式、关系运算符和关系表达式、逻辑运算符和逻辑表达式等。

2.3.1 算术运算符与算术表达式

1. 算术运算符

算术运算符是常用的运算符，用来执行简单的算术运算，Visual Basic 提供了 8 个算术运算符，表 2-1 按优先级列出了这些算术运算符。

表 2-1 Visual Basic 的算术运算符

运 算 符	含 义	优 先 级
^	乘方	1
-	取负	2
*	乘	3
/	除	3
\	整除	4
Mod	取模	5
+	加	6
-	减	6

说明:

(1) 指数运算

指数运算用来计算求方和方根,其运算符为“^”,例如:3^2表示为3的平方。当指数是一个表达式时,必须加上括号,例如:X的Y+Z次方,必须写成X^(Y+Z)。

(2) 浮点数除法与整数除法

浮点数除法运算符(/)执行标准除法操作,其结果为浮点数。例如:5/4的值为1.25。

整数除法运算符(\)执行整除运算,运算结果为整型数。整除的操作数一般是整型值,当操作数带有小数时,首先被四舍五入为整型数或长整型数,然后进行整除运算,其运行结果只截取整数部分,小数部分不做四舍五入处理。例如:8\3的结果为2,24.6\2.8的结果为8。

(3) 取模运算

取模运算符Mod用来求余数,其结果为第一操作数除以第二个操作数所得的余数,当操作数带有小数时,首先被四舍五入为整型数,然后求余数。例如:8 Mod 5的结果为3,24.6 Mod 2.8的结果为1。

2. 算术表达式

算术表达式又叫数值型表达式,它由算术运算符、数值型常量和变量、函数和圆括号组成,它的运算结果是一个数值。例如:

```
3
3+5.6
5+sin(x)
```

2.3.2 字符串运算符与字符串表达式

字符串运算符有两个,即“&”和“+”,其功能都是将两个字符串连接起来。但由于“+”运算符还有做加法的含义,容易造成混乱,所以做字符串连接运算时最好还是使用“&”运算符。例如:

```
A1="BCDE"
A2="FGHI"
A3=A1+A2      'A3 的值为"BCDEFGHI"
```

或

```
A3=A1&A2      'A3 的值也为"BCDEFGHI"
```

字符串表达式由字符串变量、常量、函数和运算符组成。

2.3.3 关系运算符与关系表达式

1. 关系运算符

关系运算符也称为比较运算符，用来对两个表达式的值进行比较，比较的结果是一个逻辑值，即真(True)或假(False)。

Visual Basic 把任何非 0 的值都看成是“真”，但一般以“-1”表示，0 则表示假，关系运算符既可以进行数值比较也可以进行字符串的比较。

Visual Basic 提供了 6 个关系运算符，如表 2-2 所示。

表 2-2 Visual Basic 的关系运算符

关系运算符	含 义	关系运算符	含 义
>	大于	<	小于
>=	大于等于	<=	小于等于
=	等于	<>	不等于

2. 关系表达式

关系表达式是由关系运算符将两个数值表达式或两个字符串表达式连接起来的式子。关系表达式的值是一个布尔类型的值，只有 True 或 False 两个取值。下面是合法的关系表达式。

```
5=3+2      表达式的值为真
5<2        表达式的值为假
```

2.3.4 逻辑运算符与逻辑表达式

1. 逻辑运算符

逻辑运算符用于对两个表达式进行逻辑运算，结果是逻辑值 True 或 False。

Visual Basic 提供的逻辑运算符有 6 种，如表 2-3 所示。

表 2-3 Visual Basic 的逻辑运算符

逻辑运算符	含 义	优 先 级	逻辑运算符	含 义	优 先 级
Not	逻辑非	1	Xor	逻辑异或	4
And	逻辑与	2	Eqv	逻辑等于	5
Or	逻辑或	3	Imp	逻辑蕴含	6

如果用 A 和 B 分别代表任意两个操作数，而用 T 代表逻辑真，用 F 代表逻辑假，则各种逻辑运算符的运算结果如表 2-4 所示。

表 2-4 各种逻辑运算符的运算结果

操作数 A	F	F	T	T
操作数 B	F	T	F	T
Not A	T	T	F	F
A And B	F	F	F	T
A Or B	F	T	T	T
A Xor B	F	T	T	F
A Eqv B	T	F	F	T
A Imp B	T	T	F	T

2. 逻辑表达式

用“逻辑运算符”连接两个或多个逻辑量组成的式子称为逻辑表达式，该表达式的返回值只有两种可能，即真(True)和假(False)。例如，下面都是合法的逻辑表达式：

Not (2>8) 其结果为 True
 (3>8)And(7<3= 其结果为 False

2.3.5 表达式的运算顺序

前面提到的所有类型以运算符都有运算优先级的的问题，在一个表达式中可以包含不同类型的运算符和函数，每一种运算都有执行的先后顺序，当不同类型的运算符出现在同一个表达式中时，计算机按一定的顺序对表达式求值，其一般顺序如下。

- (1) 首先进行函数运算，其优先级最高。
 - (2) 其次进行算术运算，其次序按算术运算符的优先级由高到低依次为
 $\wedge \rightarrow$ 取负 $\rightarrow *$, $/ \rightarrow \backslash \rightarrow \text{Mod} \rightarrow +$, $-$
 - (3) 最后进行逻辑运算，其次序按逻辑运算符的优先级由高到低依次为
 Not \rightarrow And \rightarrow Or \rightarrow Xor \rightarrow Eqv \rightarrow Imp
- 需要注意两点。

- (1) 在同一个表达式中所有同一级运算都按从左到右的顺序进行。
- (2) 括号可以改变运算的优先顺序，括号内的运算必须先执行。

例如，下列表达式中运算的优先顺序为：

7 / sin(z*x) > 3 or (x*3) ^ (-5) + 8 < 600
 ⑤ ② ① ⑦ ⑨ ① ④ ③ ⑥ ⑧

2.4 小 结

不同类型的数据有不同的操作方式和不同的取值范围。Visual Basic 提供了系统定义的基本数据类型，也允许用户根据应用的需要定义自己的数据类型。Visual Basic 基本数据类型主要有数值型、字符串型、逻辑型、日期型、可变类型及对象型等。

常量是指在程序执行期间它的值不会发生变化的一些数据，在 Visual Basic 中常量分为文字常量和符号常量两种。

变量是指在程序运行过程中，其值可以改变的量。一个变量应该有一个名字，称为变量名。变量在内存中占据一定的存储单元，在该存储单元中存放变量的值称为变量值。

运算符表示程序设计语言中对数据的最基本操作。被运算的对象称为运算量或操作数。表达式是程序设计语言中的基本语法单位，它表达一种求值规则，通常由常量、变量、函数、运算符及括号组成。Visual Basic 提供了丰富的运算符和表达式。这些运算符和表达式包括算术运算符和算术表达式、字符串运算符和字符串表达式、关系运算符和关系表达式、逻辑运算符和逻辑表达式等。

Visual Basic 为用户预定义了一批内部函数，它可分为数学函数、字符串函数、日期时间函数和随机函数等。

2.5 习 题

一、选择题

- 下列可作为 Visual Basic 变量名的是()。
A) VB123 B) 8P8 C) DIM D) X\Y
- 下列变量中，()是整型。
A) MY% B) MY# C) MY\$ D) MY!
- 下列()数据类型的变量不能存放负值。
A) long, Single B) Byte, Single C) Integer, Double D) Date, long
- 下列符号常量的声明中()是不合法的。
A) Const PI AS integer=100 B) Const PI AS Double=3.1415
C) Const PI AS Single=Log(2) D) Const X = "623"
- 设 a=3, b=10, 则下列表达式的值为真的是()。
A) A>=B AND b>10 B) (a>b) OR (b>0)
C) (a<0) EQV (b>0) D) (-3+5>a) AND (b>0)
- 设 a=2, b=3, c=4, d=5, 则表达式 NOT a<=c OR 4*c=b^2 AND b<>a+c 的值是()。
A) -1 B) 1 C) True D) False

7. 可以同时删除字符串两边空白字符的函数是()。

- A) Trim B) Ltrim C) Rtrim D) Space

8. 函数 $\text{Int}(\text{Rnd} * 99 + 1)$ 产生随机整数的范围是()。

- A) [1,99] B) [1,99] C) [1,100] D) [1,100]

二、填空题

1. 表达式 $(8 - (7 * 9 - 13) / 5 / 2)^2$ 的值是_____。

2. 在 Visual Basic 中, 字符串常量要用_____号括起来, 日期型常量要用_____括起来。

3. 执行下列语句后, 输出的结果是_____和_____。

```
as="AbcdefghI"  
Print Ucase(as)  
Print Mid (as, 2,4)
```

4. 表达式 $\text{Ucase}(\text{Mid} "abcd", 2, 3) + "123"$ 的值是_____。

第 3 章 Visual Basic 程序控制结构

教学提示：一般的计算机程序总是由若干条语句组成的。从执行方式上看，从第一条语句到最后一条语句完全按顺序执行，是简单的顺序结构；若在程序执行过程中，根据用户的输入或中间结果有选择地执行若干不同的任务则为选择结构；如果在程序的某处，需要根据某项条件重复地执行某项任务若干次，直到满足或不满足某个条件为止，这就构成了循环结构。顺序、选择和循环结构是程序流程控制的 3 个基本结构。本章主要讨论 Visual Basic 中组成程序流程控制的结构和相关语句。

教学要求：掌握顺序结构、选择结构和循环结构的相关控制语句的语法格式和使用方法，具有使用这 3 种基本结构进行程序设计的能力。

3.1 顺 序 结 构

1966 年，Bohra 和 Jacopin 提出了结构化算法的 3 种基本结构：顺序结构、选择结构和循环结构。目前已经得到证明，无论多么复杂的程序，都是由上述 3 种基本结构中的一种或多种的组合构成的。

顺序结构是按照程序段书写的顺序执行的语句结构。例如，有如图 3.1 所示的语句，则要先执行语句 A，然后才执行语句 B，两者是顺序执行的关系。



图 3.1 顺序结构

顺序结构是一种最基本的结构，表明事情发生的先后顺序。比如在日常生活中，总是先给水壶加水，然后才能烧水。在编写程序时，这种先后次序尤其重要。

3.1.1 赋值语句

赋值语句是程序设计中最常用的语句之一，在程序中，需要大量的变量存储程序中用到的数据，所以对变量进行赋值的赋值语句也会在程序中大量出现。赋值语句将表达式的值赋给变量或对象的属性。

赋值语句包括两种。一种是用来对一般的变量进行赋值的赋值语句，此语句用关键字

Let 描述；另一种是用来对对象型的变量进行赋值的赋值语句，用关键字 Set 描述。

赋值语句的语法格式为：

```
[Let] 变量或属性 = 表达式
Set 变量名 = 表达式
```

第一条语句中的 Let 是可选项，它在 Basic 语言的早期版本的赋值语句中被要求使用，在 Visual Basic 程序的赋值语句中通常被省略。表达式可以是任何数值型或字符串变量、常量或表达式。第二条语句的 Set 关键字是必需的。

在使用赋值语句时，需要注意数据类型的匹配问题。若将变量声明为数值型的，如整型、浮点型等，那么就不能将字符串表达式的值赋给该变量。同样，若将变量声明为字符串型的，那么就不能将数值型的值赋给该变量。比如，下面的语句就会产生错误：

```
Dim a As Integer
a = "Student"
```

当表达式的类型与变量的类型不一致时，强制转换成左边的精度。

```
Dim a as Integer, b as Double
a = 10 / 3      'a 中的结果为 3
b = 10 \ 3      'b 中的结果为 3.0
```

在进行变量赋值时，布尔型和日期型都被看作是数值型的。定义为 Variant 类型的变量不存在类型匹配的问题。例如，以下语句可以正常执行：

```
Dim a
a = "Hello!"
a = 100
```

3.1.2 数据输入和输出

1. 用输入框(InputBox)输入数据

InputBox 函数用于打开一个对话框，等待用户输入或选择一个按钮。若用户单击【确定】按钮或按下回车键，则返回用户在文本框中输入的内容。具体格式为：

```
InputBox (提示[, 标题][, 默认][, x 坐标位置][, y 坐标位置])
```

其中：

- “提示”为一个字符串表达式，作为对话框中的提示信息，不能省略。若要多行显示，则必须在每行行末加入回车(chr(10))和换行(chr(13))控制符。
- “标题”为一个字符串表达式，在对话框的标题区显示。若省略，则将应用程序名放入标题栏。
- “默认值”为一个字符串表达式，提供给用户的默认输入内容。
- “x 坐标位置”和“y 坐标位置”为整型表达式，它们确定对话框左上角在屏幕上的位置，屏幕左上角为原点，单位为缇(Twip)。

下面举例说明 InputBox 函数的使用。例如，以下语句让用户输入学生的成绩：

```
Str1=InputBox("请输入学生人数:", "学生成绩管理", "60")
```


此语句产生的对话框的提示信息为“请输入学生人数：”，标题栏为“学生成绩管理”，文本框内的默认值为“60”。执行此语句显示的对话框如图 3.2 所示。

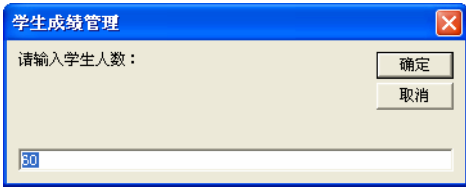


图 3.2 InputBox 函数示例

当用户输入信息后，若单击【确定】按钮，则输入信息将被赋给变量 Str1，若用户单击【取消】按钮，那么 Str1 的值为“”。无论用户输入什么类型的信息，该函数只能将它作为字符串返回，因此，如果用户想要取得其他类型的信息，要进行数据类型转换。

2. 用 Print 方法输出数据

Print 方法的语法格式为：

```
Object.Print [outputlist]
```

其中：

- Object 的取值可以为某个窗体的名字或 Debug。若为窗体的名字则在窗体上输出，否则在立即窗口中输出。
- Outputlist 参数表示要打印的表达式或表达式列表，若省略则打印一空白行。它具有以下语法格式：

```
{ Spc(n) | Tab(n) } Expression charpos
```

其 Outputlist 各部分的说明如表 3-1 所示。

表 3-1 Print 方法的参数说明

参 数	说 明
Spc(n)	可选项。在输出中插入空白字符，n 为要插入的空白字符个数
Tab(n)	可选项。将插入点定位在绝对列号上，n 表示列号。若无参数则将插入点定位在下一个打印区的起始位置
Expection	可选项。要输出的数值或字符串表达式
charpos	可选项。指定下一个字符的插入点，使用分号(;)将插入点定位在上一个被显示的字符之后。若省略，则在下一行打印下一字符

下面是一个使用 Print 语句的例子。

```
Private Sub Form_Click()  
Print "12345678901234567890"  
Print Tab(5); "5"; Tab(10 - 2); "8", " '注意这里 10-2>当前位置 5，故同行显示"  
Print Tab(5); "5"; Tab(10 - 6); "4"; Tab(20 - 15); "5"; "  
'注意这里 10-6<当前位置 5，故换行显示；20-15>当前位置 4，故同行显示"  
End Sub
```

程序执行结果如图 3.3 所示。

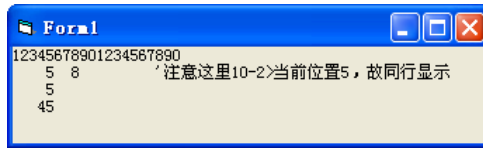


图 3.3 Print 语句示例

3. 用消息框(MsgBox)输出数据

Msgbox 函数用于在对话框中显示消息，并等待用户做出判断。它的语法格式为：

变量 = MsgBox(提示[, 按钮][, 标题])

其中：

- “提示”参数是向用户显示的信息。
- “按钮”参数为一个整型表达式，决定对话框中的按钮的数目和类型，以及对话框上的图标类型。按钮一般取值为“是”、“否”、“确定”、“取消”等。
- “标题”参数给出对话框的标题。

“按钮”和“标题”两个参数是可选的。Msgbox 函数还有一些其他参数，但它们不常用，在此不做介绍。例如，在用户关闭窗口之前，要询问用户的意见，显示询问对话框，可使用下面的语句：

```
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    'vbCance、vbQuestion 和 vbOKCancel 为系统常量
    Dim result as VbMsgBoxResult 'VbMsgBoxResult 为一数据类型
    result =MsgBox("确定要退出?", vbQuestion+vbOKCancel, "确认退出")
End Sub
```

程序执行结果如图 3.4 所示。

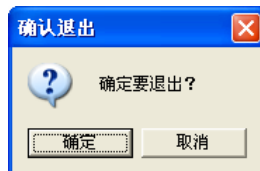


图 3.4 MsgBox 函数示例

3.1.3 注释、暂停与程序结束语句

1. 注释语句

注释是在程序中加入一些评注，往往起着提供编写程序的日期、编写人、解释程序代码的作用，其根本目的是为程序的阅读和修改提供信息，提高程序的可读性和可维护性。注释的方法有两种：使用 Rem 关键字或撇号 (')。二者的用法基本相同，在一行中撇号 (') 或 Rem 关键字后面的内容为注释内容。它们之间的区别在于使用 Rem 关键字时，必须使用冒号 (:) 将其与前面的语句隔开。例如：

```
Dim sum As Integer
sum = 1 : Rem 给 Sum 赋初值
Print sum
```

可以用一个撇号 (') 来代替 Rem 关键字。若使用撇号, 则所要注释的语句行不必加冒号。上例中的语句可以写为

```
sum = 1 ' 给 Sum 赋初值
```

2. 暂停(Stop)语句

Stop 语句用来暂停程序的执行, 是一种以编程方式设置断点的替代方法。当调试器遇到 Stop 语句时, 它将中断程序的执行(进入中断模式)。不同于 End 语句, Stop 语句不重置变量或返回设计模式。它可以从【调试】菜单中选择【继续】命令继续运行应用程序。其一般格式如下:

```
Stop
```

3. 结束(End)语句

End 语句用于结束一段程序的运行, 可以放在任何事件过程中, 其一般格式如下:

```
End
```

3.1.4 应用举例

顺序结构是最简单的一种结构, 不需要任何控制语句, 只需按照一定的次序编写即可。下面就通过一个例子说明顺序结构程序的设计方法。

【例 3.1】 由用户输入学生的数学、语文和英语成绩, 然后求出总成绩和平均成绩, 并显示在窗体上。

在编写程序之前, 首先要分析有多少元素要输入, 需要定义多少个变量, 然后才开始程序设计。

```
Private Sub Form_Click()
Dim a As Integer, b As Integer, c As Integer
Dim avg As Single, total As Single
a = InputBox("请输入数学成绩: ")
b = InputBox("请输入语文成绩: ")
c = InputBox("请输入英语成绩: ")
avg = (a + b + c) / 3
total = a + b + c
Print "总成绩为: "; total
Print "平均成绩为: "; avg
End Sub
```

3.2 选 择 结 构

在日常生活中, 经常会按照一定的条件做出相应的决定, 在程序中同样如此。选择结

构是结构化程序设计的基本结构之一，它是根据一定的条件来决定执行何种操作。

一般来说，选择是二支的，即条件非真即假，在执行选择结构时按照指定的条件进行判断，然后选择其中一组语句执行。比如，判断是否要上体育课，如果下雨则不上，否则照常上课。有些情况下，可供选择的结果可能多于两种，此时程序就有多种可能流向，这种结构称为多分支选择结构。比如根据用户输入的百分制成绩给出其等级评价，如果是 60 分以下则为不及格；若是 60~69 分，则为及格；若是 70~79 分，则为中等……。

3.2.1 单行结构条件语句

1. If...Then 语句

If...Then 语句的语法格式如下：

```
If <条件表达式> Then <语句>
```

其中，条件表达式通常为比较运算表达式，但它也可以是任何计算数值的表达式。如条件表达式是变量，则变量的值必须是 True(数值 0) 或 False(非零值)，例如：

```
If Flag=True Then
```

也可以省略“=True”，让 Visual Basic 自动检查变量 Visible 的值是 True 还是 False，例如：

```
If Flag Then
```

同理，语句 If Flag=False Then 也可以写成 If NOT Flag Then。

If...Then 语句的执行过程如图 3.5 所示。

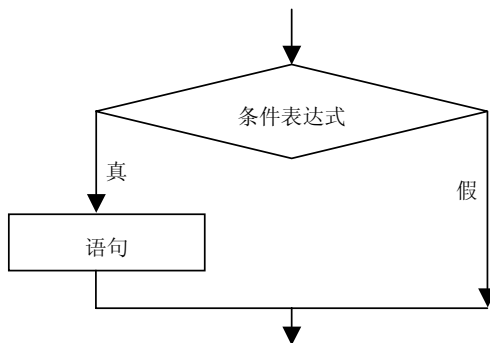


图 3.5 If...Then 语句的执行过程

需要注意的是，在 If...Then 结构中，Then 后面的语句必须与 If 和 Then 写在同一行上；若要使用该结构执行多条语句，这些语句必须书写在同一行上，并且用冒号分隔。下面举例说明 If...Then 语句的使用方法。

```
Sub Form_Click()  
    Dim Temp As Single  
    Temp = Val(InputBox("输入当前气温?"))  
    '如果气温低于 5 度，弹出消息对话框，提示气温过低  
    If Temp < 5 Then MsgBox "气温过低"  
End Sub
```

3.2.2 块结构条件语句

Visual Basic 提供的块结构条件语句主要有以下两种：

- If...Then...End If
- If...Then...Else

下面就来具体介绍它们的使用方法。

1. If...Then...End If 语句

If...Then...End If 语句的语法格式如下：

```
If <条件表达式> Then
    <程序段>
End If
```

该语句的执行过程是若条件表达式为 True(非零)，则执行 Then 与 End If 之间的程序段，否则执行 End If 后的下一条语句。其执行过程如图 3.6 所示。

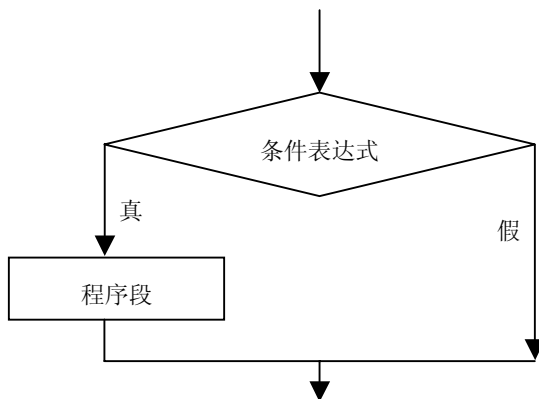


图 3.6 If...Then...End If 语句的执行过程

例如，已知两个数 a 和 b，找出其中较大的一个，使得 a 大于 b。具体实现如下：

```
If a<b Then
    t=a : a=b : b=t
End If
```

【例 3.2】火车站行李费的收费标准是 50kg 以内(包括 50kg)以 50kg 计,超过部分 0.50 元/kg。编写程序,要求根据输入的任意重量,计算出应付的行李费。

当重量在 50kg 以内时,收费是固定的,如果超出 50kg,则超出部分要根据重量收费。具体实现如下：

```
Private Sub Form_Click()
    Dim weight as single,pay as single
    weight= Val(InputBox("请输入行李的重量: "))
    pay = 50* 0.2
    If weight > 50 Then
        pay =(w - 50)* 0.5 + pay
    End If
End Sub
```

```

End If
Print "行李费一共为";pay
End Sub

```

2. If...Then...Else 语句

If...Then...Else 语句使得用户的程序具有判断的能力，其语法格式如下：

```

If <条件表达式> Then
    <程序段 1>
Else
    <程序段 2>
End If

```

该语句首先判断条件，若条件表达式为 True，则执行程序段 1；否则，则执行程序段 2。其执行过程如图 3.7 所示。

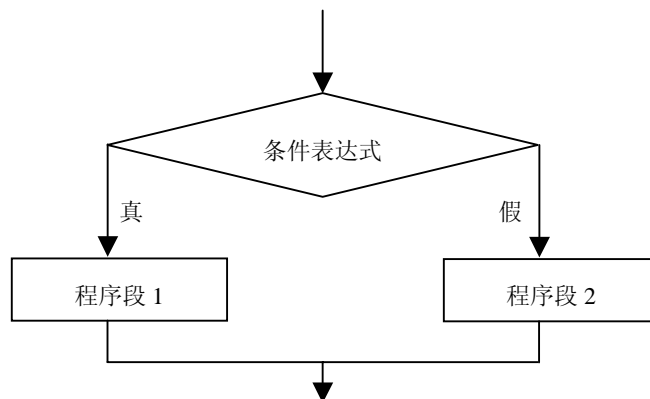


图 3.7 If...Then...Else 语句的执行过程

下面举例说明该语句的用法。

【例 3.3】 输入三角形 3 条边的长度，然后判断是否能够组成一个三角形，若可以则计算三角形的面积，并显示在窗体上；否则提示输入的长度不合格请重新输入。

在窗体的单击事件中添加如下代码：

```

Private Sub Form_Click()
Dim a As Integer, b As Integer, c As Integer
Dim s As Single
a = Val(InputBox("请输入三角形第一条边的长度: ", "求三角形的面积"))
b = Val(InputBox("请输入三角形第二条边的长度: ", "求三角形的面积"))
c = Val(InputBox("请输入三角形第三条边的长度: ", "求三角形的面积"))
If a + b < c Or b + c < a Or a + c < b Then
    MsgBox "输入的边长不能组成三角形!"
Else
    s = Sqr((a + b) ^ 2 + (b + c) ^ 2 + (a + c) ^ 2)
    Print "三角形的面积为: ", s
End If
End Sub

```

3.2.3 多分支选择语句

Visual Basic 提供的多分支选择语句有如下两种。

- If...Then...ElseIf
- Select Case

下面就来具体介绍它们的使用方法。

1. If...Then...ElseIf 语句

If...Then...ElseIf 语句的一般格式如下：

```
If <条件表达式 1> Then  
    <程序段 1>  
ElseIf <条件表达式 2> Then  
    <程序段 2>  
...  
ElseIf <条件表达式 n> Then  
    <程序段 n>  
[Else  
    默认程序段]  
End If
```

该语句首先判断条件表达式 1，若为 True，则执行程序段 1；否则检查条件表达式 2 是否为 True，如为 True，则执行程序段 2；若条件表达式 2 为 False，则判断条件表达式 3 是否为 True，如为 True，则执行程序段 3，依次类推……，直到条件表达式 n，若条件表达式 n 也为 False，这时若存在[Else 默认程序段]，则执行默认程序段，否则跳到 End If 的下一条语句执行(n>=2)。其执行过程如图 3.8 所示。

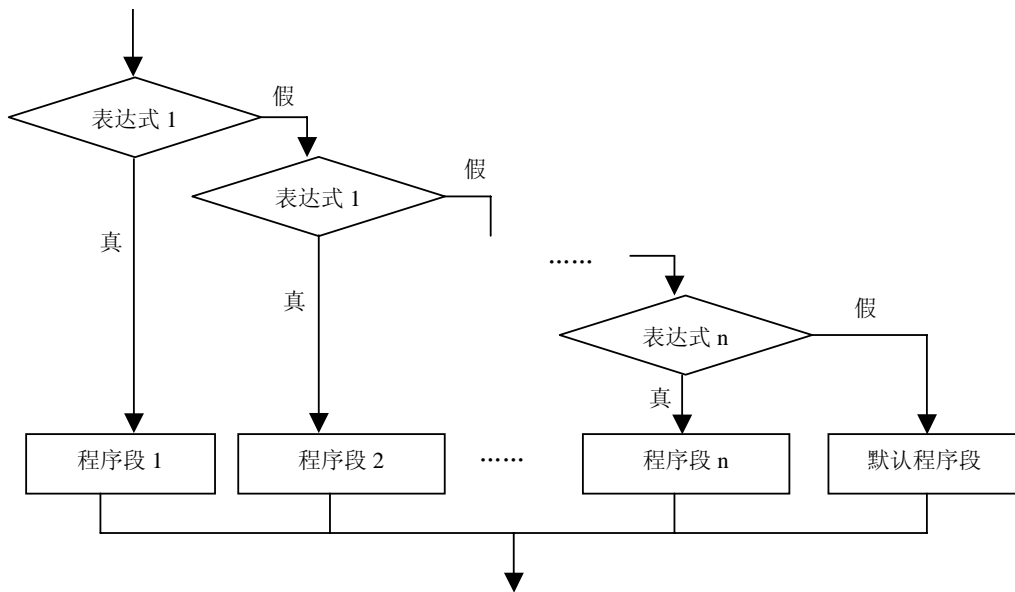


图 3.8 If...Then...ElseIf 语句的执行过程

要注意的是,不管有几个 ElseIf 子句,程序执行完一个语句块后,其余 ElseIf 子句不再执行。当多个 ElseIf 子句中的条件都成立时,只执行第一个条件成立的子句中的语句块。因此,在使用 ElseIf 语句时,要特别注意各判断条件的前后次序。

【例 3.4】 利用 If...Then...Else 语句编写求下列函数的值的程序。

$$y = \begin{cases} 1-x & x < 1 \\ (1-x)(2-x) & 1 \leq x \leq 2 \\ -(2-x) & x > 2 \end{cases}$$

根据题意,在窗体上添加一个命令按钮和标签控件。具体实现如下:

```
Private Sub Form_Click()  
    Dim x As Double, y As Double  
    x = Val(InputBox("请输入 x 的值: "))  
    If (x < 1) Then  
        y = 1 - x  
    ElseIf (x >= 1 And x <= 2) Then  
        y = (1 - x) * (2 - x)  
    Else  
        y = -(2 - x)  
    End If  
    Print "y="; y  
End Sub
```

2. Select Case 语句

多分支选择结构虽然可以用 If...Then...Else 来实现,但是判断的层次比较多时,会导致程序可读性差,不易维护。这种情况下,可以改用分情况选择语句 Select Case 语句。

Select Case 语句的语法格式如下:

```
Select Case <测试表达式>  
    Case 表达式列表 1  
        程序段 1  
    [Case 表达式列表 2  
        程序段 2]  
    ...  
    [Case Else  
        默认程序段]  
End Select
```

其中:

“测试表达式”可以是一个字符串或数学表达式。

“表达式列表”内容为常量,必须与测试表达式的类型相同,可以是以下形式之一。

(1) 一组枚举表达式,用逗号分隔,例如:

```
Case 2, 4, 6, 8
```


(2) 表达式 1 To 表达式 2, 较小的数值必须显示在 To 之前, 例如:

```
Case -7 To 7
Case "a" To "abc"
```

(3) Is <关系运算符> <常量>, 例如:

```
Case Is <5, Is>50
```

(4) 以上表达式的组合, 例如:

```
Case 2, 4, Is<-5
```

Select Case 语句首先处理一个测试表达式, 并只计算一次。然后将表达式的值与结构中的每个 Case 的值进行比较, 若相等, 就执行与该 Case 相关联的程序段, 若都不相等则执行 Case Else 后的默认程序段或执行 Select Case 语句的下一条语句(省略 Case Else 情况下)。Case Else 虽然是可选的, 但是最好使用, 以防遗漏测试数据。

下面给出了一个使用 Select Case 语句的实例, 读者可根据它加深对 Select Case 语句的使用的理解。

```
Private Sub Form_Click()
Dim Age As Integer
Age = Val(InputBox$("请输入年龄")) '输入数值
Select Case Age '根据不同的年龄范围输出不同的内容
    Case 1 To 12
        MsgBox "您是一个儿童"
    Case 13 To 19
        MsgBox "您是一个少年"
    Case Is > 19, Is<150
        MsgBox "您是一个成人"
    Case Else
        MsgBox "您的年龄超出了范围!!"
End Select
End Sub
```

3.2.4 应用举例

【例 3.5】 设计一个登录程序, 当用户输入的口令正确时, 显示“已成功进入本系统”, 否则, 显示“口令错! 请重新输入”。如果连续 3 次输入了口令仍不正确, 则提示“您无权使用本系统”。

分析: 假设使用一个文本框 Text1 来接收口令, 运行时用户输入完口令并按回车键后系统才对输入的口令进行检查, 因此本例使用了 Text1 的 KeyPress 事件。

当焦点位于文本框内, 按下键盘上任一键后会产生 KeyPress 事件, 同时返回按键代码 KeyAscii。回车键的代码为 13, 所以程序首先判断用户是否在 Text1 中按下了回车键, 若 KeyAscii=13, 表示口令输入完成。

说明：本程序中使用了一个窗体级变量 I 来统计输入错误口令的次数。变量 I 只在第一次判断口令时被初始化为 0，以后每次执行该过程时，若口令错误，则 I 的值累加 1，因此，当 I 的值为 3 时，表示用户已经连续 3 次输入了错误口令。程序代码如下：

```
Private Sub Text1_KeyPress(KeyAscii As Integer)
Static I As Integer
If KeyAscii = 13 Then
    If Text1.Text = "abc" Then
        Label2.Caption = "您已成功进入本系统"
    ElseIf I < 3 Then
        I = I + 1
        Label2.Caption = "口令错！请重新输入"
        Text1.SetFocus
    Else
        MsgBox "您无权使用本系统"
    End If
End If
End Sub
```

3.3 循环结构

循环是重复执行一组指令，重复次数由一定的条件决定，结构中反复执行的部分称为循环体。如果是无条件循环，循环体代码将无限地执行下去(即死循环)，这种情况当然应该避免。循环结构实现的方法基本上有两种，一种是指定一个条件表达式，当表达式的值为 True(或者是 False)时执行循环体，否则就退出循环；另一种是指定循环次数。

3.3.1 For...Next 循环结构

当可以预知程序中循环的次数时，可用 For...Next 语句。For...Next 循环中有一个计数器变量，决定循环的次数。For...Next 语句的语法格式如下：

```
For 循环控制变量 = 初值 To 终值 [Step 步长]
    [程序段]
    [If 条件式 Then Exit For]    '在特定条件下退出循环
    [程序段]
Next [循环控制变量]
```

其中：

- “循环控制变量”、“初值”、“终值”、“步长”是数值型变量。
- “步长”若为正数，则终值应该大于初值；若为负数，则终值应该小于初值。

循环开始时，循环控制变量的值为初值，每执行一次循环体，它的值要加一次步长的值，然后判断其值是否小于等于(或大于等于，在步长为负值的情况下)终值，如果判断结果为“真”，则继续执行循环体，否则退出循环。For...Next 语句的执行过程如图 3.9 所示。

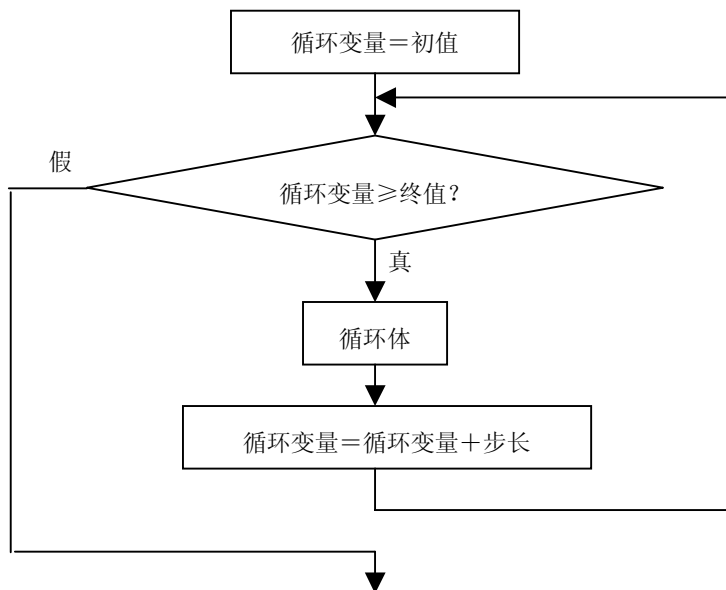


图 3.9 For...Next 语句的执行过程

例如，要输出 1~6 之间所有整数的平方值，可使用如下语句：

```

Private Sub Form_Click()
    Dim I, J, k As Integer
    I = 6: J = 1
    For k = I To J Step -1
        Print k^2
    Next k
End Sub
  
```

【例 3.6】 计算 $S=1+2+3+\cdots+n$ 。(n 值由用户输入)

这是一个“累加求和”的问题。解决这类程序时，通常是引入一个用来存放“和”值的单元 S，然后循环地向 S 中加入一个加数，最后，在变量 S 中存放的就是累加的结果。

程序代码如下：

```

n = InputBox("input n")
S = 0
For I = 1 To Val(n)
    S = S + I
Next I
Print S
  
```

在这个程序中就是循环地向 S 中加了 n 次数，第 1 次加的是 1，第 2 次加的是 2，……，第 n 次加的是 n，即每次的累加项总是与次数相同，因为循环控制变量 I 表示了次数，所以，在循环体中累加项用 I 来表示。为了使结果不受到影响，所以将 S 的初始值设置为 0。在“累加”问题中，关键在于设置了一个用来表示累加和的变量，然后通过循环依次地向其中放入一个有规律变化的累加项。除此以外，还有“累乘求积”的问题，累乘与累加

的区别除了运算不同之外，就是用来表示积的变量的初始值应设置为 1，因为 1 不影响运算的结果。在解决这类问题时，关键在于确定循环次数和找到每次要相加或相乘的项与循环变量之间的关系。有的累加或累乘问题中，每次的累加项或累乘项与循环变量之间不存在任何关系，这种情况下可以单独设置一个变量来表示这个项。

3.3.2 Do...Loop 循环结构

Do...Loop 语句是最常用且较有效的循环结构。它有两种形式。

1. 当型循环

当型循环是先判断条件，当条件满足时反复执行循环体，直到条件不满足为止。循环体有可能一次也不执行。其语法格式如下：

```
Do [ { While | Until } 条件表达式]
    [程序段]
    [If 条件式 Then Exit Do]    '在特定条件下退出循环
    [程序段]
Loop
```

其中，条件表达式是可选参数，可以是数值表达式或字符串表达式，其值为 True 或 False。若条件表达式的值是 Null，则会被当作 False。程序段包括一条或多条语句。当型循环的执行过程如图 3.10 所示。

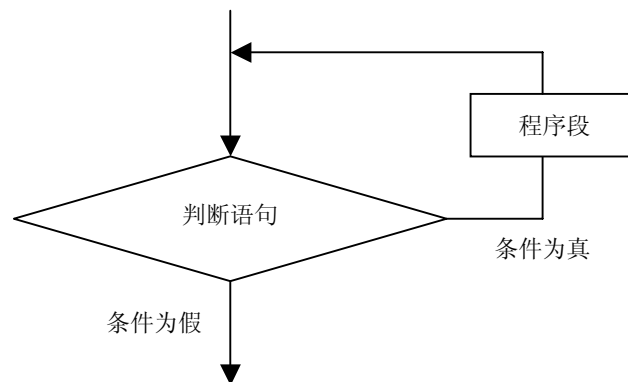


图 3.10 当型循环的执行过程

2. 直到型循环

直到型循环是先执行循环体，后判断条件，当条件满足时继续执行循环，否则退出。循环体至少会被执行一次。其语法格式如下：

```
Do
    [程序段]
    [If 条件式 Then Exit Do]    '在特定条件下退出循环
    [程序段]
Loop [ { While | Until } 条件表达式]
```

其中，条件表达式是可选参数。它可以是数值表达式或字符串表达式，其值为 True 或

False。如果条件表达式的值是 Null，则会被当作 False。程序段包括一条或多条语句。

在 Do...Loop 语句中可以在任何位置放置 Exit Do 语句，随时跳出 Do...Loop 循环。Exit Do 语句通常用于条件判断之后，在这种情况下，Exit Do 语句将控制权转移到紧接在 Loop 命令之后的语句。直到型循环的执行过程如图 3.11 所示。

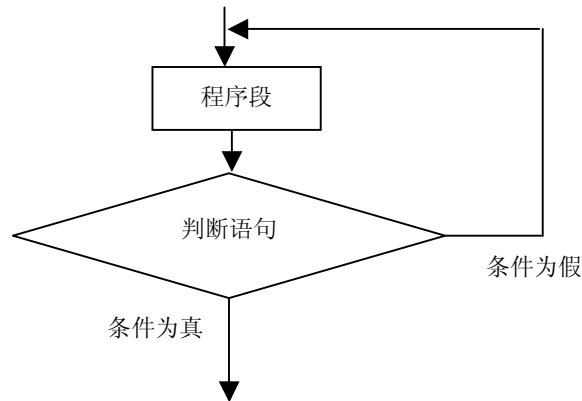


图 3.11 直到型循环的执行过程

下面将详细介绍 Do...Loop 语句的 4 种结构。

(1) Do While <条件表达式>

[程序段]

Loop

当条件表达式为 True 时，循环继续进行，直到条件表达式变为 False 为止。这种循环被称为“当型循环”。例如要计算 5 的阶乘，可编程如下：

```

Private Sub Form_Click()
    Dim I As Integer, Sum As Integer
    I = 1          'I 的初值赋为 1
    P = 1
    Do While I <= 5
        P = P * I
        I = I + 1
    Loop
    Print P
End Sub
  
```

(2) Do Until <条件表达式>

[程序段]

Loop

当条件表达式为 False 时，循环继续进行，直到条件表达式变为 True 为止。例如上例可以改写为：

```

Private Sub Form_Click()
    Dim I As Integer, Sum As Integer
    I = 1
    P = 1
    Do Until I > 5
  
```

```
P = P * I
I = I + 1
Loop
End Sub
```

程序输出的结果与上例相同。

(3) Do
 [程序段]
Loop While <条件表达式>

当条件表达式为 **True** 时，循环继续进行，直到条件表达式变为 **False** 为止。例如：

```
Private Sub Form_Click()
    Dim I As Integer
    I = 6
    Do
        Print I
        I = I + 1
    Loop While I <= 5
End Sub
```

程序输出结果为：6。

请读者分析一下，当本程序的循环体改为

```
Do While I <= 5
    Print I
    I = I + 1
Loop
```

此时，程序的输出结果是否有变化？为什么有变化？请找出原因。如果循环体不变，只改变 **I** 的初值，将 **I** 的初值赋为 1，结果会是什么？

(4) Do
 [程序段]
Loop Until <条件表达式>

当条件表达式为 **False** 时，循环继续进行，直到条件表达式变为 **True** 为止。例如：

```
Private Sub Form_Click()
    Dim I As Integer
    I = 6
    Do
        Print I
        I = I + 1
    Loop Until I > 5
End Sub
```

程序输出结果为：6。

【例 3.7】 编写程序，计算如下多项式，要求精度小于 10^{-12} 。

$$\frac{1}{1^2} + \frac{1}{2^2} + \cdots + \frac{1}{n^2} + \cdots$$

在窗体单击事件中添加如下代码：

```
Private Sub Form_Click()  
    Dim sum As Double, temp As Double  
    Dim i As Double  
    sum = 0  
    temp = 1  
    i = 1  
    Do While (temp >= 0.000000000001)  
        sum = sum + temp  
        i = i + 1  
        temp = 1 / i ^ 2  
    Loop  
    Print sum  
End Sub
```

3.3.3 While...Wend 循环结构

对于简单的循环可以用 **While...Wend** 语句，它的功能比其他的循环语句简单。其一般格式为：

```
While <条件表达式>  
    [程序段]  
Wend
```

该循环格式中的条件表达式一般为布尔表达式，也可以是数值和字符表达式，结果为 **True** 或 **False**，用来表示一个判断条件。

该语句首先计算给定的条件表达式的值，若结果为 **True**(非 0 值)，则执行循环体，当遇到 **Wend** 语句时，控制返回并继续对条件表达式进行测试，若仍然为 **True**，重复上述过程；若条件表达式的结果为 **False**，则直接执行 **Wend** 后面的语句。

这种循环结构的循环体内一般应该包括对“条件”进行改变的语句，使条件表达式的结果发生变化。否则，若初始条件成立，则每次执行完循环体后再检验条件，条件仍然成立，再执行循环体，这样无限执行下去，不能结束，就形成了“死循环”。若初始条件不成立，则循环体一次也不执行，循环就毫无意义。

While 循环与 **For** 循环的区别在于：**For** 循环对循环体执行指定的次数；而 **While** 循环则是在给定的条件为真时重复一组语句的执行。也就是说，通过 **While** 循环可以指定一个循环终止的条件，而使用 **For** 循环只能进行指定次数的重复。因此，当需要由数据的某个条件是否满足来控制循环时，使用 **While** 循环比较灵活。

【例 3.8】 找出 1000 以下的能够被 5 和 17 整除的最大的 10 个数并将它们输出。

分析：这里专门设置一个变量 **found** 记录当前找到的数的个数，只要 **found** 的值小于 10，说明还没有找足，循环继续进行。窗体事件过程代码如下：

```
Private Sub Form_Click()  
    a = 1000  
    found = 0  
    While (found <> 10)  
        a = a - 1
```

```
        If a Mod 5 = 0 And a Mod 17 = 0 Then
            found = found + 1
        Print a
    End If
Wend
End Sub
```

While...Wend 循环也可以是多层的嵌套结构。每个 Wend 匹配最近的 While 语句。与 While...Wend 语句相比, Do...Loop 语句提供了一种结构化与适应性更强的方法来执行循环。Do...Loop 语句中的条件表达式既可以放在循环之首,也可以放在循环之尾。此外, Do...Loop 循环可以在条件没有满足的情况下退出循环(当遇到 Exit Do 语句)。

3.3.4 循环的嵌套

一个循环体内又包含另一个完整的循环体结构,称为循环的嵌套。内嵌的循环中还可以嵌套循环,这就是多层循环的嵌套。在嵌套结构中,对嵌套的层数没有限制,有几层嵌套,就称为几重循环。多重循环的执行过程是:外循环每执行一次,内循环都要从头到尾执行一遍。同类循环和不同种类的循环都可以互相嵌套。嵌套的循环在程序设计中起着很重要的作用。例如,下面的程序段实现计算 $1!+2!+3!+4!$ 的结果并打印结果:

```
Dim I,J As Integer
Dim Sum1,Sum2 As Integer
Sum1=0:Sum2=1
For I=1 To 4
    For J=1 To I
        Sum2=Sum2*J
    Next J
    Sum1=Sum1+Sum2
Next I
Print Sum1
```

嵌套时需要注意,内层循环必须完全包含在外层循环之内,不能相互“交叉”,内层循环和外层循环应该使用不同的循环控制变量。另外,在多重循环的任何一层循环中都可以使用 Exit Do 或 Exit For 退出循环,但要注意只能退出 Exit Do 或 Exit For 语句所对应的最内层循环,而不是一次退出多层循环。

【例 3.9】 编程计算 $\text{Sum}=1+(1+2)+(1+2+3)+\cdots+(1+2+3+\cdots+n)$, 其中 n 由用户输入。

分析:这是一个累加问题,共有 n 项相加,可以设置存放累加和的变量为 Sum,而对于第 I 个累加项 $1+2+\cdots+I$, 又是一个累加问题,设存放该累加项的变量为 Sum1,因此可以用双重循环实现。用外循环对 I 依次取 1、2、 \cdots 、 n 值,对于每一次的累加项(如第 I 项),用内循环控制求 $1+2+\cdots+I$ 。设文本框 Text1 用于输入总项数 n , Text2 用于输出总和 Sum,运行时单击窗体计算结果。在窗体单击事件中添加如下代码:

```
Private Sub Form_Click()
    n = Val(Text1.Text)
    Sum = 0
    For I = 1 To n
```



```
Sum1 = 0           ' 在计算每个累加项之前，将存放累加和的变量清零
For J = 1 To I
    Sum1 = Sum1 + J
Next J
Sum = Sum + Sum1
Next I
Text2.Text = Sum
End Sub
```

3.3.5 应用举例

【例 3.10】 求任意两个整数 a、b 的最大公约数。

分析：所谓最大公约数是被 a、b 都能整除，并且其值不可能超过 a、b 中较小的一个的最大数。因此可以循环地依次将该范围内的每一个整数进行验证，直到发现一个能够同时被 a 和 b 整除的数为止。

新建工程，在窗体上添加一个命令按钮和两个文本框，编写程序如下：

```
Private Sub Command1_Click()
    Dim a As Integer, b As Integer, min As Integer
    a = Val(Text1.Text)
    b = Val(Text2.Text)
    If a > b Then
        b = min
    Else
        a = min
    End If
    For i = min To 1 Step -1
        If (a Mod i = 0) And (b Mod i) = 0 Then
            Print "最大公约数为", i
            Exit For           ' 得出解之后，循环停止
        End If
    Next i
End Sub
```

【例 3.11】 编程判断用户输入的数是否为素数。

分析：素数是指除了 1 和该数本身外，不能被任何整数整除的数。判断一个自然数 $n(n \geq 3)$ 是否为素数，只要依次用从 2 到 \sqrt{n} 作除数去除 n ，若 n 不能被其中任何一个数整除，则 n 为素数。

可以在 While 循环语句的循环体中编写语句，依次从 2 到 \sqrt{n} 作除数去试除 n ，并不断判断 n 是否被当前数整除。若 n 被某数整除，表明 n 不是素数，则退出循环；若直到循环结束，还没有一个数能整除 n ，则输出“ n 为素数”。

可以在程序中设置布尔变量 flag，flag = False 表示在程序执行过程中没有找到除了 1 和它本身以外能整除 n 的数，即 n 为素数；若 flag=True，则说明找到了某个能整除 n 的正整数，即 n 不是素数。

新建工程，编写代码如下：

```
Private Sub Form_Click()      '单击窗体时执行
    Dim n As Integer, km As Integer, flag As Boolean, i As Integer
    n = InputBox("请输入一个正整数(≥3)")
    k = Int(Sqr(n))
    flag = False
    i = 2
    While i <= k And flag = False
        If n Mod i = 0 Then
            flag = True
        Else
            i = i + 1
        End If
    Wend
    If flag = False Then
        Print n; "是素数。"
    Else
        Print n; "不是素数。"
    End If
End Sub
```

3.4 小 结

在程序设计语言中控制结构用于指明程序的执行流程。Visual Basic 语言提供的基本控制结构可以分为 3 种类型。

- (1) 顺序结构：按照先后顺序依次执行程序中的语句；
- (2) 选择结构：按照给定条件有选择地执行程序中的语句；
- (3) 循环结构：按照给定规则重复地执行程序中的语句。

顺序结构是 Visual Basic 程序中执行流程的默认结构。在一个没有选择和循环结构的程序中，语句将按照书写的先后顺序(从左到右，自上而下)被依次执行。但除了极其简单的情况之外，程序的顺序结构一般不能满足解决实际问题的需要。

本章讨论了顺序结构语句中的赋值、数据输入、输出、注释、暂停和结束语句等。

赋值语句的语法格式为：

[Let] 变量或属性 = 表达式

或

Set 变量名 = 表达式

Visual Basic 常用的键盘数据输入和输出使用 InputBox 函数和 MsgBox 函数。Print 方法可以作用于窗体，用于输出有关内容。

选择结构是 3 种基本结构之一。在大多数程序中都会包含选择结构。它的作用是，根据所指定的条件是否满足，决定从给定的两组操作中选择其一。Visual Basic 提供了 If...Then ...Else 语句和 Select Case 语句实现选择结构。

If...Then ...Else 语句包括 3 种形式：

- (1) 行 If...Then ...Else 语句，其语法格式如下：

```
If <条件表达式> Then <语句> [Else <语句> ]
```

(2) 块 If…Then…Else 语句，其语法格式如下：

```
If <条件表达式> Then
    <程序段 1>
Else
    <程序段 2>
End If
```

(3) If…Then…ElseIf 多分支选择语句，其一般格式如下：

```
If <条件表达式 1> Then
    <程序段 1>
ElseIf <条件表达式 2> Then
    <程序段 2>
...
ElseIf <条件表达式 n> Then
    <程序段 n>
[Else
    默认程序段]
End If
```

Select Case 语句的一般格式为：

```
Select Case <测试表达式>
    Case 表达式列表 1
        程序段 1
    [Case 表达式列表 2
        程序段 2]
    ...
    [Case Else
        默认程序段]
End Select
```

循环控制结构是程序中的另一个基本结构。在实际问题中，常常需要进行大量的重复处理，循环结构可以使程序设计者只写很少的语句，而让计算机反复执行，从而完成大量类似的计算。Visual Basic 提供了 For 语句、Do … Loop 语句和 While 语句实现循环结构。

For 语句的一般格式为：

```
For 循环控制变量 = 初值 To 终值 [Step 步长]
    [程序段]
    [If 条件式 Then Exit For] '在特定条件下退出循环
    [程序段]
Next [循环控制变量]
```

Do…Loop 语句提供了当型循环和直到型循环两种描述方式，分别为：

```
Do[ { While | Until } 条件表达式]
    [程序段]
    [If 条件式 Then Exit Do] '在特定条件下退出循环
    [程序段]
Loop
```

和

```
Do
    [程序段]
    [If 条件式 Then Exit Do]    '在特定条件下退出循环
    [程序段]
Loop [ { While | Until } 条件表达式]
```

while 语句的一般格式为:

```
While <条件表达式>
    [程序段]
Wend
```

3.5 习 题

一、选择题

1. 执行下面的程序段后, 变量 S 的值为()。

```
S=5
For i =3.5 To 8.7 Step 0.5
    S=S+1
Next i
```

A) 14 B) 15 C) 16 D) 17

2. 下面的循环体的执行次数为()。

```
For I=30 to -10 Step -4
    Print I
Next I
```

A) 7 B) 9 C) 11 D) 12

3. 在窗体上添加一个命令按钮, 然后编写如下事件过程:

```
Private Sub Command1_Click()
    x = 5: a = 0: b = 0
    Do
        a = a+b-x
        a = Val(InputBox("请输入 a 的值"))
        b = Val(InputBox("请输入 b 的值"))
        x = Val(InputBox("请输入 x 的值"))
    Loop Until x=-1
    Print a
End Sub
```

程序运行后, 单击命令按钮, 依次在输入对话框中输入 4、3、2、1、0、-1, 则输出结果为()。

A) 0 B) 1 C) -1 D) 2

4. 阅读下面的程序段:

```
a=0
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

5. 在窗体上添加一个命令按钮(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
  Print n; x
End Sub
```

程序运行后, 单击命令按钮, n 和 x 的值分别为()。

- A) 1 和 0 B) 2 和 72 C) 3 和 50 D) 4 和 168

二、填空题

1. 以下程序的功能是: 生成 10 个 50~100 之间的随机整数, 输出其中能被 3 整除的数并求出它们的和。请填空。

```
Private Sub Command1_Click()
  For i=1 To 10
    x=Int(_____ *200+_____)
    If _____=0 Then
      Print x
      S=S+__
    End If
  Next i
  Print "Sum=";S
End Sub
```

2. 由用户输入 3 个数, 找出其中的最大值和最小值。

```
Private Sub Form_Click()
  Dim a, b, c, T
  a = _____
  b = _____
  c = _____
  If a < b Then _____
  If _____ Then T = a: a = c: c = T
```

```
If b < c Then T = b: b = c: c = T
Print "最大值为:"; _____
Print "最小值为:"; _____
End Sub
```

三、编程题

1. 由用户输入一个整数，编程判断该数是奇数还是偶数。
2. 找出 100 以内的被 5 整除余 3、被 4 整除余 2 的数并输出到窗体上。
3. 找出 1000 以内所有的水仙花数并输出。水仙花数即该数的个位的立方加上十位的立方加上百位的立方等于这个数本身。例如： $153=1^3+5^3+3^3$ 。
4. 编程输出九九乘法表。

第 4 章 Visual Basic 常用内部控件

教学提示：控件是组成程序外部表象的基本元素，通过合理整齐地布置控件，可以方便地使用程序，使程序的易用性大大提高。本章主要介绍常用控件的属性、事件和方法。

教学要求：掌握命令按钮、标签框、文本框、复选框、单选按钮、框架、图像框、图片框、滚动条以及定时器等控件的基本要素和使用方法。

4.1 概 述

在 Visual Basic 中，控件分两类：一类是标准控件，又叫内部控件，也就是 Visual Basic 预设的控件，这类控件是在 Visual Basic 启动时工具箱内的控件；另一类是外挂控件，必须通过 Visual Basic 的【工程】菜单中的【添加用户控件】来增加控件，又称为 ActiveX 控件。本章中主要介绍的是常用的 12 个标准控件，如表 4-1 所示。

表 4-1 Visual Basic 常用内部控件

控件名称	图 标	类 名	说 明
命令按钮		CommandButton	命令按钮，可以和多个按钮组合使用
标签		Label	为用户显示不可交互操作或不可修改的文本
文本框		TextBox	提供用户交互区或显示文本
复选框		CheckBox	与多个复选框组成的多项选择中，允许选择多个选项
单选按钮		OptionButton	与多个单选按钮组成的多项选择中，只允许做一种选择
框架		Frame	为控件提供可视的功能化容器
图像框		Image	显示位图、图标、Windows 图元文件、JPEG 或 GIF 文件
图片框		PictureBox	显示位图、图标、Windows 图元文件、JPEG 或 GIF 等文件。也可显示文本或者作为其他控件的可视容器
列表框		ListBox	显示项目列表，用户可以从中选择
组合框		ComboBox	文本框和列表框的组合，即可以输入又可以选择
滚动条		HscrollBar,VscrollBar	用户通过滚动条选择指定区间的数据
定时器		Timer	按一定的时间间隔执行定时器事件

在窗体上创建了一个控件后，控件会自动获得一个名字，新对象的唯一名字由对象类型加上一个唯一的整数组成。例如，第一个文本框控件的 Name 属性是 Text1，第二个文本框名字就是 Text2。但实际程序的编写中，为了操作方便、提高程序的可读性，可以考虑根据控件在程序中的实际作用，为其另取一个合适的名称。建议使用：英文缩写+数字编码作为控件的名称。

控件的命名规则如下。

- (1) 对象的 Name 属性必须以一个字母开始。
- (2) 最长可达 40 个字符。
- (3) 它可以包括数字和带下划线的字符，但不能包括标点符号或空格。

Name 属性是每个控件都有的，具有唯一性，不同类型的控件不能共享相同的名字。每个控件除了 Name 属性外，其余属性都因它的作用而有所区别，并由于功能上的特殊性而包含一些自身特殊或常用的属性、事件和方法。

4.2 命令按钮

命令按钮控件是使用最为广泛的控件之一。它可以控制一个程序的开始、中断和结束。命令按钮控件的名称(Name)和 Caption(标题)的默认值都为：Command1、Command2、…。

4.2.1 命令按钮的常用属性

1. Caption 属性(字符串类型)

控件的 Caption 属性值为显示在控件上的标题，运行时用户在界面上看到的是 Caption 值。并不是所有的控件都有这个属性，比如文本框、驱动器或目录或文件列表框、图像或图片框、定时器、滚动条、组合框、数据库等控件就没有这个属性，初始状态 Caption 与名称属性同名。

2. Enabled 属性(逻辑类型)

Enabled(可用性)属性只能取值为 False 或 True：当值为 False 时控件不可用，为灰色；当值为 True 时控件可用。

Enabled 属性可以在设计时设置，也可以在运行时用赋值语句为其赋值。

3. 命令按钮的其他属性

(1) Appearance 属性(整数 0、1)

值为 1 则以 3D 效果显示该控件，值为 0 则不然，该属性运行时只读。

(2) BackColor 属性(整数类型)

背景色的设置，可以在调色板中选择，或者采用系统的调色方案。

(3) Cancel 属性(逻辑类型)

该属性值决定按钮是否为一个取消按钮，当按 Esc 键时，Cancel 属性为 True 的命令按钮(只有一个)的 Click 事件过程被调用。

(4) Default 属性(逻辑类型)

该属性值决定哪一个命令按钮控件是窗体的默认命令按钮，即不论焦点处于任何控件上(非命令按钮)，在按下回车键时，都会调用默认命令按钮的 Click 事件。窗体中只能有一个命令按钮可以为默认命令按钮，当某个命令按钮的 Default 设置为 True 时，窗体中其他的命令按钮自动设置为 False。

(5) Font 属性

- 属性名 FontBold(逻辑类型)，值为 True 则显示文字字体加粗。
- 属性名 FontItalic(逻辑类型)，值为 True 则显示文字为斜体。
- 属性名 FontName(字符串类型)，值为控件上所显示字体类型的名称。
- 属性名 FontSize(整数类型)，值为控件上所显示的文字字号的大小。
- 属性名 FontStrikethru(逻辑类型)，值为 True，则控件上所显示的文字被加删除线，值为 False 则无删除线。
- 属性名 FontUnderline(逻辑类型)，值为 True，则控件上所显示的文字下加下划线，值为 False 则无下划线。

(6) Left、Top、Height、Width 属性(数值类型)

控件的位置属性，以缇(twip)为单位，1 英寸=1440 缇。

- Left 属性值为控件的左边界与它所在容器左边界之间的距离。
- Top 属性值为控件的上边界与它所在容器上边界之间的距离。
- Height 属性值为控件的高度。
- Width 属性值为控件的宽度。

修改以上 4 个属性可以设定控件的位置和大小。

(7) Style、Picture 属性

Style 属性值决定按钮的显示方式是否以图像形式出现：值为 0 则按钮上只能显示文字；值为 1 则在按钮上显示图像：图像文件的名由 Picture 属性值确定。

(8) Visible 属性(逻辑类型)

判断控件是否为可见，值为 True 和 False，当为 False 时，按钮不可见。

(9) Index 属性(整数类型)

当控件为一控件数组时，此属性值为该控件在数组中的下标值。

4.2.2 命令按钮的常用事件

命令按钮标题上所显示的按钮功能是由该控件相应的事件过程赋予的。相应的动作激发相应的事件过程，对于命令按钮最常用的事件就是 Click 单击事件。命令按钮没有双击事件，它解释为两次连续的单击事件。

【例 4.1】 设置一个用于卸载当前窗体的命令按钮。

将控件名称改为 cmdExit，把 Caption 属性设为“退出”，其 Click 事件过程如下：

```
Private Sub cmdExit_Click()  
    Unload Me          ' 卸载当前窗体，此处 Me 是指当前窗体  
End Sub
```

运行时单击窗体中显示的【退出】按钮，则激活了它的 Click 事件，执行 Unload Me 语句使得窗体被卸载。

4.3 文本控件

与文本相关的标准控件是标签和文本框。标签只能用来显示文本，而文本框可以在用户之间进行交互，既可以显示文本，又可以输入文本。

4.3.1 标签

标签控件默认的控件名称和 **Caption** 属性值都为：**Label1**、**Label2**、…。**Label** 控件是图形控件，可以显示用户不能直接改变的文本。

标签的用途非常广，几乎在所有的设计中都会用它来做一些说明。运行时不能在标签框上直接输入文字，但可以用程序中的语句来改变 **Label** 控件显示的文本。

标签有如下几个常用属性。

(1) **Caption** 属性(字符串类型)

标签控件的主要作用是在应用程序界面上加入说明，用户在界面上看到的是它的 **Caption** 属性，所以 **Caption** 属性是标签最重要的属性。它的许多其他属性，比如字体、颜色等，也是为标题的字体而设置的。

由于运行时，**Caption** 属性的值可以被改写，因此也时常通过改写该属性值而在界面上显示运行结果。

(2) **AutoSize** 属性(逻辑类型)

该属性决定控件是否自动改变大小，以显示其全部内容。当字符串 **Caption** 的字符数超过设定的字符串 **Caption** 的宽度时，有以下两种情况。

- 若 **AutoSize** 属性值为 **True**，则自动改变控件大小以显示全部内容；
- 若 **AutoSize** 属性值为 **False**(默认值)，则保持控件大小不变，超出部分不予显示。

(3) **WordWrap** 属性 (逻辑类型)

该属性用来指示 **AutoSize** 属性设置为 **True** 时，标签控件是否要进行水平或垂直展开以适合其 **Caption** 属性中指定的文本的要求。

(4) **Backstyle** 属性(整数类型，取值 0，1)

该属性值用于指示标签是否透明。

- **Backstyle** 属性值为 0，透明(与窗体同色)；
- **Backstyle** 属性值为 1(默认值)，不透明。

(5) **Bordstyle** 属性(整数类型，取值 0，1)

该属性值用于判断控件是否有边框。

- **Bordstyle** 属性值为 0(默认值)，无边框；
- **Bordstyle** 属性值为 1，有边框。

标签控件也会用到如 **Click**、**DbClick** 等事件。但在程序设计中，习惯上还是将其作为文本显示使用。

4.3.2 文本框

文本框控件名称和 Text 属性的默认值都为：Text1、Text2、…。

TextBox 控件有时也称作编辑字段或编辑控件，它可以在程序运行过程中接收数据(字符型)，也可以用来显示程序的运行结果。

1. 文本框的常用属性

文本框的常用属性有如下几个。

(1) Text 属性(字符串类型)

该属性是文本框控件最重要的属性之一。可以在设计时设定 Text 属性，也可以在运行时用直接在文本框内输入或向 Text 属性赋值的方法来改变该属性的值。

给文本框控件的 Text 属性赋值的语法为：

文本框控件名.Text = <字符串>

(2) MaxLength 属性(整数类型，取值在 0~65535 之间)

该属性值设定在文本框控件中能够输入的最大字符数，默认值为 0，表示在文本框中输入的字符数没有限制。

该属性值不得大于 65535，若在其取值范围内设定了一个非 0 值，则 Text 尾部多出的部分被截断。

(3) MultiLine 属性(逻辑类型)

该属性值设定 Text 字符串中是否接收换行符。

- MultiLine 属性值为 False(默认值)，文本框中的字符只能在一行中显示。
- MultiLine 属性值为 True，则文本框中的字符可以多行显示。设计时，在属性窗口中直接写入 Text，按回车键则换行；运行时，用赋值语句修改 Text 属性，必须加入回车、换行符才可换行。

例如：Text1.Text = "未到达边界" + Chr(13) + Chr(10) + "另起一行。".

(4) ScrollBars 属性(整数类型，取值 0、1、2、3)

该属性值决定是否为文本框加滚动条。当文本过长，可能超过文本输入/显示边界时，应为该控件添加滚动条。

- ScrollBars 属性值为 0(默认值)，无滚动条；
- ScrollBars 属性值为 1，加水平滚动条；
- ScrollBars 属性值为 2，加垂直滚动条 (当 MultiLine 属性为 True 时该设置有效)；
- ScrollBars 属性值为 3，同时加水平和垂直滚动条。

(5) PasswordChar 属性(字符类型)

该属性值设定输入文本的特殊显示字符，在设计密码程序时非常有效。其值只能为 1 个字符，默认值为空。

仅当 MultiLine 属性为 False 且 PasswordChar 值为非空格符时，该属性设置有效。无论用户按下了什么字符，在输入框中一律是用 PasswordChar 值显示(非实际字符)。

如在 Form_Load 事件中做如下设置，窗体装入后直接在 Text1 的输入框中输入 "1234567"，可以看到输入框内显示的内容是 "*****".

```
Private Sub Form_Load()  
    Text1.PasswordChar = "*"
End Sub
```

(6) 文本编辑属性

SelStart 属性(数值类型)表示文本框内被选中文本的起始位置, 计数从 0 开始。

SelLenght 属性(数值类型)表示文本框内被选中文本的长度。

SelText 属性(字符串类型)是文本框内被选中的文本。

在 Windows 操作系统中, 剪贴板是常用的文本编辑帮助工具。把文本框和剪贴板对象结合使用, 可以方便地实现文本的复制、剪切和粘贴。

有关剪贴板操作的几个方法, 简单介绍如下:

● Clear 方法

清除剪贴板中的内容, 应用举例:

```
Clipboard.Clear
```

● GetText 方法

将剪贴板中的文本复制到指定文本框的光标处或复制给字符串变量。应用举例:

```
Text1.SelText=Clipboard.GetText  
str1=Clipboard.GetText
```

● SetText 方法

将选中文本送入剪贴板, 应用举例:

```
Clipboard.SetText(Text1.SelText)
```

【例 4.2】 设计一个简单的文本编辑器程序, 文本输入在某个文本框控件 Text1 中, 程序具有复制、剪切、粘贴、删除、清除和退出功能。该程序的界面如图 4.1 所示。界面中有关控件的设置如表 4-2 所示。



图 4.1 例 4.2 的界面设计

表 4-2 例 4.2 的控件设计

控件名称	控件显示	控件功能
Command1	复制	复制选中的文字到剪贴板
Command2	剪切	剪切选中的文字到剪贴板
Command3	粘贴	粘贴文字到光标所在处
Command4	删除	删除选中的文字
Command5	清除	清除文本框中所有的内容
Command6	退出	退出程序
Text1		显示文本

程序代码如下：

```
Private Sub Command1_Click()
    Clipboard.Clear '清空剪贴板
    Clipboard.SetText (Text1.SelText) '将 Text1.Text 内被选中的文本送剪贴板
End Sub
Private Sub Command2_Click()
    Clipboard.Clear '清空剪贴板
    Clipboard.SetText (Text1.SelText) '将 Text1 控件选中的文本送剪贴板
    Text1.SelText = "" '删除 Text1 控件选中的文本
End Sub
Private Sub Command3_Click()
    Text1.SelText = Clipboard.GetText '将剪贴板中的文本复制到指定文本框的光标处
End Sub
Private Sub Command4_Click()
    Text1.SelText = "" '将 Text1.Text 内被选中的文本置为空串
End Sub
Private Sub Command5_Click()
    Text1.Text = "" '将 Text1.Text 置为空串
End Sub
Private Sub Command6_Click()
    End
End Sub
```

2. 文本框的常用事件

文本框的常用事件有如下两个。

(1) Change 事件

当文本框中的内容发生改变时触发 Change 事件。

(2) Keypress 事件

Keypress 事件由用户在文本框控件的输入框中按任意键触发。Keypress 事件过程有一个形参变量 KeyAscii，调用该过程时，触发调用的按键的 ASCII 码值被赋与 KeyAscii。

文本框输入完后，一般习惯于按回车键表示输入结束(回车键的 ASCII 码为 13)。下列事件过程可以在输入过程结束后，在对话框中显示“good student!”。

```
Private Sub Text1_KeyPress(KeyAscii As Integer)
    If KeyAscii = 13 Then
        MsgBox("good student!")
    End If
End Sub
```

【例 4.3】 编写一个密码验证程序，用户只允许输入 3 次密码，如果错误，则退出程序。

在窗体中放置两个标签 Lable1 和 Lable2，Lable1 的 Caption 属性设置为“密码输入框”，Lable2 的 Caption 属性设置为空，再放置一个文本框 Text1，其 Text 属性设置为空，用于输入密码。

程序代码如下：

```
Dim password As String
Dim flag As Byte
Private Sub Form_Load()
    ' 初始量的设置, 文本框的属性也可以放在属性栏中
    ' 直接设置
    Text1.PasswordChar = "*"
    ' 设置文本框为密码框, 密码显示为*
    Text1.MaxLength = 6
    ' 设置密码最大长度为 6 位, 超出部分被截除
    password = "123456"
    ' 设置初始密码为 123456
    Label2.FontSize = 15
    Label2.Alignment = 2
End Sub
Private Sub Text1_KeyPress(KeyAscii As Integer)
    If KeyAscii = 13 Then
        ' 按下回车键后判断密码
        flag = flag + 1
        If Text1.Text = password Then
            ' 密码正确则进入后续程序, 在这里示例直接退出
            Label2.Caption = ""
            MsgBox ("密码正确!")
            End
        Else
            If flag = 1 Then
                Label2.Caption = "第一次密码错误"
            ElseIf flag = 2 Then
                Label2.Caption = "第二次密码错误"
            ElseIf flag = 3 Then
                Label2.Caption = ""
                flag = 0
                MsgBox ("您无权进入, 密码错误!")
            End
        End If
    End If
End Sub
```

在这里利用了一个变量 Flag 来计算输入的次数, 每次回车后如果系统判断输入的不是正确的密码, 则 Flag 加 1, 允许输入 2 次, 当 Flag 为 3 时, 输入被拒绝, 程序退出。

4.4 单选按钮、复选框和框架

单选按钮是为了在多个选项中选择唯一一个合适的值, 而复选框则是在多个选项中选择多个结果, 功能上虽然不同, 但在使用时二者的很多属性和事件都很相似。

4.4.1 单选按钮

单选按钮在计算机应用程序中使用非常普遍, 对于具有互斥性的答案, 一般都采用单选按钮作为选择, 比如性别的选择, 男或者女。

单选按钮的属性以及名称的默认值都为 Option1、Option2、…。

1. 单选按钮的常用属性

单选按钮的常用属性有如下一些。

(1) Caption 属性(字符类型)

Caption 属性值是显示在控件上的文本，是单选按钮的标题。

(2) Alignment 属性(取值为整数 0、1)

Alignment 属性决定单选按钮的标题(Caption 属性值)在控件上的位置：

属性值为 0 时表示左对齐(Left Justify)，即单选按钮的标题在右边，此为默认方式。如图 4.2 所示，控件 Option1 的标题为 Option1。

属性值为 1 时表示右对齐(Right justify)，即单选按钮的标题在左边，如图 4.2 所示，控件 Option2 的标题为 Option2。

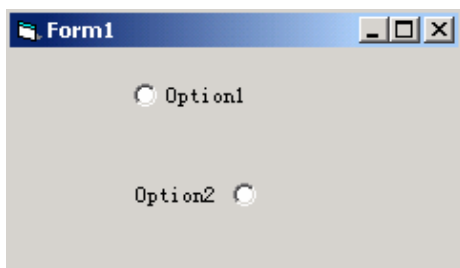


图 4.2 单选按钮对齐属性说明

(3) Enabled 属性(逻辑类型)

控件的 Enabled 属性值为 False 时，控件显示为灰色，运行时不可用。就是说，当该控件的事件发生时，相应的事件过程一个也不会被调用。

(4) Index 属性(整数类型)

Index 属性值标识由单选按钮组成的控件数组中某个按钮的索引值，利用控件数组的方式使用单选按钮，可以简化代码，增强代码的可读性。具体在下章数组中有详细说明。

(5) TabIndex 属性(整数类型)

建立控件时，Visual Basic 自动为其分配一个 TabIndex 值(Menu、Timer、Data、Image、Line 和 Shape 等控件除外，这些控件不包括在 Tab 键顺序中)，利用 Tab 键可以在控件之间切换焦点。

(6) Value 属性(逻辑类型)

反映控件状态的属性，返回 True 表示已选择了该按钮；False(默认值)表示没有选择该按钮，使用这个属性可以判断哪个按钮被选中。

2. 单选按钮的常用事件

和命令按钮一样，单选按钮最常用的事件也是 Click 事件和 GotFocus(取得焦点)事件。

【例 4.4】 利用单选按钮改变标签控件上的字体。程序运行时，显示如图 4.3 所示的界面。

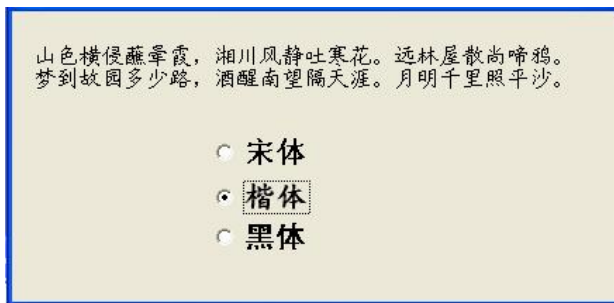


图 4.3 例 4.4 的结果图

程序代码如下：

```
Private Sub Form_Load()  
    Label1.AutoSize = True  
    Label1.FontSize = 13  
    Label1.Caption = "山色横侵蘸晕霞，湘川风静吐寒花。远林屋散尚啼鸦。" + Chr(10) + Chr(13) + "梦到故园多少路，酒醒南望隔天涯。月明千里照平沙。"  
End Sub  
Private Sub Option1_Click()  
    Label1.FontName = "宋体"  
End Sub  
Private Sub Option2_Click()  
    Label1.FontName = "楷体_gb2312"  
End Sub  
Private Sub Option3_Click()  
    Label1.FontName = "黑体"  
End Sub
```

字体的设置要注意，一个是双引号的问题，程序设计要在半角状态下；另外楷体的字体名称是"楷体_gb2312"，不要与别的字体类型混淆起来。

4.4.2 复选框

复选控件和单选控件选择结果的方式不同，是否被选中可以由它的属性 `Value` 的值进行判断。一些常用的属性的复选框和单选框基本相同，这里就不再赘述。

复选框控件的 `Caption` 属性以及名称的默认值都为 `Check1`、`Check2`、…。

1. 复选框的常用属性

复选框的常用属性有如下一些。

(1) `Index` 属性(整数类型)

该属性值为复选框控件数组的下标，一般来说，使用控件数组时这是不可少的属性，通过它可以区分同一组复选框。

(2) `Value` 属性(整数 0、1、2)

复选标志，这是复选框最重要的属性，它的值与复选框控件的状态有关，其默认值为 0。

`Value` 属性值为 0，则复选框内为空白；`Value` 属性值为 1，则复选框内显示一个“√”标志；`Value` 属性值为 2，则复选框内为灰色的“√”标志。

运行时单击复选框，如果原先 Value 属性值为 0(同时框内显示空白)，单击后 Value 属性值变为 1(同时框内显示“√”标志)。如果原先 Value 属性值为 1 或 2(同时框内显示黑白或灰色的“√”标志)，单击后 Value 属性值变为 0(同时框内显示空白)。

反复单击同一复选框控件时，其 Value 属性值只能在 0、1 之间交替变换。

2. 复选框的常用事件

复选框控件的常用事件一般为 Click 事件，复选框不支持鼠标双击事件，系统把一次双击解释为两次单击事件。

【例 4.5】 利用复选框设置标签框显示文字的字型变化：加下划线、加粗、斜体。程序运行时，显示如图 4.4 所示的界面。



图 4.4 例 4.5 的结果图

程序代码如下：

```
Private Sub Check1_Click()  
    If Check1.Value = 1 Then  
        Label1.FontBold = True  
    Else  
        If Check1.Value = 0 Then Label1.FontBold = False  
    End If  
End Sub  
Private Sub Check2_Click()  
    If Check2.Value = 1 Then  
        Label1.FontUnderline = True  
    Else  
        If Check2.Value = 0 Then Label1.FontUnderline = False  
    End If  
End Sub  
Private Sub Check3_Click()  
    If Check3.Value = 1 Then  
        Label1.FontItalic = True  
    Else  
        If Check3.Value = 0 Then Label1.FontItalic = False  
    End If  
End Sub
```

当复选框的 Value 属性为 1 时，复选框被选中，产生各种字型变化的效果。而当它为 0 时，字型恢复到默认的状态。由图 4.4 可见，字型变化的结果是可以相互叠加的。

4.4.3 框架

框架(Frame)是一种容器控件,在容器中的控件,不仅可以随容器移动,而且控件的位置属性也是以相对于容器内的位置设置的。窗体、图片框都是容器。

Frame 控件不仅可以作为其他控件的容器,而且可用 Frame 控件将其他控件分为可标识的控件组,比如单选按钮可以通过 Frame 分组,从而产生多个选择。

框架控件的 Caption 属性以及名称的默认值都为 Frame1、Frame2、…。

框架控件必须先建立,然后在框架中添加其他的控件,不能简单地把已建立的控件拖到框架中去。

【例 4.6】 利用框架建立一个设置字体、字型、字号的对话框。

字型可以使用复选框,但字体和字号只能单选,必须分组才能消除选项之间的互斥性。利用框架可以将两个功能分开。

在窗体中放置 3 个框架,其中 Frame1 为字型设置,里面放置 3 个复选框 Check1(粗体)、Check2(下划线)和 Check3(斜体); Frame2 为字号设置,里面放置 3 个单选按钮 Option1(10)、Option2(20)和 Option3(30); Frame3 为字体设置,里面放置 3 个单选按钮 Option4(黑体)、Option5(楷体)和 Option6(宋体)。程序运行时,显示如图 4.5 所示的界面。



图 4.5 例 4.6 的设计及显示结果图

程序代码如下:

```
Private Sub Form_Load()
    Text1.Text = "Visual Basic"
End Sub
Private Sub Check1_Click()
    If Check1.Value = 1 Then
        Text1.FontBold = True
    Else
        If Check1.Value = 0 Then Text1.FontBold = False
    End If
End Sub
Private Sub Check2_Click()
    If Check2.Value = 1 Then
        Text1.FontUnderline = True
    Else
        If Check2.Value = 0 Then Text1.FontUnderline = False
    End If
End Sub
Private Sub Check3_Click()
    If Check3.Value = 1 Then
        Text1.FontItalic = True
    Else
        If Check3.Value = 0 Then Text1.FontItalic = False
    End If
End Sub
Private Sub Option1_Click()
    If Option1.Value = 1 Then
        Text1.FontSize = 10
    End If
End Sub
Private Sub Option2_Click()
    If Option2.Value = 1 Then
        Text1.FontSize = 20
    End If
End Sub
Private Sub Option3_Click()
    If Option3.Value = 1 Then
        Text1.FontSize = 30
    End If
End Sub
Private Sub Option4_Click()
    If Option4.Value = 1 Then
        Text1.FontName = "SimHei"
    End If
End Sub
Private Sub Option5_Click()
    If Option5.Value = 1 Then
        Text1.FontName = "KaiTi"
    End If
End Sub
Private Sub Option6_Click()
    If Option6.Value = 1 Then
        Text1.FontName = "Songti"
    End If
End Sub
```

```
End If
End Sub
Private Sub Check3_Click()
If Check3.Value = 1 Then
Text1.FontItalic = True
Else
If Check3.Value = 0 Then Text1.FontItalic = False
End If
End Sub
Private Sub Option1_Click()
Text1.FontSize = 10
End Sub
Private Sub Option2_Click()
Text1.FontSize = 20
End Sub
Private Sub Option3_Click()
Text1.FontSize = 30
End Sub
Private Sub Option4_Click()
Text1.FontName = "黑体"
End Sub
Private Sub Option5_Click()
Text1.FontName = "楷体_gb2312"
End Sub
Private Sub Option6_Click()
Text1.FontName = "宋体"
End Sub
```

4.5 列表框和组合框

列表框和组合框都可以为用户提供选项列表，用户可以在控件列表项中进行选择，所以在事件和属性方法上有很多相似之处。但二者又有所不同，使用时要根据需要进行选择。

4.5.1 列表框

列表框控件的名称的默认值为 List1、List2、…。列表框没有 Caption 属性。

列表框的作用是提供选项进行选择，如果选项太多，超出列表框设计时的长度，则 Visual Basic 会自动给列表框加上垂直滚动条，为了能正确操作，列表框的高度不应少于 3 行。它不具备编辑功能，程序运行后用户不能脱离代码而改变列表框项目内容，只能选择项目。

1. 列表框的常用属性

列表框的常用属性有如下一些。

(1) List 属性(字符串数组)

列表框控件的各个表项是使用数组的方式保存的，数组的每一个元素存储列表框控件的一个表项。因此，利用索引可以访问列表项目。

需要注意的是,数组元素最小下标值的设定对于列表框来说是无效的,如设定“Option Base 1”,但对于列表框来说,第一个表项的索引值总是 0。

(2) ListCount 属性(正整数)

该属性值为控件列表部分项目的个数,由于索引值从 0 开始计数,所以 ListCount-1 是最后一个项目的 Index。因此,列表框 List 属性下标值的范围为 0~ListCount-1。

(3) ListIndex 属性(整数类型)

该属性值为被选中表项的索引,通过 ListIndex 属性,可以区分已经选中和未被选中的表项。

表达式“List1.List(List1.ListIndex)”的值应解析为:List1.ListIndex 是选中项的索引数值,而 list(List1.Listindex)是列表框所对应的选中项。

(4) MultiSelect 属性(整数 0、1、2)

利用列表框控件的该属性,可以为列表框设置“单选”或“允许多选”属性。

- 值为 0: 只能单选(默认值),若选中一个表项则其他表项取消突出显示。
- 值为 1: 可以多选,被选中的表项都被突出显示。
- 值为 2: 扩展多选,可以用鼠标在列表栏内拖动、选中相邻的若干个表项。

(5) Text 属性(最后一次选中的表项,字符串类型)

该属性用来返回当前选中的表项内容。

对于单选的列表框控件 List1,字符串 List1.list(List1.ListIndex)与 List1.Text 相等,都是被选中表项的文本。

(6) Selected 属性(逻辑类型)

Selected 属性标识一个数组,数组各元素为:Selected(0)、Selected(1)、…、Selected(列表框控件名.ListCount-1),若列表框控件的第 i 个表项被选中,则 Selected($i-1$)的值为 True(若为 Selected($i-1$)赋值 True,则列表框控件的第 i 个表项被突出显示)。

在允许多项选择的情况,应利用 Selected 属性区分哪些表项被选中。

下列语句可以在窗体上显示列表框控件 List1 被选中的表项:

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

(7) SelCount 属性(整数类型)

该属性值为被选中表项的个数。

(8) Sorted 属性(逻辑类型)

确定列表框控件的各表项是否按字母数字升序排列:设为 True 时按序排列,设为 False 时不按序排列。该属性的默认值为 False。

(9) Style 属性(整数 0、1)

值为 0(默认值),为标准样式,如图 4.6 左边的列表框所示。

值为 1,为复选框样式,如图 4.6 右边的列表框所示。

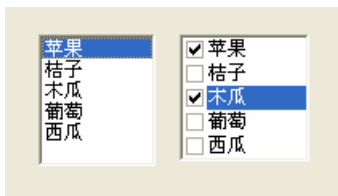


图 4.6 列表框控件 Style 示例

如图 4.6 所示，若列表框控件的 Style 属性值为 1，无论 MultiSelect 属性取何值，该列表框在实际使用上允许多选。

2. 列表框的常用方法

列表框的常用方法有如下几个。

(1) AddItem 方法

AddItem 方法用于将项目添加到 ListBox 控件中，其一般格式为：

列表框控件名.AddItem 表项文本[, 索引号]

其中，索引号可以指定项目文本的插入位置，如果省略索引号，表项文本自动加到列表框末尾。

如下列 Form_Load 事件可在装入窗体时，为列表框控件 List1 添加若干表项。

```
Private Sub Form_Load()
    List1.AddItem "苹果"
    List1.AddItem "桔子"
    List1.AddItem "香蕉"
    List1.AddItem "木瓜"
    List1.AddItem "火龙果"
    List1.AddItem "榴莲"
End Sub
```

列表框控件的表项也可以在设计时添加，其方法为：在属性窗口内选中 List 属性，在下拉框中添加表项，用 Ctrl+Enter 组合键换行。运行时也可以用赋值语句在列表框控件中添加或修改表项：

列表框控件名.List(List1.ListIndex) = 表项文本

(2) Clear 方法

Clear 方法用于清空列表框控件中所有表项，其一般格式为：

列表框控件名.Clear

(3) RemoveItem 方法

RemoveItem 方法用于删除列表框中指定的表项，其一般格式为：

列表框控件名.RemoveItem 索引值

语句 List1.RemoveItem List1.ListIndex 的功能即为删除所选项。

3. 列表框的常用事件

列表框的常用事件有以下两个。

(1) Click 单击事件

运行时单击列表框控件的某一表项，可以使该表项在未选状态和选中状态之间切换，同时触发该列表框控件的 Click 事件。

(2) DblClick 双击事件

双击直接运行表项所对应的事件。

【例 4.7】 本程序运行时，界面如图 4.7 所示。左右两边列表框的单个选项可以互相换位，也可以一次把所有的选项切换到另一侧。双击左侧列表框也可以完成右移，双击右侧列表框也可以完成左移。

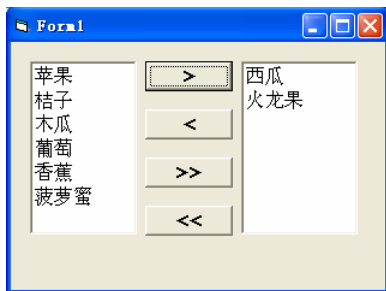


图 4.7 例 4.7 的界面设计

程序代码如下：

```
Private Sub Form_Load()                                ' 为 list1 控件添加数据
    List1.AddItem "苹果"
    List1.AddItem "桔子"
    List1.AddItem "木瓜"
    List1.AddItem "葡萄"
    List1.AddItem "西瓜"
    List1.AddItem "香蕉"
    List1.AddItem "火龙果"
    List1.AddItem "菠萝蜜"
End Sub

Private Sub Command1_Click()                            ' 移动表项至右边的列表框
    If List1.ListIndex < 0 Then                        ' 判断是否选择了表项
        MsgBox "没有选择！"
        Exit Sub
    Else
        List2.AddItem List1.Text                      ' 添加选中的表项到右边的列表框中
        List1.RemoveItem List1.ListIndex
    End If
End Sub

Private Sub Command2_Click()                            ' 移动表项至左边的列表框
    If List2.ListIndex < 0 Then                        ' 判断是否选择了表项
        MsgBox "没有选择！"
        Exit Sub
    Else
        List1.AddItem List2.Text                      ' 添加选中的表项到左边的列表框中
        List2.RemoveItem List2.ListIndex
    End If
End Sub
```

```

Else
    List1.AddItem List2.Text           '添加选中的表项到左边的列表框中
    List2.RemoveItem List2.ListIndex  '删除此表项
End If
End Sub
Private Sub Command3_Click()          '将所有表项全部添加到右边的列表框中
    Do While List1.ListCount          '判断是否到达最后一个表项
        List2.AddItem List1.List(0)  '每次移动最前面的表项
        List1.RemoveItem 0
    Loop
End Sub
Private Sub Command4_Click()          '将所有表项全部添加到左边的列表框中
    Do While List2.ListCount
        List1.AddItem List2.List(0)
        List2.RemoveItem 0
    Loop
End Sub
Private Sub List1_DblClick()          '双击列表框表项，也可以右移
    Command1_Click
End Sub
Private Sub List2_DblClick()          '双击列表框表项，也可以左移
    Command2_Click
End Sub

```

4.5.2 组合框

组合框是文本框和列表框的组合控件，它可以接收程序运行过程中的数据，使用灵活，既可以和文本框一样输入数据，又可以在组合框的已有选项中进行选择，因此它也同时具有文本框和列表框的大部分属性和事件。

组合框控件的 Text 属性以及名称的默认值都为 Combo1、Combo2、…。

1. 组合框的常用属性

组合框和文本框、列表框相同的属性不再重复，这里只介绍组合框的一些特殊属性。

(1) Style 属性(整数 0、1、2)

该属性决定组合框的样式，是只读属性，只能在设计时设置。

- Style 属性值为 0(该属性的默认值)，为下拉式组合框。下拉式组合框包括一个文本框和一个下拉式列表框，单击右端的箭头可以引出下拉式列表框，用户可以从其中作出选择；也可以在文本框中键入文本。
- Style 属性值为 1，为简单组合框。简单组合框包括一个文本框和一个非下拉式列表框，用户可以从列表中作出选择，也可以在文本框中键入文本。如果建立该控件时所画的列表框区域不够长，列表框可自动形成垂直滚动条。
- Style 属性值为 2，为下拉式列表框。下拉式列表框包括一个不可输入的文本框和一个下拉式列表框。单击箭头可以引出列表框，它限制用户输入。

图 4.8 所示的第一个组合框控件的 Style 属性值为 0，它不仅可以下拉、弹出选项的列表框，也可以在文本框内编辑；第二个组合框控件的 Style 属性值为 1，它类似于列表框控

件，但可以在文本框内输入；第三个组合框控件的 Style 属性值为 2，它不允许用户输入，其余特性与 Style 属性值为 0 的组合框情况相同。

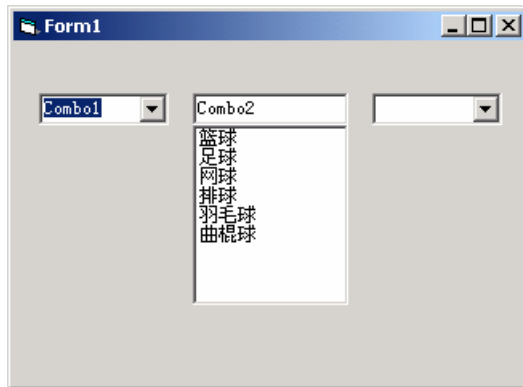


图 4.8 组合框控件 Style 属性不同设置的显示效果

(2) Text 属性(字符串类型)

对于组合框而言，一次只能选取一个选项，所以 Text 即为选中表项的文本。

2. 组合框的常用事件

组合框控件的事件取决于它的 Style 的属性值，只有简单组合框(Style=1)才可以接收 DbClick 事件，其他两种组合框可以接收 Click 事件和 DropDown 事件。而下拉式列表框(Style=2)的情况无法接收文本框的事件，如 Keypress 等。具体情况视问题不同而有所不同。

(1) Click 事件

用户单击组合框控件的列表部分选择表项的同时触发 Click 事件，此时 ListIndex 属性值就是组合框中所选表项的索引。

(2) DropDown 事件

用户单击组合框的下拉箭头时，触发事件。

(3) KeyPress 事件

对于 Style 属性值为 0 或 1 的组合框控件，KeyPress 事件可以用于修改或添加列表部分的表项。该事件由在其文本框中按任何键触发，一般应在按回车键(ASCII 码为 13)时执行修改或添加表项的操作。

下列 Combo1 控件的 KeyPress 事件过程可在文本框内新表项的输入结束(以回车为标志)后向组合框添加该表项：

```
Private Sub Combo1_KeyPress(KeyAscii As Integer)
    If KeyAscii = 13 Then
        Combo1.AddItem Combo1.Text
    End If
End Sub
```

【例 4.8】 设置两个组合框，分别显示运动项目“大球”和“小球”，大球的选项为“足球、篮球、排球”，小球的选项分别为“乒乓球、羽毛球”。程序运行时，显示如图 4.9 所示的界面。

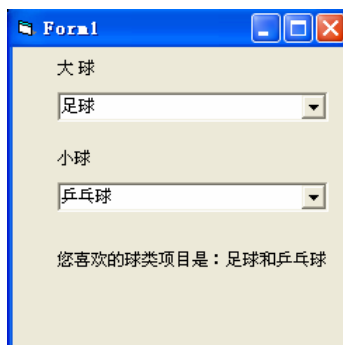


图 4.9 例 4.8 的界面设计

程序代码如下：

```
Private Sub Form_Load()
    Combo1.AddItem "足球" '为组合框添加数据
    Combo1.AddItem "篮球"
    Combo1.AddItem "排球"
    Combo2.AddItem "乒乓球"
    Combo2.AddItem "羽毛球"
    Label3.AutoSize = True
    Label3.Caption = ""
End Sub
Private Sub Combo1_Click()
    Label3.Caption = "您喜欢的球类项目是：" + Combo1.Text '选择第一个
End Sub '选项后显示的结果
Private Sub Combo2_Click()
    If Label3.Caption = "" Then '开始就选择小球则提醒先选择大球，同时
        MsgBox "请先选择大球类" '清空 combo2 的显示
        Combo2.Text = ""
    Else '在上一个选项的基础上继续选择
        Label3.Caption = Label3.Caption + "和" + Combo2.Text
    End If
End Sub
```

4.6 图像框和图片框

图像框和图片框控件都是为了更好地表现程序设计者的意图，以图形化的方法完成设计需要的一种手段。图像框一般作为图片显示居多，而图片框除了和图像框一样外，还可以担当框架的作用，使用更灵活，用途更广泛。

4.6.1 图片框

图片框(Picture Box)不仅可以用来显示图像，还可以作为其他对象的容器、显示图形方法的输出结果和 Print 方法输出的文本。其名称的默认值都为 Picture1、Picture2、…。

1. 图片框的常用属性

图片框的常用属性有以下几个。

(1) Picture 属性

这是图形控件最重要的属性，在使用时有两种方法。

界面设计时，选中该属性，在弹出的 LoadPicture 对话框中选择要显示的图片文件，相应的图片随之被加载到图片框中。

程序运行时可用 LoadPicture 函数装入图片，一般格式为：

```
[对象.]Picture=Loadpicture("文件名")
```

例如，Picture1.Picture=LoadPicture("C:\zhang\t.bmp")，该函数的参数是一个字符串表达式，包括驱动器、文件夹和文件名；加载一个空文件将清除图片框中原有的图片，如 Picture1.Picture=LoadPicture("")。

加载到图片框控件上的图片文件可以是以下格式之一：位图(.bmp、.dib、.cur)、图标(.ico)、图元(.wmf)、增强型图元(.emf)、JPEG 或 GIF。也可以来自 Windows 的各种绘图程序。

(2) AutoSize 属性

该属性值为 True 时，图片框的边界会随着所装入图片的大小变化而变化，图片将不考虑窗体上其他控件而调整大小，可能会覆盖其他控件，使用时注意调试。

(3) CurrentX 和 CurrentY 属性

用来设置下一个输出的水平(CurrentX)或垂直(CurrentY)坐标。这两个属性只能在运行时使用。如下所示：

```
Picture1.CurrentX=1000  
Picture1.CurrentY=2000
```

则下一个输出点在图片框内的(1000,2000)坐标处。

2. 图片框的特殊使用

和 Frame 控件一样，PictureBox 控件上也可以画出其他控件，这些控件随 PictureBox 移动而移动，它们的 Left 和 Top 属性是相对于 PictureBox 而言的，与窗体无关。

图片框还经常用来显示 Print 方法输出的文本和图形方法的输出结果。具体参照第 9 章中介绍的图形方法。

图片框也有很多事件，如 Chang、Click 等，使用方法视具体情况而定。

【例 4.9】 通过图片框交换图片。

在窗体中放置 4 个图片框，Picture1 用于显示第 1 幅图片，Picture2 用于显示第 2 幅图片，中间图片框为 Picture3，为了交换而设置，程序运行过程中它一直处于不可见状态，对结果的构图没有影响，Picture4 作为一个容器控件，用于装载命令按钮。在 Picture4 中设置两个命令按钮，一个是【交换图片】按钮，一个是【结束】按钮。设计的界面如图 4.10 所示。

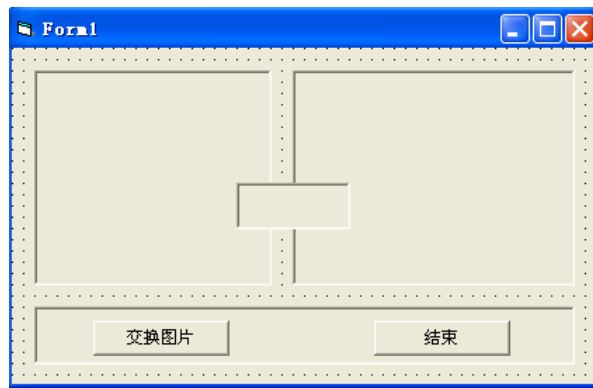


图 4.10 例 4.9 的界面设计

程序代码如下：

```
Private Sub Form_Load()  
    Picture3.Visible = False          '隐藏中间控件  
    Picture1.Picture = LoadPicture("f:\or.jpg")  
    Picture2.Picture = LoadPicture("f:\rose.jpg")  
End Sub  
Private Sub Command1_Click()  
    Picture3.Picture = Picture1.Picture '通过 picture3 进行交换  
    Picture1.Picture = Picture2.Picture  
    Picture2.Picture = Picture3.Picture  
End Sub  
Private Sub Command2_Click()  
End  
End Sub
```

程序运行结果如图 4.11 所示。



图 4.11 例 4.9 的运行结果

4.6.2 图像框

图像框只能用于显示图像，不支持图形方法，也不能当做容器来使用。在图形的使用上它和图片框有很多相似之处，这里不再复述。

作为图形控件,图像框在使用时需要较少的系统资源而且加载速度也比图片框更快。

和图片框不同的是图像框在加载图片时,可以通过 **Stretch** 属性使得图片适应图像框的大小,前面介绍的图片框控件,当它的 **AutoSize** 属性设置为 **True** 时,图片框的大小会随所装入的图片的大小而变化,这样可以得到图片的原始大小,但有时当所加载的图片比较大时,可能会影响窗体上其他控件的显示。而 **Stretch** 属性为 **True** 时,图片随控件大小调整。

图像框控件可接收 **Click** 等事件,因此可以充当图形命令按钮。

4.7 滚 动 条

滚动条控件分为水平滚动条(**HScrollBar**)控件和垂直滚动条控件(**VScrollBar**),在项目列表很长或者信息量很大时,可以使用滚动条来提供简便的定位。滚动条可以作为输入设备,或者速度、数量的指示器来使用。例如,可以用它控制计算机游戏的音量,或控制颜色值。

垂直和水平滚动条只是在滚动方向上不同,其余的属性和事件都是相同的。

水平滚动条控件名称的默认值为 **Hscroll1**、**Hscroll2**、...

垂直滚动条控件名称的默认值为 **Vscroll1**、**Vscroll2**、...

4.7.1 滚动条的常用属性

滚动条的常用属性有以下一些。

(1) Value 属性(整数类型)

在滚动条上的滑块所处的位置决定其 **Value** 属性的值,滑块处于最顶端或最左端时值最小,反之值最大。

改变滚动条 **Value** 属性的方法有 4 种。

- 直接在属性窗口中设定。
- 用鼠标单击滚动条两端的箭头,改变 **Value** 属性值。
- 用鼠标将滚动块(滚动条中间的矩形图像)沿滚动条拖动。
- 用鼠标单击滚动块两侧的空白部分,使滚动块以翻页的速度移动。

(2) Max 和 Min 属性(整数类型)

Max、**Min** 属性值是对滚动条 **Value** 属性取值范围的限制,分别代表最大、最小值。

(3) LargeChange 属性(整数类型)

该属性确定:当用户单击滚动块和滚动箭头之间的区域时,滚动条控件的 **Value** 属性值的改变量。

(4) SmallChange 属性(整数类型)

该属性确定:当用户单击滚动箭头时,滚动条控件的 **Value** 属性值的改变量。

4.7.2 滚动条的常用事件

滚动条的常用事件有以下两个。

(1) Change 事件

运行时，无论用何方法改变滚动条控件的 Value 属性值，都会导致该控件的 Change 事件被调用。

(2) Scroll 事件

该事件过程在只有在拖动滚动滑块时被调用。

【例 4.10】 设计一个图片框，使得它的背景颜色随着滚动条值的变化而变化。程序运行时，显示如图 4.12 所示的界面。

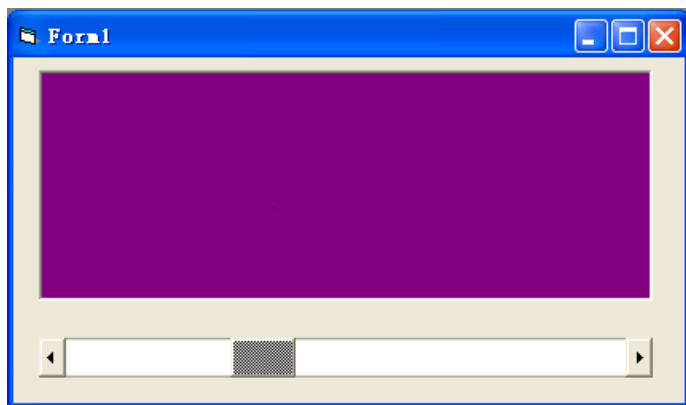


图 4.12 例 4.10 的运行界面

程序代码如下：

```
Private Sub Form_Load()  
    HScroll1.Min = 0           '滚动条初始设置  
    HScroll1.Max = 15  
    HScroll1.LargeChange = 2  
    HScroll1.SmallChange = 1  
End Sub  
Private Sub HScroll1_Change()  
    Picture1.BackColor = QBColor(HScroll1.Value) '图片背景颜色随滚动  
End Sub                                         '的变化而变化  
Private Sub HScroll1_Scroll()  
    Picture1.BackColor = QBColor(HScroll1.Value)  
End Sub
```

说明：QBColor 是一个颜色函数，值在 0~15 之间变化。题中，Scroll 事件和 Chang 事件的代码完全一样，因为 Chang 事件只有在 Value 值变化返回后事件才被激发，为了看到滑块的滚动过程中 Value 值的变化所造成的影响，可补充 Scroll 事件使程序完整。

4.8 定 时 器

定时器控件借用计算机内部的时钟，实现了由计算机控制、每隔一个时间段自动触发一个事件。它在运行时是不可见的，所以在设计阶段可以放置在窗体的任意位置。定时器控件默认的控件名称为 Timer1、Timer2、…。

1. 定时器的常用属性

定时器的常用属性有以下两个。

(1) Interval 属性(整数类型)

该属性表示定时的时间间隔，以毫秒为单位(即设置为 1000 时，时间间隔为 1 秒)。

Interval 属性值为 0，则定时器不起作用；Interval 属性的最大值为 65535。

(2) Enabled 属性(逻辑值)

当 Enabled 属性值为 True(默认值)时，激活定时器开始计时；当 Enabled 属性值为 False 时，定时器处于休眠状态，不计时间。

2. 定时器的 Timer 事件

定时器控件只可能响应一个事件，即该控件的 Timer 事件。在控件的 Enabled 属性值为 True 时，Interval 属性值的设定决定了间隔多少时间调用一次 Timer 事件。

【例 4.11】 设计一个如图 4.13 所示的电子时钟程序。

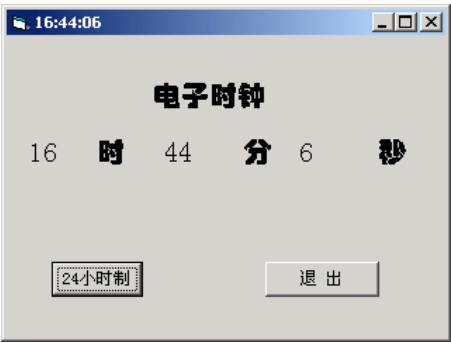


图 4.13 例 4.11 的界面

图 4.13 中界面中的控件的设置如表 4-3 所示。

表 4-3 例 4.11 的控件设置

控件名称	控件显示
Command1	Caption 为“24 小时制”，它用来切换制式：12 小时制和 24 小时制
Command2	Caption 为“退出”
Label1	Caption 为“电子时钟”
Label2	空，运行显示从系统内提取时间的“时”

续表

控件名称	控件显示
Label3	Caption 为“时”
Label4	空，运行时显示从系统内提取时间的“分”
Label5	Caption 为“分”
Label6	空，运行时显示从系统内提取时间的“秒”
Label7	Caption 为“秒”
Timer1	Interval=1000

程序代码如下：

```

Private Sub Command1_Click() ' 设定活动按钮 command1 使它可以在制式之间转换
Static nflag As Boolean
If nflag = False Then
Command1.Caption = "12 小时制"
nflag = True
Else
Command1.Caption = "24 小时制"
nflag = False
End If
End Sub
Private Sub Command2_Click() ' 退出按钮
End
End Sub
Private Sub Timer1_Timer()
Form1.Caption = Time ' 窗体的 Caption 设置成活动的时间，
' 它随时钟的变化而变化，即使是最小化也可以看见时
' 间的变化

If Command1.Caption = "12 小时制" Then
Label2.Caption = Hour(Now) Mod 12
Else
Label2.Caption = Hour(Now)
End If
Label4.Caption = Minute(Now)
Label6.Caption = Second(Now)
End Sub

```

该程序放大显示的是计算机系统的时间，只是每隔 1 秒钟将显示结果刷新 1 次。

4.9 小 结

控件是在图形用户界面上输入、输出信息、启动事件程序等交互操作的图形对象，是进行可视化程序设计的基础和重要工具。本章主要介绍了 Visual Basic 工具箱中提供的 12 个常用控件，这些控件有命令按钮、标签框、文本框、复选框、单选按钮、框架、图像框、图片框、列表框、组合框、滚动条和定时器，介绍了每类控件的常用属性、事件和方法。

4.10 习 题

一、选择题

1. 标签控件的标题和文本框控件的显示文本的对齐方式由()属性决定。
A) WordWrap B) AutoSize C) Alignment D) Style
2. 按 Tab 键时, 焦点在各个控件之间移动的顺序是由()属性决定的。
A) Index B) TabIndex C) TabStop D) SetFocus
3. 在文本框中按下回车键接收 KeyPress 事件时, 回车所响应的 ASCII 码为()。
A) 12 B) 14 C) 13 D) 127
4. 单选按钮选中状态的 Value 值是()。
A) 0 B) 1 C) False D) True
5. 要使复选框不响应 Click 时间, 可设置复选框的()属性。
A) Apperance B) Style C) Enabled D) TabIndex
6. 要在同一窗体中安排两组单选按钮, 可用()控件予以分隔。
A) 文本框 B) 框架 C) 列表框 D) 组合框
7. 列表框中表示表项值的属性是()。
A) ListIndex B) List C) Selected D) Caption
8. 以下()语句将删除列表框 List1 中的最后一项。
A) List1.RemoveItem List1.ListCount B) List1.Clear
C) List1.List(List1.ListCount-1)= "" D) List1.RemoveItem List1.ListCount-1
9. 要把组合框中的文本加入到列表项中的第 3 项, 应执行语句()。
A) Combo1.AddItem 3, Combo1.Text B) Combo1.AddItem Combo1.Text3
C) Combo1.AddItem Combo1.Text2 D) Combo1.AddItem 3, Combo1.Text
10. 要使得图片适合控件的大小, 以下合适的语句是()。
A) Picture1.Strech=True B) Image1.AutoSize=True
C) Picture1.AutoSize=True D) Image1.Strech=True
11. 单击滚动条两端任意一个滚动箭头, 将触发该滚动条的()事件。
A) KeyPress B) Scroll C) Click D) Change
12. 在窗体上画一个名称为 Timer1 的计时器控件, 要求每隔 0.5 秒发生一次计时器事件, 则以下正确的属性设置语句是()。
A) Timer1.Interval=0.5 B) Timer1.Interval=5
C) Timer.Interval=50 D) Timer1.Interval=500
13. 在窗体(名称为 Form1)上画一个名称为 Text1 的文本框和一个名称为 Command1 的命令按钮, 然后编写一个事件过程。程序运行以后, 如果在文本框中输入一个字符, 则把命令按钮的标题设置为“计算机等级考试”。以下能实现上述操作的事件过程是()。

A) Private Sub Text1_Change()

```
    Command1.Caption="计算机等级考试"  
End Sub
```

B) Private Sub Command1_Click()

```
    Caption="计算机等级考试"  
End Sub
```

C) Private Sub Form1_Click()

```
    Text1.Caption="计算机等级考试"  
End Sub
```

D) Private Sub Command1_Click()

```
    Text1.Text="计算机等级考试"  
End Sub
```

14. 假定在图片框 Picture1 中装入了一个图形,为了清除该图形(不删除图片框),应采用的正确方法是()。

A) 选择图片框,然后按 Del 键

B) 执行语句 Picture1.Picture=LoadPicture(" ")

C) 执行语句 Picture1.Picture=""

D) 选择图片框,在属性窗口中选择 Picture 属性,然后按回车键

15. 在窗体上画一个文本框和一个计时器控件,名称分别为 Text1 和 Timer1,在属性窗口中把计时器的 Interval 属性设置为 1000,Enabled 属性设置为 False,程序运行后,如果单击命令按钮,则每隔 1 秒钟在文本框中显示一次当前的时间。以下是实现上述操作的程序:

```
Private Sub Command1_Click()  
    Timer1._____  
End Sub  
Private Sub Timer1_Timer()  
    Text1.Text = Time  
End Sub
```

在_____处应填入的内容是()。

A) Enabled=True

B) Enabled=False

C) Visible=True

D) Visible=False

二、填空题

1. 组合框有_____和_____两种控件的功能。

2. 在列表框中删除所有表项的语句是_____。

3. 滚动条触发 Change 事件是由_____属性决定的。

4. 文本框的 MaxLength 为 0 时,表示_____。

5. 窗体上有一个名称为 List1 的列表框,一个名称为 Text1 的文本框,一个名称为 Command1、标题为“计算”的命令按钮。程序运行后,将把 1~100 之间能够被 7 整除的数添加到列表框。如果单击【计算】按钮,则对 List1 中的数进行累加求和,并在文本框中显示计算结果,如图 4.14 所示。以下是实现上述功能的程序,请填空。

```

Private Sub Form_Load()
For i=1 To 100
If i Mod 7 =0 Then
    1
End If
Next
End Sub
Private Sub Command1_Click()
Sum =0
For i=0 To 2
Sum =Sum+ 3
Next
Text1.Text=Sum
End Sub

```

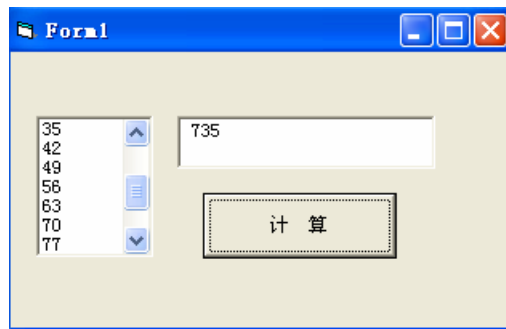


图 4.14 填空题 5 的效果图

6. 有 5 个 Command 按钮，分别是剪切、复制、粘贴、删除、清空按钮，请填空。

```

Private Sub 复制_Click()
Clipboard.Clear
Clipboard.SetText ( 1 )
End Sub
Private Sub 剪切_Click()
Clipboard.Clear
Clipboard.SetText ( 2 )
3
End Sub
Private Sub 粘贴_Click()
4
End Sub
Private Sub 删除_Click()
5
End Sub
Private Sub 清空_Click()
Text1.Text = 6
End Sub

```

三、编程题

1. 设计一个密码验证程序，在第一个窗体上有一个文本框，输入最大长度为 6 位的字符串，如果密码正确，则进入第二个窗体，该窗体上有一个从左至右滚动的标签“欢迎光临！”，如果密码错误，则被拒绝，最多允许输入 3 次。
2. 如图 4.15 所示，建立 1 个文本框，1 个列表框，在文本框输入单词后按下回车键即可显示在列表框里，双击列表框中某个单词则可删除之。
3. 如图 4.16 所示，建立 1 个文字字型、字体变化程序。
4. 在窗体上制作 1 个动态秒表，其中有两个命令按钮控制开始和结束。

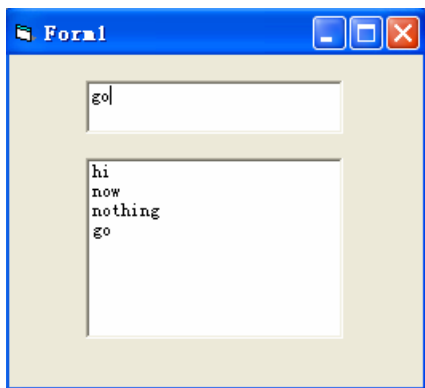


图 4.15 编程题 2 的效果图



图 4.16 编程题 3 的效果图

第 5 章 数 组

教学提示：数组是有序数据的集合。数组中的每一个元素都属于同一个数据类型，用一个统一的数组名和下标可唯一地确定数组中的元素。在处理大批数据时，例如大批量数据的排序、统计、矩阵运算等，使用数组可以缩短程序代码、提高程序的可读性和执行效率。

根据数组对内存占有的情况可将其分为动态和静态数组。动态数组使用完成后即释放数组空间，也可重新利用。静态数组事先分配存储空间，运行过程中不再改变。

本章主要介绍数组的定义和使用、多维数组的理解和应用以及在使用过程中静态数组和动态数组的灵活运用。

教学要求：正确理解数组的概念，掌握数组的定义和使用方法，能够正确使用一维数组、二维数组、动态数组和控件数组。

5.1 数组的基本概念

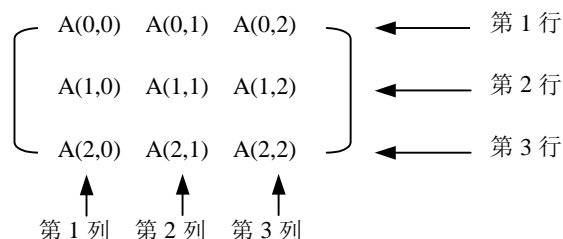
5.1.1 数组与数组元素

中学数学中已经学过关于数组的定义：一组可顺序索引并具有相同内部数据类型的元素。数组中每个元素具有唯一的索引号。如： $S = [S_1, S_2, S_3]$ 。例如，班级中有 30 名同学，如果期末计算成绩，标示每个学生某门课程的成绩为 $S_1, S_2, S_3, S_4 \cdots S_{30}$ ， S 代表班级，1、2、3、4 代表学号， S_1 代表班级中学号为 1 号的同学的成绩，以此类推，代码可以标识出班级中所有同学的成绩，利用这些有规则的数据进行求和、求平均数等统计运算非常简单方便。

在 Visual Basic 中把一组具有相同名字、不同下标的数据称为数组，表示为 $A(n)$ ，其中 A 为数组的名称， n 为下标变量，不同的下标表示不同的数据。 $A(1)$ 表示 A 数组的第一个元素， $A(n)$ 表示 A 数组的第 n 个元素。

5.1.2 数组的维数

数组在使用时有多种形式，有时需要表示坐标数据或其他多维数据时，需要使用多维数组。例如，3 行 3 列的二维数组 $A(3,3)$ ：



每个数据都是由它的下标唯一标识，例如 A(1, 1)表示第 2 行第 2 列的数据，A(2, 2)表示第 3 行第 3 列的数据。这里要说明的是在 Visual Basic 里除非加以特殊说明，计数都从 0 开始。这样的多维数据的表示容易定位归类，方便处理。

5.2 数组的定义

5.2.1 静态数组的定义

数组在使用时要严格遵循先定义后使用的原则，目的是通知计算机为其留出所需要的空间。在程序的通用对象声明部分可用语句 `Option Base 1` 声明所有数组第 1 个元素下标为 1，`Option Base 0` 声明所有数组的第 1 个元素下标为 0(默认值)。`Option Base` 语句必须放在数组定义之前。

数组声明示例：

```
Dim y(5) As Integer ' 声明 y 是 Integer 类型数组
```

其中，不加 `Option Base` 语句说明的情况下，y 数组的最小下标从 0 开始，最大下标为 5，即数组有 6 个数：y(0)、y(1)、y(2)、y(3)、y(4)和 y(5)。

如果程序中有语句“`Option Base 1`”，则如下语句：

```
Dim m(6) As Single
```

所定义的数组 m 的元素有 m(1)、m(2)、m(3)、m(4)、m(5)和 m(6)共 6 个，不包括 a(0)，其最大下标和数组个数吻合。

数组还有一种表示方式不受 `Option Base` 语句的影响，即程序员在定义数组时指定数组的上下界线，如：`Dim x(1 to 5) As Double`，表示数组 x 的元素有 x(1)、x(2)、x(3)、x(4)和 x(5)。

多维数组的定义和一维数组相似，定义时分别标识出每一维下标的界线即可，例如：`Dim S(2,3)`在不加特殊说明的情况下，表示一个 3 行 4 列的二维数组，数据元素的类型为变体类型。

5.2.2 动态数组的定义

使用定长数组要预先知道需要存储的数据量的大小，但有时数据量的大小不能确定，这就需要用到动态数组。动态数组不需要在声明中指出它的存储量的大小，它可以在运行时根据需求动态改变其大小或上下界，使用完成后内存即被释放，从而有效利用了存储空间。

动态数组在创建时需要在数据声明部分先定义数据类型，但不需要说明数据的大小，然后在使用中根据需要确定数组的元素个数。即可用 **ReDim** 语句分配数组的元素个数，称为数组重定义。例如：

```
Dim s() As Integer    '声明动态数组 s 是整型
...
n=val(inputbox("请输入数组的大小! "))
Redim s(n)            '定义数组最大下标为 n
...
```

动态数组在程序的声明阶段先说明数组的类型，在程序运行时动态地改变数组的大小。**ReDim** 语句有很多具体的限制。

(1) 仅可以在过程级出现。这意味着可以在过程中而不是在类或模块级重新定义数组。

(2) **ReDim** 语句无法更改数组变量的数据类型。

(3) **ReDim** 语句可以用来更改已被正式声明的数组的一个或多个维度的大小。即利用 **Redim** 可以多次定义同一个数组，随时修改数组中元素的个数，但不能修改维数。

数据可以被保留也可以被删除，具体操作见保留动态数组的内容。

5.3 数组的基本操作

5.3.1 数组元素的输入与输出

数组元素由于它的规整性，使得利用它输入输出大批量的数据十分方便，当数据具有一定的规则时，尤其能显示出它的优越性。

1. 一维数组的输入输出

(1) 规则数据或自动生成数据的输入和输出

【例 5.1】 随机生成 10 个两位正整数，并在窗体上打印出来。

```
Option Base 1          '数组下界从 1 开始计数
Private Sub Form_Click()
    Dim i As Integer    '数据定义
    Dim x(10) As Integer
    For i = 1 To 10      '数据输入
        x(i) = Int(Rnd * 90) + 10
    Next i
    For i = 1 To 10      '数据输出
        Print x(i);
    Next i
End Sub
```

规则数据的输入更简单，只要在循环中说明数组的递增或递减规律即可。例如，从字符“a”开始，打印“acegikmoqs”这 10 个字符，代码如下：

```
x(1) = Asc("a")        '设定起始点
For i = 1 To 9
    x(i + 1) = x(i) + 2
```

```

        Print Chr(x(i));
    Next i
    Print Chr(x(i))          '打印最后一个

```

(2) 不规则数据的输入和输出

有些数组中的数据没有一定的规则可以遵循，比如学科成绩、学生姓名等，每个数据都需要一个个地录入，输出时也要根据一定的要求进行输出。

【例 5.2】 输入 5 个学生的姓名，输出其中姓张的同学的姓名。

```

Option Base 1
Private Sub Form_Click()
    Dim i As Integer
    Dim x(5) As String
    For i = 1 To 5
        x(i) = InputBox("请输入姓名: ")
        If Left(x(i), 1) = "张" Then Print x(i);
    Next i
End Sub

```

程序中直接把输入和输出放在一个循环里处理，动态输入，动态输出。

2. 二维数组的输入输出

二维数组的输入输出是根据二维数组的特点，对数组一行一行扫描的结果。这样就产生了两层循环，外层循环处理行，内层循环处理列。

【例 5.3】 把如下数据放入一个 3 行 3 列的数组中，并打印出来。

2,3,5
4,6,8
4,7,9

```

Option Base 1
Private Sub Form_Click()
    Dim i As Integer
    Dim s(3, 3) As Byte
    For i = 1 To 3          '数据输入
        For j = 1 To 3
            s(i, j) = Val(InputBox("请输入数据: "))
        Next j
    Next i
    For i = 1 To 3          '数据输出
        For j = 1 To 3
            Print s(i, j);   '同行打印
        Next j
        Print                '换行
    Next i
End Sub

```

二维数组的输出要注意在一行打印完后要换行。

3. 初始化数据

数据在定义以后都有一个默认的数据，默认数据的赋值规则如数据类型说明中所述，数值类型初值为 0，字符型数组初始值为空等。这对一维、二维数组都适用。

在 Visual Basic 中有允许对变体一维数组中的数据进行初始化的函数 Array，如：

```
Dim S                                '声明变量 S 为变体类型
S=Array(1,2,3,4)                    '为 S 数组的每个数据赋初值
```

5.3.2 数组元素的复制

数组元素可以像变量一样灵活操作。

1. 单个元素的复制

对于一维和二维数组而言，每个数据，如 A(2)是 A 数组的第 3 个数据，B(1,2)是 B 数组的第 2 行第 3 列的数据，都可以单独当做一个普通变量来赋值。例如：

```
A(2)=B(1,2)
```

即把 B(1,2)的值赋给了 A(2)。

2. 整个数组的复制

数组的复制仍然要使用循环来完成。如：

```
Dim a(10) As Integer
Dim b(10) As Integer
For i = 1 To 10
    b(i) = a(i)
Next i
```

对于二维数组，只需要改成两层循环，b(i,j)=a(i,j)，行对应行，列对应列，就可以完全复制过去。

当然，数据有时候需要一些简单的变化，这时只需对数组的下标进行操作，非常灵活。

【例 5.4】 对一个自动生成两位正整数的 5 行 5 列的二维数组进行转置。

分析：转置的概念就是将列的数据和行的数据交换，用如下一个 3 行 3 列的数组说明：

$$\begin{bmatrix} 12, 23, 45 \\ 24, 68, 96 \\ 41, 78, 22 \end{bmatrix} \longrightarrow \begin{bmatrix} 12, 24, 41 \\ 23, 68, 78 \\ 45, 96, 22 \end{bmatrix}$$

代码如下：

```
Option Base 1
Private Sub Command1_Click()
Dim a(5, 5) As Integer
Dim b(5, 5) As Integer
For i = 1 To 5                '数据自动生成
For j = 1 To 5
a(i, j) = Int(Rnd * 90) + 10
Next j
```



```

Next i
For i = 1 To 5                '打印 a 数组
    For j = 1 To 5
        Print a(i, j);
    Next j
    Print
Next i
Print
For i = 1 To 5                'b 数组转向继承 a 数组的数据
    For j = 1 To 5
        b(j, i) = a(i, j)
    Next
Next
For i = 1 To 5                '打印 b 数组
    For j = 1 To 5
        Print b(i, j);
    Next
    Print
Next
End Sub

```

其中 $b(j, i) = a(i, j)$ 正是 b 数组转向的复制了 a 数组的数据的代码, 它用于将行列进行互换:

$a(1,2) \longleftrightarrow b(2,1)$ 、 $a(1,3) \longleftrightarrow a(3,1) \cdots$

5.3.3 保留动态数组的内容

利用 ReDim 语句动态分配数组时, 原数组的内容都会丢失, 数组元素的值都被初始化 (数值型被置为 0, 字符型被置为空串, 逻辑型被置为 False 等)。

如果要保留原数组的值, 就要在数据的重置时在语句中加上 Preserve 关键字, Visual Basic 将把这些元素从现有数组中复制到新数组。

格式如下:

```
ReDim [ Preserve ] 数组名 (数组维数)
```

例如: 有如下代码:

```

Dim I as Integer, M() As Integer    '声明数组类型
ReDim M(5)                          '动态定义了 6 个元素
For I = 0 To UBound(M)
    M(I) = I                        '赋值
Next I
Redim Preserve M(15)                '重新定义数组 M 并保留原数据

```

语句 ReDim Preserve M(15)重新定义了数组的大小, 但保留了数据原始的值, 扩充的部分由数据类型的默认值填充, 这里是 0。如果重新定义的数组元素个数少于原数组的元素个数, Visual Basic 会自动截取。

说明: Ubound(数组名)是取数组最大下标的函数, 与之对应的是 Lbound(数组名), 它提取的是数组的最小下标。

5.4 数组应用举例

在数组的使用中,排序的重要性是显而易见的。排序又称为分类(Sorting)算法,是程序设计中常用的算法。排序的方法非常多,这里介绍选择排序法(Selection Sort)。


【例 5.5】 用随机函数产生 20 个 2 位正整数,用选择法排序法将它们按值从小到大排序输出。

选择排序法说明:在数组中搜寻出最小(大)的数,把它与第 1 个数对调,这样,第 1 个位置上放的是最小(大)的数,然后在此基础上,在数组中从第 2 个数开始寻找第 2 个最小(大)的数,再与第 2 个数互换,依次类推,直至全部排列完成。

本程序可以利用一个哨兵来帮助完成,哨兵每次出发都是寻找本次循环中的最小(大)值。


以如下 6 个数为例(Option Base 1),首先设定一个哨兵 $a(k)$,初始状态和 $a(1)$ 一样大小:

12	72	6	89	3	83
----	----	---	----	---	----



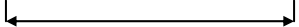
第 1 次排序:哨兵遍历数组,找到所有数中最小的数 3,其下标为 5,然后把哨兵 $a(k)$ 与 $a(1)$ 互换,即 $a(1)$ 与 $a(5)$ 互换,结果如下。

3	72	6	89	12	83
---	----	---	----	----	----




第 2 次排序:哨兵变成 $a(2)$,从第 2 个数开始,找到最小的数 $a(3)$,把 $a(2)$ 与 $a(3)$ 互换,此时,第 1 和第 2 个数已经有序,其后的排序从第 3 个数开始,结果如下。

3	6	72	89	12	83
---	---	----	----	----	----




第 3 次排序:哨兵变成 $a(3)$,从第 3 个数开始,寻找到最小数为 12,即 $a(5)$,将 $a(k)$ 与 $a(3)$ 互换。前 3 个数有序,结果如下。

3	6	12	89	72	83
---	---	----	----	----	----



第 4 次排序:哨兵变成 $a(4)$,从第 4 个数开始,找到最小数为 72,即 $a(5)$,将 $a(4)$ 与 $a(5)$ 互换。前 4 个数有序,结果如下。

3	6	12	72	89	83
---	---	----	----	----	----



最后一次排序:哨兵变成 $a(5)$,从第 5 个数开始,找到最小数为 83,将 $a(5)$ 与 $a(6)$ 交换。全部排序工作完成,结果如下。

3	6	12	72	83	89
---	---	----	----	----	----

一般地, N 个数经过 $N-1$ 次排序后均有序。通过以上选择法排序的展开分析, 可以归纳出选择排序法算法如下:

```

For i=1 to n-1
    找出 a(i)至 a(n)间最小的数组元素下标 k
    交换 a(i)与 a(k)
Next i
' 以下为找出 a(i)至 a(n-1)之间值最小的数组元素下标 k 的代码
k=i                      ' 假设下标为 i 的元素值最小
For j=i+1 to n
    If a(j)<a(k) then k=j
Next j

```

程序代码如下:

```

Private Sub Form_click()
    Dim a(1 to 20) As Integer, temp As Integer
    Dim i As Byte, j As Byte, k As Byte
    For i=1 To 20                ' 产生数据
        a(i)=Int(Rnd*90)+10
    Next i
    For i=1 To 19                ' 排序
        k=i                      ' 设定哨兵
        For j=i+1 To 20          ' 查找最小值
            If a(j)<a(k) Then k=j
        Next j
        Temp=a(i): a(i)=a(k): a(k)=Temp ' 交换哨兵(最小值)和排序位的数据
    Next i
    For i=1 To 20                ' 打印输出
        Print a(i);
    Next i
    Print
End Sub

```

【例 5.6】 把一个字符插入到一个基本有序的字符串中。

分析: 输入字符串的长度不固定, 所以采用动态数组的形式, 而插入的动作包括以下几个。

- (1) 比较插入字符的 ASCII 码在字符串中的大小, 确定插入位置。
- (2) 把已有字符串从插入位置开始到最末端向后逐个移位, 如有数组 a , 即 $a(i+1)=a(i)$ 。
- (3) 把插入字符放到留出的空位中。

以含 5 个字符的字符串为例: “abdef”, 插入字符为 “c”。

首先找到字符 “d” 为插入位, 下标为 3, 然后扩大数组, 动态增加 1 个元素位, 为 c 留下空间, 此时, 字符串为 “abdef” & “”, 然后从 “f” 字符开始到 “d” 逐个后移, 形成字符串 “abddef”, 最后把 “c” 赋值给插入位 $a(3)$, 形成字符串 “abcdef”。

程序代码如下:

```

Option Base 1
Private Sub Form_Click()
Dim a() As String
Dim s As String
Dim n As Integer
s = InputBox("请输入有序字符串: ")
n = Len(s)
ReDim a(n)
For i = 1 To n
    a(i) = Mid(s, i, 1)          '将串中字符逐个取出, 放入数组 a 中
Next i
addstr = InputBox("请添加需要插入的字符: ")
ReDim Preserve a(n + 1)         '保留数据, 增加一个数组元素
For i = 1 To n
    If addstr < a(i) Then        '寻找插入位
        f = i: Exit For
    Else
        f = n + 1
    End If
Next i
For i = n + 1 To f + 1 Step -1   '倒序重新排列
    a(i) = a(i - 1)
Next i
a(f) = addstr                   '插入字符
For i = 1 To n + 1
    Print a(i);
Next i
End Sub

```

数组应用中对二维数组的统计也是比较常用的。比如分数的求和, 求平均数等。

【例 5.7】 输入一个 3 行 4 列的数组, 求它的和以及最大值。

程序代码如下:

```

Option Base 1
Private Sub Form_Click()
Dim x(3, 4) As Single
Dim i As Byte, j As Byte, max As Single, s As Single
For i = 1 To 3                  '赋值并打印
    For j = 1 To 4
        x(i, j) = Val(InputBox("请输入数据 x(" & i & ", " & j & ") 的值: "))
        Print x(i, j);
    Next j
    Print
Next i
max = x(1, 1)                   '设定最大值 max 的初始值
For i = 1 To 3
    For j = 1 To 4
        If x(i, j) > max Then max = x(i, j)   '求最大数
        s = s + x(i, j)                       '求和
    Next j
Next i
Print "此数组的最大值为: " & max              '打印结果
Print "此数组的和为: " & s
End Sub

```

【例 5.8】 建立一个 5 行 5 列的二维数组，两条对角线上的元素为 1，其余元素为 0。

程序代码如下：

```
Private Sub Form_Click()  
    Dim s(1 To 5, 1 To 5) As Integer  
    Dim i As Integer, j As Integer  
    For i=1 To 5  
        For j=1 To 5  
            If i=j Or i+j=6 Then          '对角线上的元素赋值 1,其余赋为 0  
                s(i, j)=1  
            Else  
                s(i, j)=0  
            End If  
            Print s(i, j);                '打印输出  
        Next j  
        Print  
    Next i  
End Sub
```

程序的输出结果如下：

```
1  0  0  0  1  
0  1  0  1  0  
0  0  1  0  0  
0  1  0  1  0  
1  0  0  0  1
```

5.5 控件数组

当界面上需要若干个控件执行大致相同的操作时，可以将这些控件定义为控件数组。控件数组由一组类型相同的控件组成，这些控件共用一个相同的控件名，共享同一个事件过程。

控件数组中每个元素都有唯一与之相关联的下标，或称为索引(Index)。控件数组中的控件由它的索引，即下标来进行区分，如 Command1(0)、Command1(1)。由于控件数组是自动生成的，所以控件的下标由系统内定，与 Option Base 的设置无关，总是从 0 开始计数。

5.5.1 控件数组的建立

控件数组的建立有两种方法。

1. 直接建立

直接建立控件数组的方法如下。

(1) 先建立一个初始控件，如图 5.1 所示。

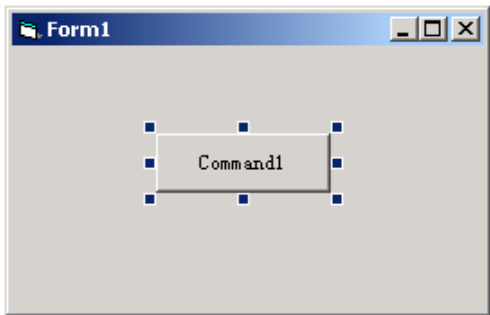


图 5.1 建立初始控件

(2) 在窗体上单击该控件后，选择【编辑】菜单中的【复制】命令(或按 Ctrl+C 键)，然后选择【编辑】菜单中的【粘贴】命令(或按 Ctrl+V 键)。此时弹出如图 5.2 所示的对话框。



图 5.2 建立数组控件对话框

(3) 选择【是】，则控件 Command1 更名为 Command1(0)，新建控件名称为 Command1(1)，继续粘贴则建立控件 Command1(2)、Command1(3)、…，它们构成了一个控件数组 Command1，每个控件都是数组中的一个元素。

2. 修改控件的名称

通过修改控件的名称来建立控件数组的方法如下。

(1) 在容器内画出所需要的所有控件(为一个类型)，如 Command1、Command2，Command3、…。

(2) 选择一个母体，如 Command1，然后对别的控件改名，改成与母体相同的名称，这时 Visual Basic 会询问是否建立控件数组，选择【是】即可。

以上两个方法都可以建立控件数组，所不同的是如果控件有显示的文本，则用直接建立方法建立的控件数组中所有控件显示的文本都相同，而通过修改名称得到的控件数组其显示文本还保留了初始建立时的状态。

5.5.2 控件数组的使用

控件数组的使用就是要灵活应用控件属性中的 Index 属性，它标识的就是控件数组的下标。

以例 4.6 为例，每个 Frame 控件中的单选按钮如果以控件数组的形式建立，程序代码就可以简化很多。

【例 5.9】 以控件数组的方式重做例 4.6。

这个程序中将例 4.6 中的两组单选按钮定义为两个控件数组，即两个 Frame 中分别以 Option1 和 Option2 做母体，派生出各自 2 个子个体。

注意在 Frame 中建立控件数组时，粘贴一定要在框架里，形成的子控件和母体控件在一个框架中。

程序代码如下：

```
Private Sub Form_Load()  
    Text1.Text = "Visual Basic"  
End Sub  
Private Sub Check1_Click()  
    If Check1.Value = 1 Then  
        Text1.FontBold = True  
    Else  
        If Check1.Value = 0 Then Text1.FontBold = False  
    End If  
End Sub  
Private Sub Check2_Click()  
    If Check2.Value = 1 Then  
        Text1.FontUnderline = True  
    Else  
        If Check2.Value = 0 Then Text1.FontUnderline = False  
    End If  
End Sub  
Private Sub Check3_Click()  
    If Check3.Value = 1 Then  
        Text1.FontItalic = True  
    Else  
        If Check3.Value = 0 Then Text1.FontItalic = False  
    End If  
End Sub  
Private Sub Option1_Click(Index As Integer)    '控件数组以 Index 作为下标  
    Select Case Index  
    Case 0  
        Text1.FontSize = 10  
    Case 1  
        Text1.FontSize = 20  
    Case 2  
        Text1.FontSize = 30  
    End Select  
End Sub  
Private Sub Option2_Click(Index As Integer)  
    Select Case Index  
    Case 0  
        Text1.FontName = "黑体"  
    Case 1  
        Text1.FontName = "楷体_gb2312"
```

```

Case 2
Text1.FontName = "宋体"
End Select
End Sub

```

由此可见，控件数组使代码更为简洁。

【例 5.10】 编写一个月食程序。

在窗体中放置一个图片控件数组 **Image1**，使用图片控件数组保存月食变化的 5 幅图片，再在窗体中放置一个图片控件 **Image2**，用来循环显示 5 幅图片中的 1 幅图片，5 幅图片不断变化的图片效果利用 **Timer** 控件来完成。设计的界面如图 5.3 所示。

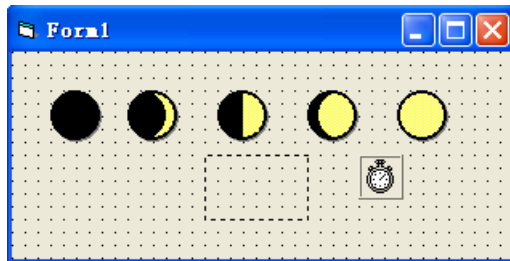


图 5.3 例 5.10 的界面设计

程序代码如下：

```

Dim i As Integer                                ' 设置 I 为窗体级变量
Private Sub Form_Load()                        ' 设置图像过程不可见
For j = 0 To 4
Image1(j).Visible = False
Next j
End Sub
Private Sub Timer1_Timer()
Image2.Picture = Image1(i).Picture            ' 由 Image2 显示变化过程
i = i + 1
If i = 5 Then i = 0                            ' 循环播放
End Sub

```

5.6 小 结

数组是有序数据的集合。数组中的每一个元素都属于同一个数据类型，用一个统一的数组名和下标来唯一地确定数组中的元素。

数组可以是一维数组，也可以是多维数组(如二维数组)。

根据数组对内存占有的情况可将其分为动态和静态数组。动态数组使用完成后即释放数组空间，也可重新利用。静态数组事先分配存储空间，运行过程中不再改变。

数组元素通过下标访问，因而数组常用循环结构处理。

数组元素为控件时，称为控件数组，用于界面设计时组织相关操作的对象。

5.7 习 题

一、选择题

1. 用下面的语句定义的数组元素的个数是()。

```
Dim s(7) as Integer
```

- A) 6 B) 7 C) 8 D) 9

2. 用下面的语句定义的数组元素的个数是()。

```
Dim s(3,4) as String
```

- A) 7 B) 12 C) 20 D) 18

3. 定义数组 Array(1 to 5,5)后, 下列哪一个数组元素不存在? ()

- A) Array(1,1) B) Array(1,0) C) Array(0,1) D) Array(5,5)

4. 在过程中定义 Dim x(1 to 10,3) As Single, 则数组占用()字节的内存空间。

- A) 132 B) 80 C) 160 D) 120

5. 在窗体上画一个名称为 Command1 的命令按钮, 然后编写如下事件过程:

```
Option Base 1
Private Sub Command1_Click()
    Dim a
    a= Array(1,2,3,4,5)
    For i=1 To UBound(a)
        a(i) = a(i)+i-1
    Next i
    Print a(3)
End Sub
```

- 程序运行后, 单击该命令按钮, 则在窗体上显示的内容是()。

- A) 4 B) 5 C) 6 D) 7

6. 有如下程序:

```
Option Base 1
Private Sub Form_Click()
    Dim arr,Sum
    Sum = 0
    arr = Array(1,3,5,7,9,11,13,15,17,19)
    For i=1 To 10
        If arr(i)/3 = arr(i)\3 Then
            Sum =Sum+arr(i)
        End If
    Next i
    Print Sum
End Sub
```

- 程序运行后, 单击窗体, 输出结果为()。

- A) 13 B) 14 C) 27 D) 15

7. 在窗体上画一个名称为 Label1 的标签, 然后编写如下事件过程:

```
Private Sub Form_Click()  
    Dim arr(10, 10) As Integer  
    Dim i As Integer, j As Integer  
    For i = 2 To 4  
        For j = 2 To 4  
            arr(i, j) = i * j  
        Next j  
    Next i  
    Label1.Caption = Str(arr(2, 2) + arr(3, 3))  
End Sub
```

程序运行后, 单击窗体, 在标签中显示的内容是()。

- A) 12 B) 13 C) 14 D) 15

8. 有如下程序:

```
Option Base 1  
Dim arr() As Integer  
Private Sub Form_Click()  
    Dim i As Integer, j As Integer  
    ReDim arr(3, 2)  
    For i = 1 To 3  
        For j = 1 To 2  
            arr(i, j) = i * 2 + j  
        Next j  
    Next i  
    ReDim Preserve arr(3, 4)  
    For j = 3 To 4  
        arr(3, j) = j + 9  
    Next j  
    Print arr(3, 2) + arr(3, 4)  
End Sub
```

程序运行后, 单击窗体, 输出结果为()。

- A) 21 B) 13 C) 8 D) 25

二、填空题

1. 控件数组的名字由_____属性指定, 而数组中的每个元素由_____属性指定。
2. Array 函数建立的数组类型必须是_____类型, 而且只对_____维数组有效。
3. 以下程序的功能是用 Array 函数建立一个含有 8 个元素的数组, 然后查找并输出该数组中的最小值, 请填空。

```
Option Base 1  
Private Sub Command1_Click()  
    Dim arr1
```

```

Dim Min As Integer, i As Integer
arr1 = Array(12, 435, 76, -24, 78, 54, 866, 43)
Min = 1
For i = 2 To 8
    If arr1(i) < Min Then 2
Next i
Print "最小值是: "; Min
End Sub

```

4. 下列程序用来在窗体上输出如下数据，请填空。

$$\begin{pmatrix} 1, 2, 3, 4, 5 \\ 2, 3, 4, 5, 1 \\ 3, 4, 5, 1, 2 \\ 4, 5, 1, 2, 3 \\ 5, 1, 2, 3, 4 \end{pmatrix}$$

```

Private Sub Form_Click()
Dim a(5, 5) As Byte, i As Byte, j As Byte
For i = 1 To 5
    For j = 1 To 6 - i
        a(i, j) = 1
    Next j
Next i
For i = 2 To 5
    For j = 2 To 5
        a(i, j) = j + i - 6
    Next j
Next i
For i = 1 To 5
    For j = 1 To 5: Print a(i, j);: Next j
    3
Next i
End Sub

```

5. 完成 10 输入数据的排序程序。

```

Private Sub Form_click()
Dim a(1 to 10) As Integer, temp As Integer
Dim i As Byte, j As Byte, k As Byte
For i=1 To 20
    a(i)=1
Next i
For i=1 To 19
    2
    For j=i+1 To 20
        If a(j)<a(k) Then 3
    Next j
    Temp=a(i): 4 : a(k)=Temp
Next i
For i=1 To 20

```

```
Print a(i);  
Next i  
Print  
End Sub
```

三、编程题

1. 求 50 个数的斐波那契(Fibonacci)数列。Fibonacci 数列问题起源于一个古典的有关兔子繁殖的问题：假设在第 1 个月时有一对小兔子；第 2 个月时小兔子成为大兔子；第 3 个月成为老兔子，并生出一对小兔子(一对老，一对小)；第 4 个月时老兔子又生出一对小兔子，上个月的小兔子变成大兔子(一对老，一对大，一对小)；第 5 个月时上个月的大兔子成为老兔子，上个月的小兔子变成大兔子，两对老兔子生出两对小兔子(两对老，一对大，两对小)……。

2. 某数组有 20 个元素，元素的值由键盘输入，要求将前 10 个元素与后 10 个元素对换。即第 1 个元素与第 20 个元素互换，第 2 个元素与第 19 个元素互换，……，第 10 个元素与第 11 个元素互换。输出数组原来各元素的值和对换后各元素的值。

3. 由键盘输入一串字符，对其排序后输出。

4. 有一个 6×6 的矩阵，各元素的值由键盘输入，求全部元素的平均值，并输出高于平均值的元素以及它们的行、列号。

5. 编程实现如图 5.4 所示的效果：文本框的背景颜色由单选按钮所组成的控件数组控制，可显示红、绿、蓝三色。也可以自行添加颜色项目。



图 5.4 编程题 5 的效果图

第 6 章 过 程

教学提示：过程是程序设计语言中的一个重要的概念，所谓过程就是完成某一特定功能的程序代码段，它由一系列可执行语句组成，是一个相对独立的实体。Visual Basic 把应用程序按功能分为多个模块。每个模块中包含实现各种操作的相互独立的过程，每个过程完成一个特定的任务。本章介绍 Sub 过程和 Function 过程的定义和使用。

教学要求：掌握 Sub 过程的定义和调用方法，掌握 Function 过程的定义和调用方法，掌握进行参数传递的方法，掌握不同变量的作用域，在编程过程中灵活运用过程。

6.1 过程的定义

6.1.1 Sub 过程

Sub 过程是指一组能够完成特定操作和相对独立的程序段(语句块)。其他过程中要执行这个操作时，可以调用 Sub 过程。

Sub 过程定义的一般格式为：

```
[Private | Public ] [ Static ] Sub 子过程名 [(参数表)]  
    局部变量或常数定义  
    语句块  
    [Exit Sub]  
    语句块  
End Sub
```

其中：

(1) Sub 为定义过程的关键字，前缀 Private、Public、Static 为可选关键字，用来说明过程的性质。

(2) Private 和 Public 用来声明该 Sub 过程是私有的(局部的)，还是公有的(全局的)；系统默认为 Public。

Public 关键字声明全局级过程，全局过程在应用程序的所有模块中都可以调用；Private 关键字声明模块级过程，只有在声明它的模块中才可以调用此过程。

(3) Static 是静态变量声明。如果定义过程时使用 Static 关键字，则该过程中的所有局部变量的存储空间只分配一次，且这些变量的值在整个程序运行期间都存在，即在每次调用该过程时，各局部变量的值一直存在。如果省略 Static，过程每次被调用时重新为其局部变量分配存储空间，当该过程结束时释放其局部变量的存储空间。

(4) 过程名与变量名的命名规则相同。在同一模块中，同一过程名不能既用于 Sub 过

程，又用于 Function 过程。

(5) 参数表类似于变量声明，它指明了调用过程传递给过程的变量个数和类型，称为形式参数(简称形参)。Sub 过程可以没有形参，也可以有一个或多个形参。当有多个形参时，各参数之间用逗号隔开。

形参声明的一般格式为：

```
[ [Optional] [ ByVal | ByRef] [ ParamArray ] <变量名> [( )] [As <类型>]
```

其中：

- **Optional** 表示参数是“可选参数”。如果参数表中某个参数声明中使用了该选项，则参数表中该参数的所有后续参数都必须都是可选的，而且必须都使用 **Optional** 关键字声明。如果使用了 **ParamArray**，则任何参数都不能使用 **Optional**。
- **ByVal** 表示该参数按值传递，**ByRef** 表示该参数按地址传递。如果没有明确指明参数的传递方式，Visual Basic 默认为 **ByRef**。
- **ParamArray** 关键字用于表示“可变参数”，只用于参数表的最后一个参数，指明最后这个参数是一个 Variant 元素的 **Optional** 数组。使用 **ParamArray** 关键字可以提供任意数目的参数。**ParamArray** 关键字不能与 **ByVal**、**ByRef** 或 **Optional** 一起使用。
- 变量名代表参数的变量的名称，遵循标准的变量命名约定。如果是数组变量，则要在数组名后加上一对小括号。

有关过程中的参数和参数的传递在 6.3 节中将详细讨论。

(6) **End Sub** 用于表示本 Sub 过程的结束。在 Sub 与 **End Sub** 之间的语句块称为过程体。在过程体内可以用 **Exit Sub** 语句强制从 Sub 过程中退出，结束过程的执行。

Sub 过程可以放在标准模块和窗体模块中，Visual Basic 的 Sub 过程分为事件过程和通用过程两大类。

事件过程是 Visual Basic 预定义好的过程，过程名和参数的类型与个数不能改变，编程人员只能填写事件过程中的语句。通用过程则是编程人员根据需要自己定义的，过程名及参数的个数和类型均可以自己设定。

1. 事件过程

事件过程是当发生某个事件时，对该事件做出响应的程序段，它是 Visual Basic 应用程序的主体。在前面几章中写的程序段就是作为事件过程而存在的。

事件过程由 Visual Basic 声明，用户不能增加或删除。当用户对某个对象发出一个动作时，Windows 会通知 Visual Basic 产生了一个事件，Visual Basic 会自动调用与该事件相关的事件过程。事件过程是附加在窗体或控件上的，通常总是处于空闲状态，直到响应用户引发的事件或系统引发的事件才被调用。

窗体事件过程定义为：Form_事件名，其一般格式为：

```
Private Sub Form_<事件名> ([参数表])
    [语句组]
End Sub
```

需要注意的是，不管窗体名字如何定义，在窗体事件过程定义中只能使用 **Form**，而在程序中对窗体进行引用才会用到窗体名字。如果正在使用 MDI(Multiple Document Interface)窗体，则事件过程定义为：MDIForm_事件名。

例如，单击窗体的事件过程为：

```
Private Sub Form_Click()  
    ' 过程体  
    ...  
End Sub
```

控件事件过程的定义为：控件名_事件名，其一般格式为：

```
Private Sub <控件名>_<事件名> ([参数表])  
    [语句组]  
End Sub
```

例如，一个命令按钮(设该命令按钮的名称为 CmdOk)的单击事件过程代码为：

```
Private Sub CmdOk_Click()  
    ' 过程体  
    ...  
End Sub
```

2. 通用过程

当几个不同的事件过程要执行同样的动作(即可能需要使用同一段程序代码)，为了不必重复编写代码，可将这段代码独立出来，编写成一个共用的过程，这种过程通常称为通用过程，它独立于事件过程之外，可供其他过程调用。通用过程既可保存在窗体模块中，也可以保存在标准模块中。

通用过程与事件过程不同，通用过程并不是由对象的某种事件激活，也不依附于某一个对象。建立通用过程有两种方法：直接在代码编辑窗口中输入过程代码或使用添加过程对话框。

例如，编写一个通用过程 **Reverse**，将一个正整数逆序输出，如 1234 输出为 4321，12300 输出为 321。

```
Private Sub Reverse(ByVal x As Integer)  
    s = 0  
    Do While x <> 0  
        r = x Mod 10  
        s = s * 10 + r  
        x = x \ 10  
    Loop  
    Print s  
End Sub
```

6.1.2 Function 过程

函数是过程的另一种形式，当过程的执行返回一个值时，使用函数就比较简单。**Visual Basic** 包含了许多内部的函数，如 **sin**、**abs**、**int**、**rnd** 等。用户在编写程序时，只需写出一个函数名并给定参数就能得出函数值。当在程序中需要多次用到某一公式或要处理某一函数关系，而又没有现成的内部函数可使用，**Visual Basic** 允许使用 **Function** 语句编写用户自定义的 **Function** 函数过程。

与 **Sub** 过程一样，**Function** 过程也是一个独立的过程，可传递参数，执行一系列语句

并改变其参数的值。与 Sub 过程不同的是，Function 过程可返回一个值到调用的过程。

Function 过程定义的一般格式为：

```
[Private | Public ] [ Static ] Function 函数过程名([参数列表]) [As 类型]
    局部变量或常数定义
    语句块
    函数过程名 = 表达式
    [Exit Function]
    语句块
    函数过程名 = 表达式
End Function
```

其中：

- (1) Function 为定义函数过程的关键字，其前缀 Private、Public、Static 为可选关键字，用来说明该函数过程的性质。
 - (2) As 类型是函数返回值的数据类型，与变量一样，如果没有 As 子句，默认的数据类型为 Variant。
 - (3) 函数过程名=表达式：在函数体中用该语句给函数赋值，在关键字 Function 与 End Function 之间的函数体中，应该至少有一条给函数过程名赋值的语句。最后一次给函数过程名赋的值就是该函数过程的返回值。如果在 Function 过程中没有给函数名赋值，则该 Function 过程的返回值为数据类型的默认值。例如，数值函数返回值为 0，字符串函数返回值为空字符串，可变类型(Variant)函数返回值为空值。
 - (4) Exit Function 语句用于提前从 Function 过程中退出，程序接着从调用该 Function 过程语句的下一条语句继续执行。在 Function 过程的任何位置都可以有 Exit Function 语句。但用户退出函数之前，应该有给函数过程名赋值的语句被执行。
- 例如，下面是一个计算三角形面积的函数过程，函数名为 TriArea、函数返回值的数据类型为 Single，函数的形参为三角形三条边的边长 a、b 和 c，返回值为该三角形的面积。Function 过程代码如下：

```
Private Function TriArea(ByVal a As Single, ByVal b As Single, ByVal c As Single)
    Dim p As Single, s As Single
    p = (a + b + c) / 2
    s = Sqr(p * (p - a) * (p - b) * (p - c))
    TriArea = s
End Function
```

与 Sub 过程一样，Function 过程既可以直接在代码编辑窗口中输入过程代码，也可以使用添加过程对话框来定义 Function 过程。由于函数过程具有返回值，还需对系统自动产生的函数过程模板进行适当的修改，即在过程名后面加上对其返回值类型的定义和说明。

6.2 过程的调用

一个过程定义后，必须通过调用才能执行。事件过程通过发生相关事件调用或用语句

调用，用户定义的 Sub 通用过程或 Function 函数过程通过语句调用。

6.2.1 调用 Sub 事件过程

Sub 事件过程可有事件自动调用，也可以被在同一模块中的其他过程使用调用语句调用。

用调用语句调用 Sub 过程有两种方式：使用 Call 语句和直接使用 Sub 过程名。其一般格式为：

```
Call 过程名[(参数列表)]  
或者  
过程名 [参数列表]
```

其中：

(1) 参数列表中，在调用语句中的参数称为实在参数(简称实参)。实参可以是常量、变量、表达式和数组。

(2) 使用 Call 语句调用时，参数必须在括号内，当被调用过程没有参数时，则括号也可省略；用过程名调用时，参数列表不加括号。

(3) 执行调用语句时，Visual Basic 将控制转移到被调用的 Sub 过程。

【例 6.1】 检查文本框中输入的数据是否是数值。

在文本框(txtInput)中输入数据，在文本框失去焦点时检查输入的数据是否为数值，在按钮“检查”(cmdCheck)的单击事件中也进行检查。

程序代码如下：

```
Private Sub txtInput_LostFocus()  
    If IsNumeric(txtInput.Text) = True Then  
        MsgBox ("输入的是数值")  
    Else  
        MsgBox ("输入的是文字")  
    End If  
End Sub  
  
Private Sub cmdCheck_Click()  
    Call txtInput_LostFocus ' 调用事件过程  
End Sub
```

6.2.2 调用 Sub 通用过程

调用 Sub 通用过程的语法和调用 Sub 事件过程的相同。不同的是，Sub 通用过程只有通过语句调用后才被执行，否则不被执行。

【例 6.2】 编写一个 Sub 过程 Divisor 输出两个正整数的最大公约数。单击窗体时，通过 InputBox 输入两个正整数，然后调用过程 Divisor 输出结果。

求两个正整数的最大公约数采用辗转相除法，算法如下：

- (1) 输入两个正整数 m, n;
- (2) 计算 m 除以 n 的余数 r;
- (3) 若 r=0, 则转步骤(6);

- (4) 用 n 替换 $m(m=n)$, 用 r 替换 $n(n=r)$;
 (5) 转步骤(2);
 (6) n 的值就是 m 、 n 两个数的最大公约数。

程序代码如下:

```
Private Sub Form_Click()
    Dim x%, y%
    x = Val(InputBox("请输入一个正整数"))
    y = Val(InputBox("请再输入一个正整数"))
    Print x; "和"; y; "的";
    Call Divisor(x, y) '调用通用 Sub 过程
    x = Val(InputBox("请输入一个正整数"))
    y = Val(InputBox("请再输入一个正整数"))
    Print x; "和"; y; "的";
    Divisor x, y '调用通用 Sub 过程
End Sub
Private Sub Divisor(ByVal m As Integer, ByVal n As Integer)
    Dim r As Integer
    r = m Mod n
    Do While r <> 0
        m = n
        n = r
        r = m Mod n
    Loop
    Print "最大公约数为 "; n
End Sub
```

6.2.3 调用 Function 过程

调用 Function 函数过程的方法与调用 Visual Basic 内部函数的方法一样(如 Int(x)), 在语句中直接使用函数名, 其一般格式为:

函数名([参数列表])

【例 6.3】 输入三角形的三条边的边长, 求三角形的面积。

```
Private Function TriArea(ByVal a As Single, ByVal b As Single, ByVal c As Single)
    Dim p As Single, s As Single
    p = (a + b + c) / 2
    s = Sqr(p * (p - a) * (p - b) * (p - c))
    TriArea = s
End Function
Private Sub Form_Click()
    Dim x As Single, y As Single, z As Single
    x = Val(InputBox("请输入三角形第一条边的边长"))
    y = Val(InputBox("请输入三角形第二条边的边长"))
    z = Val(InputBox("请输入三角形第三条边的边长"))
    If x > 0 And y > 0 And z > 0 Then
        If x + y > z And x + z > y And y + z > x Then
            s = TriArea(x, y, z) '调用函数过程求三角形的面积
```

```
        MsgBox ("三角形的面积为" & s)
    Else
        MsgBox ("输入的三个数, 不能组成一个三角形!")
    End If
Else
    MsgBox ("输入的数据必须全部大于 0")
End If
End Sub
```

6.3 参数的传递

参数是过程与外部程序传输信息的纽带。外部程序调用过程时可以把数据传递给过程, 也可以把过程中的数据传递回来。在调用过程时, 需考虑调用过程和被调用过程之间的数据是如何传递的。通常在编制一个过程时, 应考虑需要输入哪些数据, 进行处理后输出哪些数据; 正确地为过程提供输入数据和正确地引用其输出数据, 是使用过程的关键, 也就是调用过程和被调用过程之间的数据传递问题。

在调用一个有参数的过程时, 首先进行的是形参与实参的结合, 实现调用过程的实参与被调用过程的形参之间的数据传递。数据传递有两种方式: 按值传递和按地址传递。

6.3.1 形参和实参

在过程定义的参数表中出现的参数称为形式参数(简称形参)。在过程被调用之前, 形参并未被分配内存, 只是说明形参的类型和在过程中的作用。形参列表中的各参数之间用逗号分隔, 形参可以是变量名和数组名。

在调用过程语句或表达式中出现的参数表称为实在参数(简称实参), 在过程调用时实参将数据传递给形参。实参表可由常量、表达式、有效的变量名、数组名组成, 实参表中各参数用逗号分隔。

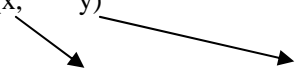
在调用一个过程时, 必须把实参传送给过程中的形参, 完成形参与实参的结合, 这种参数的传递也称为参数的结合。例如, 定义一个过程:

```
Private Sub Divisor(ByVal m As Integer, ByVal n As Integer)
...
End Sub
```

则其形参与实参的结合关系如下:

过程调用: Call Divisor (x, y)

过程定义: Sub Divisor (ByVal m As Integer, ByVal n As Integer)



在传递参数时, 形参表与实参表中对应参数的名字可以不同, 但要求形参表与实参表中参数的个数、数据类型、位置顺序必须一一对应。因为在调用过程时, 形参和实参结合是按位置结合的, 即第一个实参与第一个形参结合, 第二个实参与第二个形参结合, 依次类推。

6.3.2 参数按值传递和按地址传递

Visual Basic 中传递参数的方式有两种：一种是按值传递，另一种是按地址传递。在过程定义时，形参前加“ByVal”关键字的是按值传递，形参前加“ByRef”关键字(或省略)的是按地址传递。

1. 按值传递参数

按值传递参数时，Visual Basic 给传递的形参分配一个临时的内存单元，将实参的值传递到这个临时单元中去。实参向形参传递是单向的，如果在被调用的过程中改变了形参值，则只是临时单元的值变动，不会影响实参变量本身。当被调用过程结束返回调用过程时，Visual Basic 将释放给形参分配的临时内存单元，实参变量的值保持不变。

如果调用语句中的实参是常量或表达式，此时参数传递是按值传递。

【例 6.4】 按值传递参数示例。

```
Private Sub ChangeByVal(ByVal x As Integer, ByVal y As Integer)
    Dim t As Integer
    t = x
    x = y
    y = t
End Sub

Private Sub Form_Click()
    Dim a As Integer, b As Integer
    a = 10
    b = 20
    Print "交换前, a="; a; " b="; b
    Call ChangeByVal(a, b)
    Print "交换后, a="; a; " b="; b
End Sub
```

程序运行时，单击窗体，在窗体中输出结果为：

交换前, a=10 b=20

交换后, a=10 b=20

从输出结果可以看到，调用过程 ChangeByVal 没能交换变量的值，其原因是：过程 ChangeByVal 采用按值传递形参，过程被调用时系统给形参 x 和 y 分配临时内存单元，将实参 a 和 b 的值分别传递(赋值)给 x 和 y，在过程 ChangeByVal 中，变量 a、b 不可使用，x、y 通过临时变量 t 实现交换，调用结束返回调用过程后形参 x、y 的临时内存单元将释放，实参单元 a 和 b 仍保留原值。由此可以看出，按值传递时，参数的传递是单向的。

2. 按地址传递参数

按地址传递参数是指实参变量的内存地址传递给形参，使形参和实参具有相同的地址，这意味着形参和实参共享一段存储单元。因此，在被调用过程中改变形参的值，则相应实参的值也被改变，也就是说，与按值传递参数不同，按地址传递参数可以在被调用过程中改变实参的值。系统默认情况下是按地址传递参数的。

【例 6.5】 按地址传递参数示例。

```
Private Sub ChangeByRef(ByRef x As Integer, ByRef y As Integer)
    Dim t As Integer
    t = x
    x = y
    y = t
End Sub

Private Sub Form_Click()
    Dim a As Integer, b As Integer
    a = 10
    b = 20
    Print "交换前, a="; a; " b="; b
    Call ChangeByRef(a, b)
    Print "交换后, a="; a; " b="; b
End Sub
```

程序运行时，单击窗体，在窗体中的输出结果为：

交换前，a=10 b=20

交换后，a=20 b=10

从输出结果可以看到，调用过程 **ChangeByRef** 交换了变量的值，其原因是：过程 **ChangeByRef** 采用按地址传递形参，当调用过程 **ChangeByRef** 时，通过虚实结合，形参 **x**、**y** 获得实参 **a**、**b** 的地址，即 **x** 和 **a** 使用同一存储单元，**y** 和 **b** 使用同一存储单元。因此，在被调用过程 **ChangeByRef** 中 **x**、**y** 通过临时变量 **t** 实现交换后，实参 **a** 和 **b** 的值也同样被交换，当调用结束返回后，**x**、**y** 被释放，实参 **a**、**b** 的值就是交换后的值。

需要说明的是，并不是所有 **ByRef** 关键字修饰的形参在过程实际调用时一定是按地址传递的。只有当实参是变量名或数组名时才能实现按地址传递。如果实参是常量或表达式，实际进行的是按值传递。如果希望强制以单个变量为实参进行按值传递，可以给这个实参变量加上一个额外的小括号，这样 **Visual Basic** 就把它理解为一个表达式，实行按值传递。

例如，若将例 6-5 中的调用语句“**Call ChangeByRef(a, b)**”改为

```
Call ChangeByRef((a), b)
```

则程序运行后，单击窗体，运行结果为：

交换前，a=10 b=20

交换后，a=10 b=10

这是由于在调用时，实参变量“**a**”用括号括起来了，“**(a)**”就成为一个表达式了，实行按值传递；而实参变量 **b** 按地址传递，因此调用过程 **ChangeByRef((a), b)** 后，**a** 的值保持不变，**b** 的值发生了改变。

6.3.3 数组参数

Visual Basic 允许把数组作为一个实参传递给过程，此时要求在定义过程时相应的形参应加空括号表明是数组。调用时，相应的实参必须是数组，可以只写数组名，不必加括号。数组作参数时必须按地址传递的，形参与实参共用同一段内存单元，不能使用 **ByVal** 关

键字修饰。形参数组与实参数组的数据类型应一致。

【例 6.6】 10 个小孩围坐一圈分糖果，开始时，老师随机分给每位小孩若干糖果，如果哪位小孩的糖果数为一个奇数，向老师补要 1 块。为了公平，现进行调整，调整规则是：所有小孩同时把自己糖果的一半分给左边的小孩，糖的块数变为奇数的小孩向老师补要一块（设老师手中的糖果足以满足这些要求）。问经过多少次调整，大家的糖果数都一样？每人多少块？

程序代码如下：

```
Function IsEqual(a() As Integer)
    start = LBound(a)
    For i = start + 1 To UBound(a)
        If a(i) <> a(start) Then
            IsEqual = 0
            Exit Function
        End If
    Next i
    IsEqual = 1
End Function

Private Sub Exchange(a() As Integer)
    temp = a(UBound(a))
    For i = UBound(a) To LBound(a) + 1 Step -1
        a(i) = a(i) / 2 + a(i - 1) / 2
    Next i
    a(LBound(a)) = a(LBound(a)) / 2 + temp / 2
End Sub

Private Sub Supply(a() As Integer)
    For i = LBound(a) To UBound(a)
        If a(i) Mod 2 <> 0 Then
            a(i) = a(i) + 1
        End If
    Next i
End Sub

Private Sub Form_Click()
    Dim sugars(1 To 10) As Integer
    Dim ExCount As Integer
    ExCount = 0
    For i = 1 To 10
        sugars(i) = Val(InputBox("请输入第" & i & "个小孩的糖果数："))
    Next i
    Do While IsEqual(sugars) <> 1
        Call Supply(sugars)
        Call Exchange(sugars)
        ExCount = ExCount + 1
    Loop
    Print "经过"; ExCount; "次调整后，所有小孩的糖果数均相同。"
    Print "每个小孩的糖果数为"; sugars(1)
End Sub
```

程序中定义了 3 个过程，这 3 个过程的形参均为一个一维数组的数组名。其中，过程 `IsEqual` 用来判断数组中所有元素的值是否都相等，如果都相等则返回 1，否则返回 0；过程 `Exchange` 用来完成一次调整过程，所有小孩同时把自己糖果的一半分给左边的小孩；过程 `Supply` 用来完成补发操作，即糖的块数为奇数的小孩向老师补要一块。在窗体单击事件过程 `Form_Click()` 中用一维数组 `sugars` 来保存 10 个小孩的糖果数，再调用过程 `Exchange` 或过程 `Supply`，将数组 `sugars` 的首地址送给形参数组 `a`，因此数组 `a` 和数组 `sugars` 共享一段内存空间，在过程中对数组 `a` 中元素值进行修改，则数组 `sugars` 对应的元素值也被修改了。

由上例可以看出，数组作为参数时，应注意以下几点。

(1) 为了把一个数组的全部元素传送给一个过程，应将数组名写入形参表中，并略去数组的上下界，但括号不能省略。例如，形参“`a()`”即为数组。

(2) 被调用过程中可通过 `Lbound` 和 `Ubound` 函数确定实参数组的上、下界。

(3) 当用数组作形参时，对应的实参必须也是数组，且类型一致。

(4) 实参和形参结合是按地址传递的，即形参数组和实参数组共用一段内存单元。

6.3.4 可选参数与可变参数

在前面介绍过程的定义时提到过，在形参的前面除了可以加关键字 `ByVal` 或 `ByRef` 用于指定参数传递的方式外，形参前面还可以加关键字 `Optional` 或 `ParamArray`。`Optional` 关键字修饰的参数称为可选参数，`ParamArray` 关键字修饰的参数称为可变参数。

1. 可选参数

一般来说，一个过程在声明时定义了几个形参，则在调用这个过程时必须使用相同数量的实参。`Visual Basic` 允许在形参前面使用 `Optional` 关键字把它设定为“可选参数”。如果一个过程的某个形参为可选参数，则在调用此过程时可以不提供对应于这个形参的实参。

如果在过程定义的形参表中用 `Optional` 关键字将某个参数指定为可选参数，则参数表中此参数后面的所有其他参数也必须是可选的，并都要用 `Optional` 来修饰。

【例 6.7】 编写一个计算两个数据的乘积的函数过程，它能可选择地乘以第 3 个数。在调用时，既可以给它传递 2 个参数，也可给它传递 3 个参数。为了定义带可选参数的过程，必须在参数表中使用 `Optional` 关键字，并在过程体中通过 `IsMissing` 函数来测试调用时是否传递可选参数。当调用时，提供了可选参数，则 `IsMissing` 函数的返回值为 `False`，如果没有提供可选参数，则 `IsMissing` 函数返回值为 `True`。过程代码如下：

```
Private Function multi(ByVal a%, ByVal b%, Optional ByVal c)
    p = a * b
    If Not IsMissing(c) Then
        p = p * c
    End If
    multi = p
End Function
```

在调用上面的过程时，可以提供 2 个参数，也可提供 3 个参数，都能得到正确的结果。例如，可用下述事件过程调用：

```
Private Sub Command1_Click()  
    Print multi(4, 5)      ' 提供 2 个参数  
    Print multi(1, 2, 3)   ' 提供 3 个参数  
End Sub
```

另外，编程人员也可以给可选参数指定默认值，这样在调用时，未提供实参的形参在调用时被赋以形参类型的默认值。例如，将上面的 Multi 函数过程改写如下：

```
Private Function multi(ByVal a%, ByVal b%, Optional c As Variant = 1)  
    p = a * b * c  
    multi = p  
End Function
```

当调用 multi(4, 5) 时，没有提供形参 c 对应的实参，则 c 的值取给定的默认值 1。

2. 可变参数

如果一个过程的最后一个参数是使用“ParamArray”关键字声明的数组，则这个过程在被调用时可以接受任意多个实参。调用这个过程时使用的多个实参值均按顺序存于这个数组中。ParamArray 关键字不能与 Byval、ByRef 或 Optional 关键字针对同一个形参一起使用。使用 ParamArray 关键字修饰的参数只能是 Variant 类型，一个过程只能有一个这样的形参。当有多个形参时，ParamArray 修饰的形参必须是形参表中的最后一个。

【例 6.8】 定义一个可变参数过程，用这个过程可以求任意多个数的乘积。

```
Private Sub Multi(ParamArray Numbers())  
    p = 1  
    For Each x In Numbers  
        p = p * x  
    Next x  
    Print p  
End Sub
```

可以用任意个参数调用上述过程：

```
Private Sub Command1_Click()  
    Call Multi(4, 5)      ' 提供 2 个参数  
    Call Multi(1, 2, 3)   ' 提供 3 个参数  
    Multi 2, 3, 4, 5, 6    ' 提供 5 个参数  
End Sub
```

6.3.5 对象参数

在声明通用过程时，可以使用 Object、Control、Label、TextBox 等关键字把形参定义为对象类型。调用具有对象类型形参的过程时，应该给该形参提供类型相匹配的对象名作为实参。这时，传递给过程的就是该对象的引用，在过程中可以存取其属性和调用其方法。

【例 6.9】 对文本框中输入的数据进行检查。

一个界面中有若干个文本框用于输入数据，现要求文本框中输入的数据均为数值型的，

因此, 在文本框的失去焦点(LostFocus)事件中, 要检查输入数据是否是数值型的, 如果不是数值型的, 则出现 MsgBox 对话框提示出错, 并且用 SetFocus 方法将焦点移回文本框。

显然, 在每个文本框的失去焦点事件中, 均需要编写进行数据检查的代码, 可以编写一个通用过程 txtCheck 进行数据检查, 在文本框的失去焦点事件过程中调用这个通用过程。

编写的 txtCheck 通用过程, 将文本框作为参数, 调用这个通用过程时, 将文本框的对象名作为实参传递给形参, 从而使得同一个 txtCheck 通用过程可以对不同文本框进行检查。

程序示例代码如下:

```
' 定义文本框输入数据检查的通用过程
Private Sub txtCheck(txt As TextBox)
    If IsNumeric(txt.Text) = True Then
        txtNumber = Val(txt.Text)
    Else
        MsgBox ("输入数据出错! ")
        txt.SetFocus
    End If
End Sub
Private Sub Text1_LostFocus()
    Call txtCheck(Text1)           ' 调用数据检查通用过程
End Sub
Private Sub Text2_LostFocus()
    Call txtCheck(Text2)           ' 调用数据检查通用过程
End Sub
```

6.4 递归过程

递归就是某一事物直接地或间接地由自己组成。一个过程直接或间接地调用自身, 便构成了过程的递归调用, 前者称为直接递归调用, 后者称为间接递归调用。包含递归调用的过程称为递归过程。

例如, 下面的过程 MySub1()就是直接递归的例子。

```
Private Sub MySub1()
    ...
    Call MySub1           ' 调用自身
    ...
End Sub
```

下面的过程 OneSub()是间接递归的例子。

```
' 定义过程 OneSub()
Private Sub OneSub()
    ...
    Call TwoSub           ' 调用 TwoSub 过程
    ...
End Sub
' 定义过程 TwoSub()
```

```

Private Sub TwoSub()
...
Call OneSub      '调用 OneSub 过程
...
End Sub

```

【例 6.10】 编程计算 $n!$ 。

根据数学知识, 负数的阶乘没有定义, 0 的阶乘为 1, 正数 n 的阶乘为 $n(n-1)(n-2)\times\cdots\times 2\times 1$

可以用下式表示:

$$n! = \begin{cases} 1 & (n=0,1) \\ n(n-1)! & (n>1) \end{cases}$$

利用此式, 求 n 的阶乘可以转换为求 $n(n-1)!$ 。

在 Visual Basic 中, 可以用递归过程实现上述运算。程序如下:

```

Function fact(ByVal n As Integer)
If n < 0 Then
    MsgBox ("Data Error!")
    fact = -1
    Exit Function
End If
If n = 0 Or n = 1 Then
    fact = 1
Else
    fact = n * fact(n - 1)
End If
End Function
Private Sub Form_Click()
    Dim x As Integer
    x = Val(InputBox("请输入一个非负整数"))
    If fact(x) <> -1 Then
        Print x; "的阶乘为"; fact(x)
    End If
End Sub

```

在函数过程 fact 中, 当形参 n 的值大于 1 时, 函数的返回值为 $n*\text{fact}(n-1)$, 其中 $\text{fact}(n-1)$ 又是一次函数过程调用, 而调用的又是 fact 函数。这就是在一个函数过程中调用自身的情况, 即过程的递归调用。下面进一步讨论函数过程 fact 的调用过程。

例如当 $n=5$ 时, $\text{fact}(5)$ 的返回值是 $5*\text{fact}(4)$, 而 $\text{fact}(4)$ 的返回值是 $4*\text{fact}(3)$, 仍然是个未知数, 还要先求出 $\text{fact}(3)$, 而 $\text{fact}(3)$ 也不知道, 它的返回值为 $3*\text{fact}(2)$, 而 $\text{fact}(2)$ 的值为 $2*\text{fact}(1)$, 现在 $\text{fact}(1)$ 的返回值为 1, 是一个已知数。然后回头根据 $\text{fact}(1)$ 求出 $\text{fact}(2)$, 将 $\text{fact}(2)$ 的值乘以 3, 求出 $\text{fact}(3)$, 将 $\text{fact}(3)$ 乘以 4, 得到 $\text{fact}(4)$, 再将 $\text{fact}(4)$ 乘以 5, 得到 $\text{fact}(5)$ 。这就是说, 递归过程在执行时, 将引起一系列的调用和回代的过程。当 $n=5$ 时 fact 的调用和回代过程如图 6.1 所示。

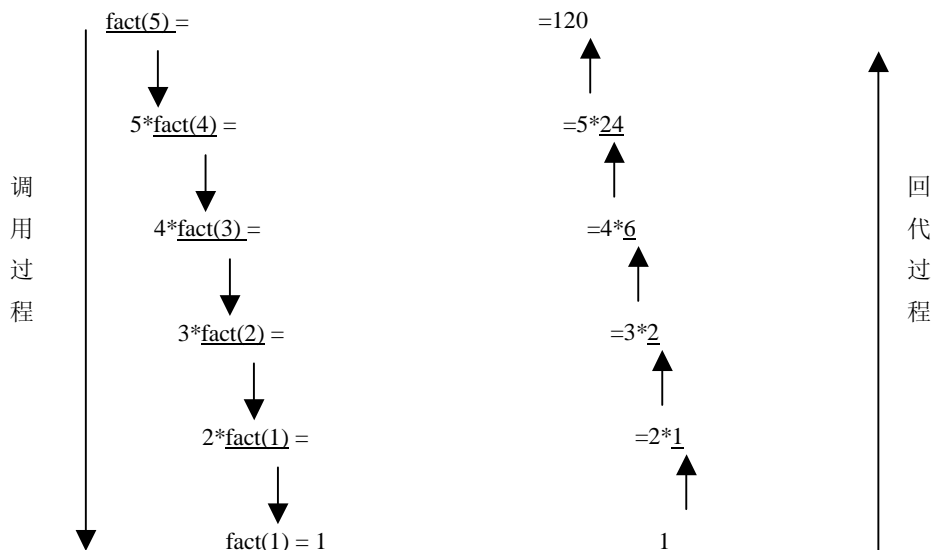


图 6.1 递归函数的调用和回代过程

从图 6.1 可以看出，递归过程不应无限制地进行下去，当调用若干次以后，就应当到达递归调用的终点得到一个确定值(例如本程序中的 $\text{fact}(1)=1$)，然后进行回代，回代的过程是从一个已知值推出下一个值。实际上这是一个递推过程。

在设计递归过程时应当考虑到递归的终止条件，在本例中，使递归终止的条件是：

```
If n = 0 Or n = 1 Then
    fact = 1
```

所以，任何有意义的递归总是由两部分组成的：递归方式与递归终止条件。

递归是一种非常有用的程序设计技术。当一个问题蕴含递归关系且结构比较复杂时，采用递归算法往往比较自然、简捷，而且容易理解。

【例 6.11】 用递归求解汉诺塔(Tower of Hanoi)问题。在 19 世纪，一种称为汉诺塔的游戏在欧洲广为流行。游戏的装置是一块铜板，上面有 3 根柱子，左柱自下而上、由大到小顺序串有 64 个金盘，呈塔形(如图 6.2 所示)。游戏的目标是把左柱上的金盘全部移到右边的柱子上。条件是，一次只能移动一个金盘，被移动的盘子必须放在其中的一根柱子上，并且不允许大盘在小盘上面。编写一个程序，打印将盘子从第 1 根柱子移到第 3 根柱子的移动次序。

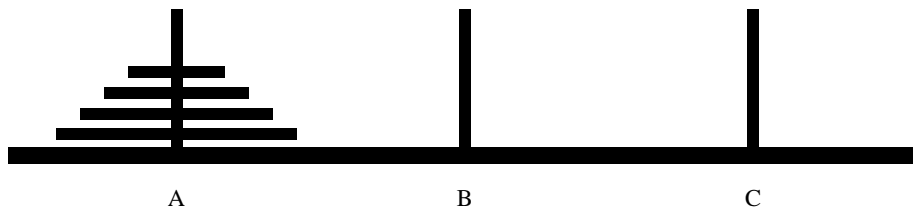


图 6.2 汉诺塔问题

假定要把 n 个盘子按规定由 a 柱借助 b 柱移到 c 柱。模拟这一过程的算法称为 $\text{hanoi}(n, a, b, c)$ ，那么，很自然的想法是：

- (1) 先把 $n - 1$ 个盘子设法借助 c 柱移到 b 柱上，记做 $\text{hanoi}(n-1, a, c, b)$ 。
- (2) 把第 n 个盘子从 a 柱移到 c 柱。
- (3) 把 b 柱上的 $n-1$ 个盘子借助 a 柱移到 c 柱，记做 $\text{hanoi}(n-1, b, a, c)$ 。

根据以上分析，编写程序如下：

```
Private Sub Hanoi(ByVal n%, ByVal A$, ByVal B$, ByVal C$)
    If n = 1 Then
        Print A; "->"; C
    Else
        Hanoi n - 1, A, C, B
        Print A; "->"; C
        Hanoi n - 1, B, A, C
    End If
End Sub
Private Sub Form_Click()
    Dim x As Integer
    x = Val(InputBox("Please input the number of disks to be moves: "))
    Hanoi x, "A", "B", "C"
End Sub
```

6.5 变量的作用域与生存期

Visual Basic 的程序模块由一些过程组成，在过程中会使用到变量。变量的定义位置不同，定义方式不同，允许被访问的范围和作用时间也不相同。变量的作用域即指变量的有效范围。变量的生存期即指变量的作用时间。

6.5.1 变量的作用域

变量的作用域决定了应用程序中哪些过程可以访问该变量。按变量的作用域不同，可以将变量分为局部变量、窗体/模块级变量和全局变量。

1. 局部变量

局部变量指在过程内用 **Dim** 语句声明的变量、未声明而直接使用的变量以及用 **Static** 声明的变量。这种变量只能在本过程中使用，不能被其他过程访问。在其他过程中如果有同名的变量，也与本过程的变量无关，即不同的过程中可以使用同名的变量。除了用 **Static** 声明的变量外，局部变量在其所在的过程每次运行时都被初始化。

2. 模块级变量

模块级变量指在窗体模块或标准模块的通用声明段中用 **Dim** 语句或 **Private** 语句声明的变量。模块级变量的作用范围为其定义位置所在的模块，可以被本模块中的所有过程访问。模块级变量在其所在的模块运行时被初始化。

如果在一个模块中的几个过程中均需要访问某些变量，则需要将这些变量定义为模块级变量。

【例 6.12】 一个猜数字小游戏。

在这个游戏的界面窗体中有 4 个标签、3 个文本框和 2 个命令按钮，运行时，单击【开始游戏】命令按钮后，计算机随机产生一个 1~100 之间的正整数，用户在文本框 TxtMyGuess 中输入自己猜测的数后，单击【猜】命令按钮提交，计算机给出猜测结果提示，并在文本框 TxtGuessCount 和 TxtGuessList 中记录猜测的次数和猜测情况，猜测正确后，不再让用户猜测，只有重新开始游戏(单击【开始游戏】命令按钮)后才可猜测。窗体的运行界面如图 6.3 所示。

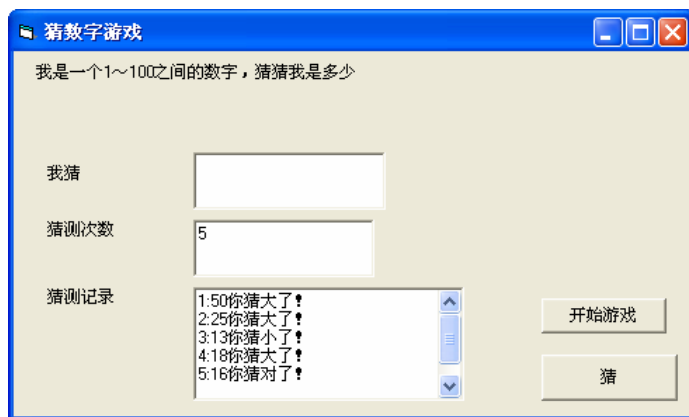


图 6.3 猜数字游戏

窗体中的控件属性设置如表 6-1 所示。

表 6-1 窗体中控件属性表

对 象	对 象 名	属 性 名	属 性 值
Form	GuessNumber	Caption	猜数字游戏
TextBox	TxtMyGuess	Text	空
	TxtGuessCount	Text	空
	TxtGuessList	Text	空
		MultiLine	True
		ScrollBars	2-Vertical
CommandButton	cmdGuess	Caption	开始游戏
	cmdStart	Caption	猜
Label	LabPrompt	Caption	我是一个 1~100 之间的数字，猜猜我是多少
	LabMyGuess	Caption	我猜
	LabGuessCount	Caption	猜测次数
	LabGuessList	Caption	猜测记录

程序代码如下：

```

Dim guessnumber, GuessCount As Integer      '模块级变量
Dim guessFlag As Boolean                    '模块级变量
Dim GuessList As String                    '模块级变量
Private Sub Form_Load()
    guessFalg = False
    Randomize
End Sub
Private Sub cmdStart_Click()
    guessnumber = Int(100 * Rnd + 1)
    guessFlag = True
    GuessCount = 0
    TxtGuessCount.Text = ""
    TxtMyGuess.text = ""
    TxtMyGuess.SetFocus
    GuessList = ""
    TxtGuessList.Text = ""
    cmdStart.Enabled = False
End Sub
Private Sub cmdGuess_Click()
    If (guessFlag) Then
        GuessCount = GuessCount + 1
        x = Val(TxtMyGuess.Text)
        If (x < guessnumber) Then
            result = "你猜小了！"
        ElseIf (x = guessnumber) Then
            result = "你猜对了！"
            guessFlag = False
            cmdStart.Enabled = True
        Else
            result = "你猜大了！"
        End If
        MsgBox (result)
        GuessList = GuessList & GuessCount & ":" & x & result & Chr(13) & Chr(10)
        TxtGuessList.Text = GuessList
        TxtGuessCount.Text = GuessCount
        TxtMyGuess.Text = ""
        TxtMyGuess.SetFocus
    Else
        MsgBox ("请开始游戏")
    End If
End Sub

```

程序中定义了 3 个事件过程 Form_Load、cmdStart_Click 和 cmdGuess_Click，在这 3 个过程中，需要对计算机产生的随机数 guessnumber、用户猜测次数 GuessCount、用户猜测情况 GuessList 以及是否可以进行猜测 guessFlag 等变量进行操作，因此需要将它们定义为模块级变量，在这 3 个过程中均可以对这些变量进行操作。

3. 全局变量

全局变量指在模块的通用声明段中用 **Public** 语句声明的变量，其作用范围为应用程序的所有过程。全局变量在应用程序运行时被初始化。

6.5.2 变量的生存期

当一个过程被调用时，系统将给该过程中的变量分配存储单元，当该过程执行结束时，是释放还是保留变量的存储单元，这就是变量的生存期问题。根据变量的生存期，可以将变量分为动态变量和静态变量。

1. 动态变量

在应用程序中的变量如果不使用 **Static** 语句进行声明，则属于动态变量。

对于过程级的动态变量，在程序运行到变量所在的过程时，系统为变量分配存储空间，并进行变量的初始化工作；当该过程结束时，释放变量所占用的存储空间，其值不再存在。模块级动态变量在运行模块时被初始化，在退出模块时释放其所占用的存储空间。全局级动态变量在应用程序执行时分配存储空间，在退出应用程序时释放存储空间。

2. 静态变量

如果一个变量用 **Static** 语句声明，则该变量只被初始化一次，在应用程序运行期间保留其值，即在每次调用该变量所在的过程时，该变量不会被更新初始化，而在退出变量所在的过程时，不释放该变量所占的存储空间。

在 **Sub** 过程、**Function** 过程的定义语句中使用 **Static** 修饰词，表明该过程内所有的变量均为静态变量。

【例 6.13】静态变量的使用示例。

在窗体中放置一个命令按钮 **Command1**，每次单击该命令按钮后，弹出一个对话框，显示“这是第几次单击命令按钮！”这一信息。

程序代码如下：

```
Private Sub Command1_Click()  
    Static num As Integer          ' 定义 num 为静态变量  
    num = num + 1  
    MsgBox ("这是第" & num & "次单击命令按钮!")  
End Sub
```

运行程序时，会发现每次单击命令按钮后，会在对话框中显示是第几次单击。如果将 **num** 不定义为静态变量，则每次单击命令按钮后，对话框中显示的信息均为“这是第 1 次单击命令按钮！”。这是由于将 **num** 定义为静态变量后，**num** 在应用程序运行期间分配一个固定存储单元，定义 **num** 变量的过程结束时，**num** 的值仍然存在，该过程再次被调用时，**num** 的值为上次过程调用结束时的值。

6.6 小 结

过程是程序设计语言中的一个重要的概念，所谓过程就是完成某一特定功能的程序代码段，它由一系列可执行语句组成，是一个相对独立的实体。

将应用程序划分为多个过程，可带来如下好处：

(1) 把一个复杂的任务划分为小任务，用过程来表达，使任务更容易理解，容易实现和维护；

(2) 过程可以被反复调用，从而避免了重复编码，使程序结构简洁。

在 Visual Basic 中用户可以定义 4 种类型的过程。

- 以关键字 Sub 开头的子过程；
- 以关键字 Function 开头的函数过程；
- 以关键字 Property 开头的属性过程；
- 以关键字 Event 开头的事件过程。

本章主要介绍了 Sub 过程和 Function 过程。

Sub 过程是指一组能够完成特定操作和相对独立的程序段(语句块)。在其他过程中要执行这个操作时，可以调用 Sub 过程。

Sub 过程定义的一般格式为：

```
[Private | Public ] [ Static ] Sub 子过程名 [(参数表)]  
    局部变量或常数定义  
    语句块  
    [Exit Sub]  
    语句块  
End Sub
```

Visual Basic 允许使用 Function 语句编写用户自定义的 Function 函数过程。Function 过程可返回一个值到调用的过程。

Function 过程定义的一般格式为：

```
[Private | Public ] [ Static ] Function 函数过程名 [(参数列表)] [As 类型]  
    局部变量或常数定义  
    语句块  
    函数过程名 = 返回值  
    [Exit Function]  
    语句块  
    函数过程名 = 返回值  
End Function
```

用调用语句调用 Sub 过程有两种方式：使用 Call 语句、直接使用 Sub 过程名。其一般格式为：

```
Call 过程名 [(参数列表)]
```

或者

```
过程名 [参数列表]
```


调用 Function 函数过程的方法与调用 Visual Basic 内部函数的方法一样，在语句中直接使用函数名，其一般格式为：

函数名([参数列表])

在调用一个有参数的过程时，首先进行的是形参与实参的结合，实现调用过程的实参与被调用过程的形参之间的数据传递。数据传递有两种方式：按值传递和按地址传递。在过程定义时，形参前有“ByVal”关键字的是按值传递，形参前有“ByRef”关键字(或省略)的是按地址传递。

Visual Basic 的程序模块由一些过程组成，在过程中会使用变量。变量的定义位置不同，定义方式不同，允许被访问的范围和作用时间也不相同。

变量的作用域决定了应用程序中哪些过程可以访问该变量。按变量的作用域不同，可以将变量分为局部变量、窗体/模块级变量和全局变量。

当一个过程被调用时，系统将给该过程中的变量分配存储单元，当该过程执行结束时，是释放还是保留变量的存储单元，这就是变量的生存期问题。根据变量的生存期，可以将变量分为动态变量和静态变量。

6.7 习 题

一、选择题

1. 以下关于函数过程的叙述中，正确的是()。
 - A) 函数过程形参的类型与函数返回值的类型没有关系
 - B) 在函数过程中，过程的返回值可以有多个
 - C) 当数组作为函数过程的参数时，既能以按值方式传递，也能以按址方式传递
 - D) 如果不指明函数过程参数的类型，则该参数没有数据类型
2. 以下关于过程及过程参数的描述中，错误的是()。
 - A) 过程的参数可以是控件名称
 - B) 用数组作为过程的参数时，使用的是“传地址”方式
 - C) 只有函数过程能够将过程中处理的信息传回调用的程序中
 - D) 窗体可以作为过程的参数
3. 以下关于变量作用域的叙述中，正确的是()。
 - A) 窗体中凡被声明为 Private 的变量只能在某个指定的过程中使用
 - B) 全局变量必须在标准模块中声明
 - C) 模块级变量只能用 Private 关键字声明
 - D) Static 类型变量的作用域是它所在的窗体或模块文件
4. Sub 过程与 Function 过程最根本的区别是()。
 - A) Sub 过程可以使用 Call 语句或直接使用过程名调用，而 Function 过程不可以
 - B) Function 过程可以有多个，Sub 过程不可以
 - C) 两种过程参数的传递方式不同

D) Sub 过程不能返回值, 而 Function 过程能返回值

5. 若希望在离开某过程后, 还能保存该过程中局部变量的值, 则应使用哪个关键字在该过程中定义局部变量? ()

- A) Dim B) Private C) Public D) Static

6. 在参数传递过程中, 使用关键字来修饰参数, 可以使之按值传递, 此关键字是()。

- A) ByVal B) ByRef C) Value D) Reference

7. 在语句 Public Sub Sort(i As Integer)中的 i 是一个按()传递的参数。

- A) 地址 B) 值 C) 变量 D) 常量

8. 在窗体上画一个名称为 Text1 的文本框如一个名称为 Command1 的命令按钮, 然后编写如下事件过程和通用过程:

```
Private Sub Command1_Click()  
    n = Val(Text1.Text)  
    If n\2 = n/2 Then  
        f = f1(n)  
    Else  
        f = f2(n)  
    End If  
    Print f; n  
End Sub  
Public Function f1(ByRef x)  
    x=x*x  
    f1=x+x  
End Function  
Public Function f2(ByVal x)  
    x=x*x  
    f2=x+x+x  
End Function
```

程序运行后, 在文本框中输入 6, 然后单击命令按钮, 窗体上显示的是()。

- A) 72 36 B) 108 36 C) 72 6 D) 108 6

二、填空题

1. 在窗体上画一个名称为 Command1 的命令按钮, 然后编写如下通用过程和命令按钮的事件过程:

```
Private Function f(m As Integer)  
    If m Mod 2 = 0 Then  
        f = m  
    Else  
        f = 1  
    End If  
End Function  
Private Sub Command1_Click()  
    Dim i As Integer  
    s = 0  
    For i = 1 To 5
```

```

        s = s + f(i)
    Next
    Print s
End Sub

```

程序运行后，单击命令按钮，在窗体上显示的是_____。

2. 在窗体上画一个名称为 Command1 的命令按钮和 3 个名称分别为 Label1、Label2、Label3 的标签，然后编写如下代码：

```

Private x As Integer
Private Sub Command1_Click()
    Static y As Integer
    Dim z As Integer
    n = 10
    z = n + z
    y = y + z
    x = x + z
    Label1.Caption = x
    Label2.Caption = y
    Label3.Caption = z
End Sub

```

运行程序，连续 3 次单击命令按钮后，则 3 个标签中显示的内容分别是、_____、_____。

3. 设有如下程序：

```

Private Sub Form_Click()
    Dim a As Integer, b As Integer
    a = 20: b = 50
    p1 a, b
    p2 a, b
    p3 a, b
    Print "a="; a, "b="; b
End Sub

Sub p1(x As Integer, ByVal y As Integer)
    x = x + 10
    y = y + 20
End Sub

Sub p2(ByVal x As Integer, y As Integer)
    x = x + 10
    y = y + 20
End Sub

Sub p3(ByVal x As Integer, ByVal y As Integer)
    x = x + 10
    y = y + 20
End Sub

```

该程序运行后，单击窗体，则在窗体上显示的内容是：a=_____和 b=_____。

4. 运行下面的程序，当单击窗体时，在窗体上显示的内容是_____。

```

Private Sub Test(x As Integer)

```

```
x=x*2+1
If x<6 Then
Call Test(x)
End If
x=2*x-1
Form1.Print x
End Sub
Private Sub Form_Click()
Test 2
End Sub
```

三、编程题

1. 给定两组已按升序排列的整型数据，使用过程编写程序把它们合并为一组仍能按升序排列的数据。

2. 编写一个函数 `reverseDigit(ByVal num as Integer)`。该函数将整数 `num` 的每个位上的数字逆序排列后返回。

3. 编写一个递归程序，求 `a` 和 `b` 的最大公约数

递归公式为：

当 `b=0` 时，`a` 与 `b` 的最大公约数为 `a`；

当 `b>0` 时，`a` 与 `b` 的最大公约数为 `b(a Mod b)` 的最大公约数。

第 7 章 鼠标和键盘事件过程

教学提示：鼠标和键盘是用户与计算机交流的重要工具。Visual Basic 应用程序可以响应多种鼠标事件和键盘事件。例如，单击 Click、双击 DblClick 事件是最基本的鼠标事件。许多控件，如窗体、图像控件等都可以检测鼠标指针的位置，并响应相应的单击、双击、左(右)键按下等事件。利用键盘事件可以响应各种键盘操作，还能解释和处理 ASCII 字符。

教学目标：了解常用的鼠标事件，能够设置鼠标光标，掌握 KeyPress 事件、KeyDown 事件和 KeyUp 事件的正确用法，理解拖放的概念。

7.1 鼠 标

7.1.1 鼠标事件

由用户操作鼠标所引起的、能够被 Visual Basic 各种对象识别的事件即为鼠标事件。当按下鼠标的按键时，发生 MouseDown 事件；松开鼠标按键时，发生 MouseUp 事件；移动鼠标时，发生 MouseMove 事件。工具箱中的大多数控件都能响应这 3 个事件，当鼠标位于某个控件或窗体上，则该控件或窗体将识别鼠标事件。

鼠标事件的语法结构基本相同，下面以窗体上发生的 MouseMove 为例进行说明。

```
Private Sub <对象名>_MouseMove([index As Integer, ] button As Integer, shift  
As Integer, x As Single, y As Single)
```

其中：

- 对象名是可以响应鼠标事件的对象名称。
- Index 参数值是一个整型数，用来唯一标识一个在控件数组中的控件。如果不是控件数组，此项省略。
- Button 参数值是一个整型数，参数的值反映事件发生时按下的是哪个鼠标键。1(常数 vbLeftButton)表示左键；2(常数 vbRightButton)表示右键；4(常数 vbMiddleButton)表示中键。
- 对于 MouseMove 事件，事件发生时，可能同时有两个或 3 个鼠标键被按下，这时 button 参数是相应的两个或 3 个值的和。例如，如果 MouseMove 事件发生时，左键和右键都被按下，则参数 button 的值是 3(=1+2)。因为移动鼠标时，可以不按下任何一个鼠标键，所以对于 MouseMove 事件，这个参数可以为 0。
- Shift 参数也是一个整数，它表明在这 3 个鼠标事件发生时，键盘上的哪一个控制键(Shift、Ctrl 和 Alt 键)被按下。1(常数 vbShiftMask)表示 Shift 键；2(常数 vbCtrlMask)

表示 Ctrl 键; 4(常数 vbAltMask)表示 Alt 键。如果同时有两个或 3 个控制键被按下, 则 shift 参数值是相应键的数值之和。例如, 当事件发生时, Ctrl 键被按下, 则 Shift 参数传递的值为 2; 如果 Shift 键和 Alt 键同时处于按下状态, 则 shift 参数值为 5。如果事件发生时, 没有键盘控制键被按下, 则这个参数的值为 0。

- 参数 x、y 用于指定鼠标指针当前的位置。采用 ScaleMode 属性指定的坐标系。

应该注意的是, 当移动鼠标时, 会不断地发生 MouseMove 事件。但是, 并不是每经过一点都会发生 MouseMove 事件, 而是在移动过程中每间隔很短的一段时间发生一个此事件, 所以, 在相同的距离上, 鼠标移动的速度越快, 产生的 MouseMove 事件就越少。由于应用程序能接二连三迅速识别大量 MouseMove 事件, 因此, 一个 MouseMove 事件过程不适合去做那些需要大量计算时间的工作。

另外, 在对象上操作一次鼠标, 会产生多个与鼠标有关的事件, 如 Click 事件、Dblick 事件、MouseDown 事件、MouseUp 事件或 MouseMove 事件。对于不同类型的对象, 这些事件的产生顺序可能不同, 还有些对象不支持其中的某个事件。所以, 在使用前一定要仔细测试。

比如, 在窗体上单击, 会依次引发 MouseDown、MouseUp、Click 这 3 个事件, 在窗体上双击, 会依次引发 MouseDown、MouseUp、Click、Dblick、MouseUp 等事件, 在命令按钮上单击, 会依次引发 MouseDown、Click、MouseUp 事件。

【例 7.1】 设计一个涂鸦程序。当在窗体上按下鼠标左键并移动时, 在窗体上画出线条, 释放鼠标左键时停止画线。按住 Shift 键, 画出的线条为红色; 按住 Ctrl 键, 画出的线条为绿色; 按住 Alt 键, 画出的线条为蓝色; 3 个键都不按时, 画出的线条为黑色。

程序代码如下:

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single,
Y As Single)
    If Button = 1 Then
        CurrentX = X      '设置画线的起点
        CurrentY = Y
    End If
End Sub
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single,
Y As Single)
    Dim DrawColor As Long
    If Button = 1 Then
        If Shift = 1 Then
            DrawColor = vbRed
        ElseIf Shift = 2 Then
            DrawColor = vbGreen
        ElseIf Shift = 4 Then
            DrawColor = vbBlue
        Else
            DrawColor = vbBlack
        End If
        Line -(X, Y), DrawColor
    End If
End Sub
```

程序运行情况如图 7.1 所示。

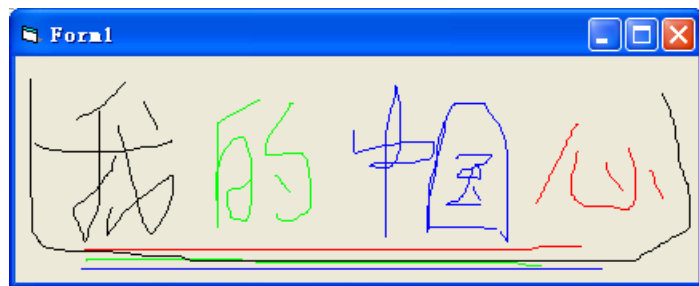


图 7.1 例 7.1 的运行情况

7.1.2 鼠标光标形状

在应用程序中，经常要根据鼠标光标所处位置或应用程序的状况来决定鼠标光标的形状，如鼠标光标形状为“I”字形表示插入文本，鼠标光标为“+”字形表示画线或画圆等。鼠标光标的形状由对象的 `MousePointer` 属性设置，属性值与形状如表 7-1 所示。

表 7-1 `MousePointer` 属性值

值	常 数	形 状
0	<code>vbDefault</code>	形状由操作系统决定(默认值)
1	<code>vbArrow</code>	箭头
2	<code>vbCrosshair</code>	+字型
3	<code>vbIbeam</code>	I 型
4	<code>vbIconPointer</code>	箭头图标
5	<code>vbSizePointer</code>	4 个方向的箭头
6	<code>vbSizeNESW</code>	指向右上和左下方向的双向箭头
7	<code>vbSizeNS</code>	指向上下的双向箭头
8	<code>vbSizeNWSE</code>	指向左上、右下方向的双向箭头
9	<code>vbSizeWE</code>	左右方向的双向箭头
10	<code>vbUpArrow</code>	向上的箭头
11	<code>vbHourglass</code>	沙漏(表示等待状态)
12	<code>vbNoDrop</code>	“不能停止”图标
13	<code>vbArrowHourglass</code>	箭头和沙漏
14	<code>vbArrowQuestion</code>	箭头和问号
15	<code>vbSizeAll</code>	四向箭头(表示改变大小)
99	<code>vbCustom</code>	通过 <code>MouseIcon</code> 属性指定的自定义图标

MouseIcon 属性使用一个图标文件来自定义鼠标指针形状。只有当 MousePointer 属性的值为 99 时,此属性才有效。在程序中应该使用 LoadPicture 函数装入磁盘文件(以 .ico 和 .cur 为扩展名的图标文件)来设置此属性。例如:

```
Form1.MousePointer = 99
Form1.MouseIcon = LoadPicture("abc.ico")
```

一般来说,改变鼠标指针的形状不是为了美观,而是向用户显示当前的工作状态。不同的状态下具有描述性的鼠标指针形状是良好用户界面的重要部分。

7.2 键 盘

在需要处理文本的地方,经常会用到键盘,因而有必要对键盘事件进行编程。Visual Basic 中,常用的键盘事件有 KeyUp、KeyDown、KeyPress 等事件。

7.2.1 KeyDown 和 KeyUp 事件

当一个对象拥有焦点时按下或松开一个键时发生 KeyDown 和 KeyUp 事件。其语法为:

```
Private Sub Form_KeyDown(keycode As Integer, shift As Integer)
Private Sub <控件名>_KeyDown([index As Integer,] keycode As Integer, shift
As Integer)
Private Sub Form_KeyUp(keycode As Integer, shift As Integer)
Private Sub <控件名>_KeyUp([index As Integer,] keycode As Integer, shift As
Integer)
```

其中:

- **keycode** 是一个整型参数,表示按键的代码。每一个按键都有相应的代码。Visual Basic 还为每个按键的代码声明了一个内部常量。例如, F1 键的代码为 112, 相应的内部常量为 vbKeyF1; Home 键的代码为 36, 内部常量为 vbKeyHome。键盘上字母与数字键的代码与其 ASCII 码是相同的,其他键的代码与相应的常量可参考 Visual Basic 的帮助文件。
- **shift** 也是一个整型参数,它指示在按下一个键时,是否同时按下了 Shift、Ctrl 和 Alt 键。此参数为 1 时,表示按下了 Shift 键;为 2 时,表示按下了 Ctrl 键;为 4 时,表示按下了 Alt 键。当这 3 个键中有不只一个键被按下,则 shift 参数是被按下键相应数值之和。例如,如果 shift 参数值为 5,表明按下了 Shift 和 Alt 两个键。如果 3 个键均未被按下,该参数值为 0。

对于 KeyDown 和 KeyUp 事件来说,带焦点的对象都可以接收所有用户击键。一个窗体只有当它的所有控件都不可见或都无效时才可以获得焦点,得到键盘事件。虽然 KeyDown 和 KeyUp 事件可应用于大多数键,但它们最常用于扩展的字符键如功能键、定位键、键盘修饰键和按键的组合,区别数字小键盘和常规数字键。

下列情况不能引用 KeyDown 和 KeyUp 事件。

- (1) 窗体上有一个命令按钮控件,并且其 Default 属性设置为 True 时按下 Enter 键。

因为此时 Enter 键将触发该命令按钮的 Click 事件而不是 KeyDown 和 KeyUp 事件。

(2) 窗体上有一个命令按钮控件，并且其 Cancel 属性设置为 True 时按下 Esc 键。因为此时 Esc 键将触发该命令按钮的 Click 事件而不是 KeyDown 和 KeyUp 事件。

(3) 窗体上有多个可拥有焦点的控件时，按下 Tab 键。

对于字母键，keycode 参数返回的总是大写形式，要知道当前的大小写形式，应该检测 shift 参数的值。

7.2.2 KeyPress 事件

当用户按下和松开一个 ANSI 键时发生 KeyPress 事件。ANSI 键包括数字、大小写字母、Enter、BackSpace、Esc、Tab 等键。方向键不会产生 KeyPress 事件。

KeyPress 事件过程的语法为：

```
Private Sub Form_KeyPress(keyascii As Integer)
Private Sub <对象名>_KeyPress([index As Integer,]keyascii As Integer)
```

其中：

- index 为整型参数，它用来唯一标识一个在控件数组中的控件。
- keyascii 参数返回一个标准数字 ANSI 键代码的 ASCII 值。keyascii 通过引用传递，对它进行改变可给对象发送一个不同的字符。将 keyascii 改变为 0 时可取消击键，此时对象接收不到字符。

需要特别注意的是，只有具有焦点的对象才能接收 KeyPress 事件。一个窗体仅在它没有可视和有效的控件或 KeyPreview 属性被设置为 True 时才能接收该事件。KeyPress 事件过程在截取 TextBox 或 ComboBox 控件所输入的击键时是非常有用的，它可立即测试击键的有效性或在字符输入时对其进行格式处理。改变 keyascii 参数的值会改变所显示的字符。将 keyascii 改变为 0 时可取消击键，这样对象便接收不到字符了。

KeyPreview 是窗体的一个属性，当此属性被设置为 True 时，窗体先于该窗体上的控件接收到键盘事件。可以利用该属性，编制窗体的键盘处理程序。例如，应用程序使用功能键时，只需要在窗体级处理击键，而不应为每个可以接收击键事件的控件编写程序。

当窗体处理完键盘事件之后，控件还会接收到键盘事件。如果只在窗体级处理键盘事件，而不允许控件接收键盘事件，只需在窗体的 KeyDown 事件过程中设置 KeyCode 为 0 即可。

一些控件能够拦截键盘事件，以致于窗体不能接收它们。例如按钮控件有焦点时按 Enter 键以及焦点在列表框控件上时按方向键。

【例 7.2】 将输入到 TextBox 控件的文本转换为大写形式。

```
Private Sub Text1_KeyPress (KeyAscii As Integer)
    Char = Chr(KeyAscii)
    KeyAscii = Asc(UCase(Char))
End Sub
```

执行本代码时，在 Text1 控件中输入字母，则可将其转化为大写形式。

【例 7.3】 编写一个简易的指法练习程序。要求最大时限为 1 分钟。

在窗体上添加 3 个标签框 Label1~label3，用来动态显示英文字母，一个命令按钮

Command1 用来开始程序, 计时器 Timer1 用来控制标签框的移动, 另一个计时器 Timer2 用来控制总时间。设计界面如图 7.2 所示。界面中控件的属性如表 7-2 所示。

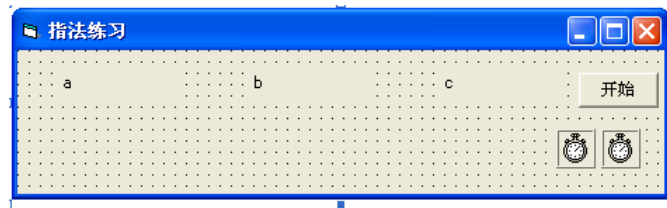


图 7.2 指法测试程序设计界面

表 7-2 例 7.3 的窗体中控件属性表

对 象	对 象 名	属 性 名	属 性 值
Form	Form1	Caption	指法练习
		KeyPreview	True
Timer	Timer1	Interval	15
	Timer2	Interval	60000
CommandButton	Command1	Caption	开始
Label	Label1	Caption	a
	Label2	Caption	b
	Label3	Caption	c

程序代码如下:

```

'n 存放正确的击键数 ,m 存放总击键数
Dim n As Integer, m As Integer
Private Sub Command1_Click()
    Form1.Cls
    Timer1.Enabled = True
    Timer2.Enabled = True
    Command1.Enabled = False
End Sub
Private Sub Form_KeyPress(KeyAscii As Integer)
    m = m + 1      '总击键数加 1
    '击键正确, 正确击键数加 1, 并清空标签
    If Chr(KeyAscii) = Label1.Caption Then Label1.Caption = "": n = n + 1
    If Chr(KeyAscii) = Label2.Caption Then Label2.Caption = "": n = n + 1
    If Chr(KeyAscii) = Label3.Caption Then Label3.Caption = "": n = n + 1
End Sub
Private Sub Timer1_Timer()
    Randomize
    '击键正确后产生新字符并将标签移到窗体底部, 随机产生小写的英文字母
    If Label1.Caption = "" Then
        Label1.Top = Form1.Height - Label1.Height
        Label1.Caption = Chr(CInt(Rnd * 26 + 97))
    
```

```

Else
    Label1.Top = Label1.Top - 10
End If
If Label2.Caption = "" Then
    Label2.Top = Form1.Height - Label2.Height
    Label2.Caption = Chr(CInt(Rnd * 26 + 97))
Else
    Label2.Top = Label2.Top - 10
End If
If Label3.Caption = "" Then
    Label3.Top = Form1.Height - Label3.Height
    Label3.Caption = Chr(CInt(Rnd * 26 + 97))
Else
    Label3.Top = Label3.Top - 10
End If
' 标签移动到窗体顶部后,重新回到底部
If Label1.Top <= 0 Then Label1.Top = Form1.Height - Label1.Height
If Label2.Top <= 0 Then Label2.Top = Form1.Height - Label2.Height
If Label3.Top <= 0 Then Label3.Top = Form1.Height - Label3.Height
End Sub
Private Sub Timer2_Timer()
    Timer1.Enabled = False
    Timer2.Enabled = False
    Print "击键次数: " & m & " 次"
    Print "正确次数: " & n & " 次"
    If m > 0 Then Print "正确率为: " & n / m * 100 & "%"
    n = 0
    m = 0
    Command1.Enabled = True
End Sub

```

7.3 拖 放

拖动(Drag and Drop)是一种将一个对象(称为源对象)拖到另一个对象(称为目标对象)上执行某种操作的功能。要拖动一个对象,可单击该对象,按住鼠标键,然后移到指定的新位置释放鼠标,放下对象。

拖动是 Windows 操作系统用户界面的一个独到之处。在 Windows 桌面、资源管理器、控制面板以及各应用程序中,随处都可以使用拖动操作。

Visual Basic 支持控件拖动和 OLE 拖动两种拖动。

7.3.1 与拖动有关的属性、事件和方法

1. DragMode 属性

拖动的形式有两种, DragMode 属性为 1 时为自动拖动, DragMode 属性为 0 时为手工拖动。启动自动拖动时,用户在源对象上按下并拖动鼠标,则源对象随着鼠标一起移动,

移动到某个目标对象上将产生 DragOver 事件, 释放鼠标时将产生 DragDrop 事件。拖放过程本身并不会将源对象放到目标对象上。要将源对象拖放到某处, 必须编写目标对象的 DragOver 事件过程或 DragDrop 事件过程。

采用手动拖放方式时, 必须在 MouseDown 事件过程中使用 Drag 方法启动拖动操作, 之后的操作同自动拖放方式。

当源对象的 DragMode 属性为 1 时, 源对象不会接受 Click 事件和 MouseMove 事件。而源对象 DragMode 属性为 0 时, 源对象可以接受 Click 事件和 MouseDown 事件。

2. DragIcon 属性

DragIcon 属性用于改变拖动图标。拖动控件时, Visual Basic 将控件的灰色轮廓作为默认的拖动图标。对 DragIcon 属性进行设置, 就可用其他图像代替该轮廓。设置 DragIcon 属性的最简单方法就是使用属性窗口。选定 DragIcon 属性后单击【属性】按钮, 在打开的【加载图标】对话框中选择图标文件(.ico, .cur)。

可将 Visual Basic 图标库或操作系统图标库中的图标分配给 DragIcon 属性。也可用图像程序创建用户自己的拖动图标。

另外, 在运行时可以采用如下方法获得一个控件的 DragIcon 属性。

```
' 将 Image2 控件的 DragIcon 属性赋给 Image1 控件的同一属性
Set Image1.DragIcon = Image2.DragIcon
' 运行时将 Image2 控件的 Picture 属性赋给 Image1 控件的同一属性
Set Image1.DragIcon = Image2.Picture
' 装入图标文件
Set Image1.DragIcon = LoadPicture("c:\WINNT\Cursors\3dsmove.ico")
```

3. Drag 方法

采用手动拖放方式时, 必须使用 Drag 方法来启动拖动操作。不过, 在自动拖放方式下, 也可以使用 Drag 方法。

Drag 方法的语法为:

```
[对象名.]Drag [action]
```

其中: 参数 action 为可选项, 是一个指定要执行的动作的 Visual Basic 常数或数值。

- action=1(vbBeginDrag): 启动控件的拖动。
- action=2(vbEndDrag): 结束拖放并引发 DragDrop 事件。
- action=0(vbCancel): 取消拖放操作, 不引发 DragDrop 事件。

【例 7.4】 将图片框控件拖放到窗体中准确的位置。

在窗体中放置一个图片框控件 Picture1, 任意设置一个 Picture 属性值。将 Picture1 的 DragMode 属性设置为 1。因为在自动拖放方式下, MouseDown 事件无效, 也就无法记住(X0, Y0)。

采用 Move 方法对控件进行移动时, 总是将控件的左上角移动到 Move 方法的参数所指的位置。为了将控件准确地拖到鼠标所指位置, 应记住鼠标在控件上按下时在控件上所处的位置(X0, Y0)。如果不记住(X0, Y0), 只采用“Move X, Y”, 则必须拖动控件的左上角, 才能在释放鼠标后使控件不出现移动。注意(X0, Y0)是相对于控件左上角的位置。

程序代码如下：

```
Dim X0 As Single, Y0 As Single
Private Sub Form_DragDrop(Source As Control, X As Single, Y As Single)
Source.Move (X - X0), (Y - Y0)
End Sub
Private Sub Picture1_MouseDown(Button As Integer, Shift As Integer, X As
Single, Y As Single)
Picture1.Drag=1      '启动控件拖放
X0 = X: Y0 = Y
End Sub
```

4. DragDrop 事件和 DragOver 事件

在拖动对象经过某个对象时，该对象发生 DragOver 事件，拖动对象并在某个对象上释放鼠标按钮时，该对象发生 DragDrop 事件。可用多种方法响应 DragDrop 事件和 DragOver 事件。DragDrop 事件和 DragOver 事件的语法为：

```
Private Sub <对象名>_DragDrop(Source As Control, X As Single, Y As Single)
Private Sub <对象名>_DragOver(Source As Control, X As Single, Y As Single,
State As Integer)
```

其中：

- Source 的类型为控件，对应于正在被拖动的控件，因而可以像使用控件一样使用它，如可引用其属性或调用其方法。x 和 y 为当前鼠标指针在目标窗体或控件中水平(x) 和垂直(y)位置的数字。这些坐标值通常用目标坐标系统来表示。
- State 是一个整数，它相应于一个控件的转变状态，该控件在相关目标窗体或控件中正在被拖动，State 为 0 时表示进入(源控件正被向一个目标范围内拖动)，为 1 时表示离去(源控件正被向一个目标范围外拖动)，为 2 时表示跨越(源控件在目标范围内从一个位置移到另一位置)。

下面的过程在拖动 Picture1 控件经过 Picture2 控件时，将 Picture1 控件中的图像“移交” Picture2 控件。注意在设计时将 Picture1.DragMode 设置为 1。

```
Private Sub Picture2_DragDrop(Source As Control, X As Single, Y As Single)
Picture2.Picture = Picture1.Picture : Picture1.Picture = LoadPicture()
End Sub
```

对源参数的使用应特别小心。因为源参数引用的是控件，但不一定清楚引用哪种控件。例如，如果控件是文本框，并试图引用 Source.Value，则由于文本框没有 Value 属性而导致运行时错误。可以使用 TypeOf 函数判断源对象的控件类型，根据源对象的控件类型决定采取的操作方式，其形式为：

```
If TypeOf <对象变量> Is <控件类型> Then
```

也可使用 TypeName 函数，该函数返回一个类型字符串。

【例 7.5】 将控件拖动到 Picture2 图片框控件上时，判断拖动的控件类型，如果是图片框，则将其图片在 Picture2 上显示，并在窗体的标题内显示拖动控件的类型。当拖离 Picture2 时，清除 Picture2 中的图片，恢复窗体标题栏的名称(设计时窗体标题为“拖放”)。

```
Private Sub Picture2_DragOver(Source As Control, X As Single, Y As Single,
State As Integer)
If TypeOf Source Is Picture Then Picture2.Picture = Source.Picture
If State = 1 Then
Form1.Caption = "拖放" : Picture2.Picture = LoadPicture()
Else
Form1.Caption = "这是" & TypeName(Source) & "控件"
End If
End Sub
```

5. Parent 属性

控件作为参数 Source 传递给过程后, Source 就继承了控件的属性和方法,包括控件的 Parent 属性,即控件所在窗体,因而可以对控件所在窗体进行各种操作。如上面的程序中的语句:

```
Form1.Caption = "拖放"
Form1.Caption = "这是" & TypeName(Source) & "控件"
```

可以改为:

```
Source.Parent.Caption = "拖放"
Source.Parent.Caption = "这是" & TypeName(Source) & "控件"
```

7.3.2 OLE 拖放

使用控件拖放可以将控件移动到其他地方。而采用 OLE 拖放,则可以将数据从一个控件或应用程序移动到另外的控件或应用程序中,如将写字板中的文本拖动到文本框中。

“拖”数据由 OLEDragMode 属性控制,“放”数据由 OLEDropMode 属性控制。

- OLEDragMode=0(Mannual), 为默认的 OLE 拖动方式,即手工拖动。
- OLEDragMode=1(Automatic), 自动实现“拖”操作。
- OLEDropMode=0(None), 默认方式,不接受 OLE “放”操作。
- OLEDropMode=1(Mannual), 手工实现“放”操作。
- OLEDropMode=2(Automatic), 自动实现“放”操作。

说明:

(1) 在 Visual Basic 中,文本框、图片框等控件完全支持自动 OLE 拖放,即不需要额外的编程。组合框、列表框等控件支持自动“拖”操作,而不支持自动“放”操作,它们的 OLEDrop Mode 属性不能设置为 2(Automatic),需要额外的编程才能实现“放”操作。

(2) 除非在拖动时同时按下 Ctrl 键或通过代码修改控件的默认性能,将数据拖动到文本框控件中时,进行的是移动而不是复制操作。

(3) 自动支持拖放操作有其局限性,例如,如果将 Word 文档中的文本拖动到文本框控件中,则 Word 文档中的所有丰富文本格式将消失,因为文本框控件不支持这种格式。如果想要确定对象所支持的是哪种数据格式或拖放效果(复制、移动或不放),或者,如果想使从中拖出数据来的控件不支持自动拖动,可使用手工的 OLE 拖动操作。

(4) 为了将 Data Object 对象中的数据放到目标部件中,应在目标部件的 OLEDrag Drop 事件过程中使用 Get Data 方法,其语法格式为:

<Data Object 对象名>.Get Data (Format)

其中 Format 用于确定 Data Object 对象数据格式的 Visual Basic 常数或值。如 vbCFText 表示文本文件，vbCFBitmap 表示 .bmp 文件。

【例 7.6】 在 Form1 窗体中放置两个列表框控件、一个文本框控件和一个标签控件。将两个列表框控件的 OLEDragMode 属性设置为“1-Automatic”，OLEDropMode 属性设置为“1-Manual”，MultiSelect 属性设置为“2-Extended”以便一次选择多行。将文本框的 MultiLine 属性设置为“True”，将 OLEDragMode 属性和 OLEDropMode 属性都设置为“Automatic”。将标签控件的 OLEDropMode 属性设置为“1-Manual”。

编写程序实现如下功能。

- (1) 将 Word 中的文本采用拖动的方法复制到中间的文本框中；
- (2) 将文本框的内容逐行拖放到左边的 List1 列表框中；
- (3) 将文本框的内容一次拖放到下边的 List2 列表框中；
- (4) 将下边 List2 列表框的内容拖放到右边的 Label1 标签上。

程序代码如下：

```
Private Sub List1_OLEDragDrop(Data As DataObject, Effect As Long, Button As Integer, Shift As Integer, X As Single, Y As Single)
    ' 系统将拖出的数据放在 Data 对象中，利用 Data 对象的 GetData 方法获取其中的文本数据，
    ' 利用 List1 控件的 AddItem 方法将数据添加到其 List 属性中
    List1.AddItem Data.GetData(vbCFText)
End Sub

Private Sub List2_OLEDragDrop(Data As DataObject, Effect As Long, Button As Integer, Shift As Integer, X As Single, Y As Single)
    List2.AddItem Data.GetData(vbCFText)
End Sub

Private Sub Label1_OLEDragDrop(Data As DataObject, Effect As Long, Button As Integer, Shift As Integer, X As Single, Y As Single)
    Label1.Caption = Data.GetData(vbCFText)
End Sub
```

程序运行时，打开一个 Word 文档，选定全部文本，按住 Ctrl 键，拖动到 Text1 文本框中，然后将 Text1 文本框中的内容逐行拖放到左边的 List1 列表框中。再将 Text1 文本框中的文本一次性拖动到 List2 列表框中，此时 List2 列表框中整个文本将在一行显示；将此行文本拖动到标签控件时，会在 Label1 中多行显示。运行结果如图 7.3 所示。

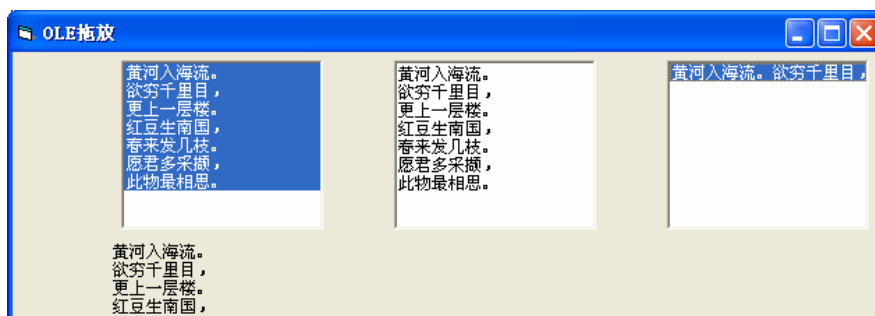


图 7.3 OLE 拖放实例

7.4 小 结

由用户操作鼠标所引起的、能够被 Visual Basic 各种对象识别的事件即为鼠标事件。当按下鼠标按键时, 发生 `MouseDown` 事件; 松开鼠标按键时, 发生 `MouseUp` 事件; 移动鼠标时, 发生 `MouseMove` 事件。用鼠标拖放对象时, 发生 `DragDown` 事件和 `DragOver` 事件。

`MouseDown`、`MouseUp`、`MouseMove` 这 3 个鼠标事件过程的格式如下:

```
Private Sub <对象名>_MouseDown ([index As Integer,] button As Integer, shift  
As Integer, x As Single, y As Single)  
Private Sub <对象名>_MouseUp ([index As Integer,] button As Integer, shift  
As Integer, x As Single, y As Single)  
Private Sub <对象名>_MouseMove ([index As Integer,] button As Integer, shift  
As Integer, x As Single, y As Single)
```

Visual Basic 中定义了 `KeyPress`、`KeyUp` 和 `KeyDown` 这 3 个键盘事件过程。当用户在键盘上按下一个 ANSI 键时, 就发生 `KeyPress` 事件; 当控制焦点位于某对象上时, 在键盘上按下任一键就触发该对象的 `KeyDown` 事件, 放开该键时, 发生 `KeyUp` 事件。

`KeyPress`、`KeyUp`、`KeyDown` 这 3 个键盘事件过程的格式如下:

```
Private Sub <对象名>_KeyPress ([index As Integer,] keyascii As Integer)  
Private Sub <对象名>_KeyDown ([index As Integer,] keycode As Integer, shift  
As Integer)  
Private Sub <对象名>_KeyUp ([index As Integer,] keycode As Integer, shift As  
Integer)
```

拖放是一种将一个对象(称为源对象)拖到另一个对象(称为目标对象)上执行某种操作的功能。要拖动一个对象, 可单击该对象, 按住鼠标键, 然后移到指定的新位置并释放鼠标可释放对象。

Visual Basic 支持控件拖放和 OLE 拖放两种拖放。

7.5 习 题

一、选择题

1. 窗体的 `MouseDown` 事件过程

```
Form_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As  
Single)
```

有 4 个参数, 关于这些参数, 正确的描述是()。

- A) 通过 `Button` 参数可判定当前按下的是哪一个鼠标键
- B) `Shift` 参数只能用来确定是否按下 `Shift` 键

C) Shift 参数只能用来确定是否按下 Alt 和 Ctrl 键

D) 参数 X、Y 用来设置鼠标当前位置的坐标

2. 程序运行后, 在窗体上单击鼠标, 此时窗体不会接收到的事件是()。

A)MouseDown

B) MouseUp

C) Load

D) Click

3. 下面的程序运行时, 为了在窗体上输出“Visual Basic 6.0”, 应在窗体上执行的操作是()。

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Button And 3 = 3 Then
        Print "Visual Basic 6.0"
    End If
End Sub
```

A) 只能按下鼠标右键

B) 只能按下鼠标左键

C) 应同时按下鼠标左、右键

D) 按下鼠标左、右键之一

4. 如果将上述事件改为 Form_Move 事件, 为实现同样的输出, 则执行的操作为()。

A) 只能按下鼠标右键并拖动

B) 只能按下鼠标左键并拖动

C) 应同时按下鼠标左、右键并拖动

D) 按下鼠标左、右键之一并拖动

5. 在窗体上画一个名称为 TxtA 的文本框, 然后编写如下事件过程:

```
Private Sub TxtA_KeyPress(keyascii as integer)
...
End Sub
```

若焦点位于文本框中, 则能够触发 KeyPress 事件的操作是()。

A) 单击鼠标

B) 双击文本框

C) 让鼠标滑过文本框

D) 按下键盘上的某个键

6. 在窗体上画一个名称为 Text1 的文本框, 要求文本框只能接收大写字母的输入。以下能够实现该操作的事件过程是()。

```
A) Private Sub Text1_KeyPress(KeyAscii As Integer)
    If KeyAscii < 65 Or KeyAscii > 90 Then
        MsgBox "请输入大写字母"
        KeyAscii = 0
    End If
End Sub

B) Private Sub Text1_KeyDown(KeyCode As Integer, Shift As Integer)
    If KeyCode < 65 Or KeyCode > 90 Then
        MsgBox "请输入大写字母"
        KeyCode = 0
    End If
End Sub

C) Private Sub Text1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Asc(Text1.Text) < 65 Or Asc(Text1.Text) > 90 Then
        MsgBox "请输入大写字母"
```

```
End If
End Sub
D) Private Sub Text1_Change()
    If Asc(Text1.Text) > 64 And Asc(Text1.Text) < 91 Then
        MsgBox "请输入大写字母"
    End If
End Sub
```

7. 在窗体上画一个名称为 Text1 的文本框，并编写如下程序：

```
Private Sub Form_Load()
    Show
    Text1.Text = ""
    Text1.SetFocus
End Sub
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single,
Y As Single)
    Print "程序设计"
End Sub
Private Sub Text1_KeyDown(KeyCode As Integer, Shift As Integer)
    Print "Visual Basic";
End Sub
```

程序运行后，如果按 A 键，然后单击窗体，则在窗体上显示的内容是()。

- A) Visual Basic
- B) 程序设计
- C) A 程序设计
- D) Visual Basic 程序设计

8. 以下叙述中错误的是()。

- A) 在 KeyPress 事件过程中不能识别键盘的按下与释放
- B) 在 KeyPress 事件过程中不能识别回车键
- C) 在 KeyDown 和 KeyUp 事件过程中，将从键盘输入的“A”和“a”视作相同的字母
- D) 在 KeyDown 和 KeyUp 事件过程中，从大键盘上输入的“1”和从右侧小键盘上输入的“1”被视作不同的字符

9. 在文本框中，当用户输入一个字符时，能同时引发的事件是()。

- A) KeyPress 和 Click
- B) KeyPress 和 LostFocus
- C) KeyPress 和 Change
- D) Change 和 LostFocus

10. 新建工程的窗体上包括两个文本框 Text1、Text2 和一个命令按钮 Command1，编写如下事件过程：

```
Dim str1 As String, str2 As String
Private Sub Command1_Click()
    Text1.Text = str1
    Text2.Text = str2
    str1 = ""
    str2 = ""
End Sub
Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
```

```

str2 = str2 & Chr(KeyCode)
End Sub
Private Sub Form_KeyPress(KeyAscii As Integer)
str1 = str1 & Chr(KeyAscii)
End Sub
Private Sub Form_Load()
Text1.Text = ""
Text2.Text = ""
Text1.Enabled = False
Text2.Enabled = False
End Sub

```

(1) 在设计时设置 KeyPreview 属性值为 False，程序运行时，在键盘上输入 abc，则 Text1 和 Text2 中显示的内容分别是()。

- A) abc 和 ABC B) abc 和 656667
C) 不显示任何信息 D) 出错

(2) 在设计时设置 KeyPreview 属性值为 True，程序运行时，在键盘上输入 abc，则 Text1 和 Text2 中显示的内容分别是()。

- A) abc 和 ABC B) abc 和 656667
C) 不显示任何信息 D) 出错

二、填空题

1. 在执行 KeyPress 事件过程时，KeyAscii 表示所按键的_____值，而 KeyCode 表示_____。

2. 编写如下代码，在程序运行时，如果想在窗体上输出 ABCD，则应执行的操作是_____。

```

Dim flag As Boolean
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single,
Y As Single)
flag = True
End Sub
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single,
Y As Single)
flag = False
End Sub
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single,
Y As Single)
If flag = False Then
Print "ABCD"
End If
End Sub

```

3. 为自定义鼠标光标，需要将_____属性设置为_____，然后将_____属性设置为图标文件。

4. 为了执行自动拖动，必须将_____属性设置为_____；为了执行手动拖动，

应将该属性设置为_____。

5. 在对象的 MouseDown 和 MouseUp 事件过程中, 当参数 Button 的值为 1、2、4 时, 分别代表按下鼠标的_____、_____和_____。

6. 在窗体上添加一个文本框 Text1, 编写如下代码:

```
Private Sub Text1_KeyDown(KeyCode As Integer, Shift As Integer)
Print Chr(KeyCode + 5) & KeyCode
End Sub
```

程序运行时, 在文本框中输入 abc, 则在窗体上输出为_____。

7. 当源对象被拖动到目标对象上方时, 在目标对象上将引发_____事件, 释放时又会引发_____事件。

8. 控件被拖动时显示的图标是由控件的_____属性决定的。

9. 把窗体的 KeyPreview 属性设置为 True, 然后编写如下两个事件过程:

```
Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
Print Chr(KeyCode)
End Sub
Private Sub Form_KeyPress(KeyAscii As Integer)
Print Chr(KeyAscii)
End Sub
```

程序运行后, 如果直接按键盘上的“A”键(即不按住 Shift 键), 则在窗体上输出的字符分别是_____和_____。

三、编程题

1. 编写一个程序, 只能向窗体中的文本框输入字母和数字字符, 并且从键盘上键入的字母如果是小写的, 则需要转换为大写字母在文本框中显示。

2. 编写一个程序, 当鼠标在窗体上移动时, 能够绘出与光标移动轨迹相接近的图形。

3. 编写类似“回收站”的程序, 当将窗体上的控件对象拖到“回收站”上并释放鼠标时, 在打开的消息框中提示是否删除该控件, 单击消息框中的【是】按钮, 删除该控件, 否则控件复原。拖动控件时, 鼠标的图标为该控件中的图标。

第 8 章 界 面 设 计

教学提示：用户界面是应用程序的一个重要组成部分，对用户而言，界面就是应用程序，它使得用户感觉不到在后台运行的代码。不论花费多少时间来优化代码，应用程序的可用性主要取决于界面的设计。本章主要介绍如何使用 Visual Basic 提供的菜单技术、多重窗体和多重文档技术设计应用程序界面。

教学要求：掌握 Visual Basic 中菜单、对话框、工具栏、多重窗体和多重文档的设计方法，能够熟练地使用它们设计应用程序界面。

8.1 菜单的设计

要执行新建文件、打开文件等操作都离不开菜单，菜单已经成为窗口界面不可或缺的一部分。菜单代表的是程序的各项命令，在进行界面设计时，一般要将功能类型一致的命令放在同一个子菜单中，功能类型不同的命令放在不同的子菜单中。设计菜单时，开发者基本都遵循同样的标准，第一个通常是“文件”，最后一个总是“帮助”，这是与 Windows 环境相一致的。考虑到用户的使用习惯，程序菜单应尽量与别的 Windows 程序菜单相一致。

8.1.1 下拉式菜单

Visual Basic 6.0 通过菜单编辑器进行菜单的设计，使用菜单编辑器可以创建新的菜单和菜单栏，还可以在已有的菜单上增加新命令，用自己的命令替换已有的菜单命令，以及修改、添加和删除已有的菜单。

右键单击当前窗体，在弹出菜单中选择【菜单编辑器】选项，弹出的菜单编辑器窗口如图 8.1 所示。

用户可以在菜单编辑器中设计自己的菜单。菜单相当于简单的按钮，它的大部分属性我们都曾经见过，包括 Name、Caption、Checked、Enabled、Index、Visible、快捷键和热键等。其中 Name 是必需的。这些属性的功能如表 8-1 所示。

表 8-1 菜单的一些主要属性

属性名称	设置值的类型	说 明
Name	字符型	编辑代码时用来引用控件的唯一标识，菜单名字的标准前缀是 mnu
Caption	字符型	出现在控件上的文本。若想定义热键可在标题中使用“&”字符，运行时可用 Alt 键加上带下划线的字符来选中该菜单项

续表

属性名称	设置值的类型	说 明
Checked	布尔型	是否在菜单上放置一个复选标志
Enabled	布尔型	菜单项是否可用，不可用时为灰色
Index	整型	用于创建菜单控件数组，即所有的菜单名称都相同，用不同的 Index 属性来区分
Visible	布尔型	设置菜单项是否可见



图 8.1 菜单编辑器

下面简要介绍菜单编辑器窗口的其他部分的功能。

- 快捷键：用户可从该下拉列表中选择快捷键的组合。
- 帮组上下文 ID：可选项，取值为数值，用来调用为菜单项准备的帮助文件的标题页。若用户按下 F1 键，则该数值用来定位和显示帮助文件。
- 协调位置：决定如何在容器窗体中显示菜单。
- 显示窗口列表：设定 MDI 程序中，菜单控件是否包含一个打开的 MDI 子窗口。
- 插入：在选中位置插入一个新的菜单项。
- 删除：删除一个菜单项。

下面开始设计一个简单菜单。输入表 8-2 列出的属性值。

表 8-2 菜单的属性值设置

标题 (Caption)	名称 (Name)
文件(&F)	mnuFile
打开(&O)	mnuOpen
退出(&X)	mnuExit
编辑	mnuEdit
帮助	mnuHelp

设计步骤如下。

(1) 打开【菜单编辑器】对话框

(2) 首先在【标题】文本框中添加【文件】，在【名称】文本框中添加 mnuFile，单击【下一个】按钮，这样就建立了第一级菜单，注意菜单设计窗口中有 4 个箭头，单击向右的箭头，出现了一排小点，这样就可以开始建立第二级菜单。

(3) 接着填写下一菜单项，【标题】设置为打开(&O)，【名称】设置为 MnuOpen，单击【下一个】按钮。重复上述操作，直至所有【文件】菜单下的二级子菜单添加完毕。

(4) 单击【下一个】按钮，然后单击左向箭头“←”，这样就可继续编辑与【文件】菜单项同级的菜单，重复步骤(2)~(4)，直至编辑完所有菜单项。

设计好的菜单如图 8.2 所示。



图 8.2 菜单示例

在菜单里经常会看到把不同项分开的分隔条，只要在需要添加分隔条的地方添加一个菜单项，把“标题”属性设置为“-”(减号)即可。

注意：“&”符号在设计时有特殊含义，如果要在菜单标题中显示“&”符号，则要在其前面加一个“&”。比如：Stop&&Exit 显示为“Stop&Exit”，而 Stop&Exit 会显示为 StopExit，它的热键是 E。

8.1.2 弹出式菜单

当用户在窗体的某个对象上单击鼠标右键，就会弹出和该对象相关的菜单，它是独立于菜单栏的浮动菜单，也成为上下文菜单。任何一个至少有一个菜单项的菜单，都可在运行时显示为弹出式菜单。

生成弹出菜单主要分为两步：

(1) 生成一个标准菜单

(2) 在某个通用事件中使用 PopupMenu 方法。

PopupMenu 方法的语法格式如下：

```
object.PopupMenu mnuname, flags, x, y, boldcommand
```

其中，各参数的说明如表 8-3 所示。

表 8-3 PopupMenu 方法参数表

参 数	说 明
object	使用弹出菜单的对象名，指定的菜单必须含有至少一个子菜单
mnuName	必选项。要显示的弹出菜单名
flags	可选项。取值为数值或常数，用来指定弹出菜单的行为和位置
x、y	可选项。指定弹出菜单的 x、y 坐标，若省略，则使用鼠标的当前坐标
boldCommand	可选项。指定弹出菜单中菜单控件的名字

flags 参数可设置的值及其含义如表 8-4 所示。

表 8-4 flags 参数的设计值

	常 数	值	说 明
位置	vbPopupMenuLeftAlign	0	默认值。弹出菜单的左边定位于 x
	vbPopupMenuCenterAlign	4	弹出菜单的中间定位于 x
	vbPopupMenuRightAlign	8	弹出菜单的右边定位于 x
行为	vbPopupMenuLeftButton	0	默认值。仅当使用鼠标左键时，弹出菜单中的项目才响应鼠标单击
	vbPopupMenuRightButton	2	使用鼠标左键或右键，弹出菜单中的项目都响应鼠标单击

要指定两种 flags，可用 or 操作符组合上述两种常数。

这里可以为上一小节创建的菜单添加弹出式菜单，具体如下：

```
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single,
Y As Single)
If Button = 2 Or vbRightButton Then
PopupMenu mnuFile
End If
End Sub
```

若想弹出一个自定义菜单，则可在利用菜单编辑器设计菜单时，将其 Visible 属性设置为 False。

8.1.3 菜单事件与菜单命令

若要上述创建的菜单可用，就要为它们的事件编写程序。菜单最常用的事件就是 Click 事件。下面举例说明。

【例 8.1】 利用菜单控制画线。其中单击【开始画线】菜单开始画线，单击【停止画线】菜单停止画线，单击【变换颜色】菜单变换画线颜色，程序运行界面如图 8.3 所示。

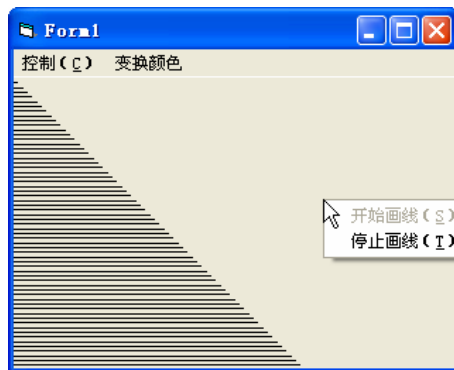


图 8.3 例 8.1 的运行界面

程序设计步骤如下。

(1) 新建“标准 EXE”工程。

(2) 建立程序用户界面。在窗体上添加 1 个定时器控件和若干个菜单，各菜单的名称、标题设置如表 8-5 所示。

表 8-5 例 8.1 的菜单属性设置

属 性	标 题
MnuControl	控制(&C)
mnuStart(mnuControl 的下一级菜单)	开始画线(&S)
mnuStop(mnuControl 的下一级菜单)	停止画线(&T)
MnuColor	变换颜色

(3) 进入代码窗口中，编写如下事件过程。

菜单 mnuStart 的响应代码如下：

```
Private Sub mnuStart_Click()  
    Timer1.Interval=10  
    mnuStart.Enabled=False  
    mnuStop.Enabled=True  
End Sub
```

菜单 mnuStop 的响应代码如下：

```
Private Sub mnuStop_Click()  
    Timer1.Interval=0  
    mnuStart.Enabled= True  
    mnuStop.Enabled= False  
End Sub
```

菜单 mnuColor 的响应代码如下：

```
Private Sub mnuColor_Click()  
    Form1.ForeColor=QBColor(11)  
End Sub
```

上述 QbColor 是一个控制颜色的函数，具体颜色取决于它的参数值，这个值的取值范围为 0~15。

定时器控件的 Timer 事件如下：

```
Private Sub Timer1_Timer()  
Static a as Integer  
a=a+50  
Line (0,a)-(a,a)  
End Sub
```

最后设置弹出菜单：

```
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single,  
Y As Single)  
If Button = 2 Or vbRightButton Then  
PopupMenu mnuControl  
End If  
End Sub
```

使用弹出式菜单的操作结果和直接使用下拉式菜单的操作结果是一样的。

8.2 对话框的设计

Windows 应用程序中的对话框主要起到显示信息和提示用户输入运行程序所必需的数据等功能，主要包括两种：自定义对话框和通用对话框。

自定义对话框与普通窗体的区别是它基本不包括菜单栏、工具栏、最小和最大化按钮、状态条和窗体滚动条。比如，Windows 中常见的【关于】对话框。

通用对话框在 Visual Basic 中被制作成为 ActiveX 控件，这一控件可利用 Windows 的资源，进行打开、保存文件和设置字体和颜色及设置打印机等操作。如果自己编写了帮助文档，还可通过它进行显示。通过使用通用对话框控件，编程人员可以轻松的把 Windows 的标准对话框加入到自己的应用程序中来。

8.2.1 自定义对话框

自定义对话框是由用户自己创建的含有控件的窗体，它与普通窗体的区别主要在于自定义对话框没有控制菜单框、最大化和最小化按钮，边框不能改变。

一般情况下，对话框有两个命令按钮：【确定】和【取消】。单击前者后开始执行相应的动作，单击后者退出该对话框；前者的 Default 属性设置为 True，后者的 Cancel 属性设置为 True。虽然这两个按钮是最常用的，但其他的按钮标题组合也可以使用。比如图 8.4 是 Visual Basic 6.0 编辑菜单中的查找对话框，它有 4 个命令按钮，没有最小化和最大化按钮，也没有控制菜单，不能改变窗口的大小。

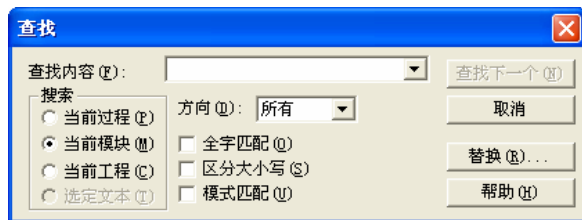


图 8.4 “查找”对话框

显示对话框是使用 Show 方法。其一般格式如下：

```
Show [style][,owner]
```

其中：owner 参数确保对话框在它的父窗体最小化时也被最小化。

对话框分为两种类型：模式和无模式的。不同对话框使用的 Show 方法的参数是不同的。

模式对话框一般用来显示重要消息，它在可以继续操作应用程序的其他部分之前，必须先被关闭。例如，一个对话框在可以切换到其他窗体或对话框之前，要求单击【确定】或【取消】按钮，则它是模式的。Visual Basic 中的通用对话框就是模式的。显示模式对话框的一般格式如下：

```
[对象].show vbModal
```

其中：对象为某个窗体的名称。

例如，frmAbout.Show vbModal 就是将名为 frmAbout 的窗体作为模式对话框来显示。

无模式对话框一般用来显示频繁使用的命令和信息，它允许在对话框和其他窗体之间转移焦点，而不必关闭对话框。当对话框正在显示时，可以在当前应用程序的其他地方继续工作。显示无模式对话框的一般格式如下：


```
[对象].show
```

例如，frmAbout.Show 就是将名为 frmAbout 的窗体作为无模式对话框来显示。

8.2.2 通用对话框

使用通用对话框代替自定义对话框可使得应用程序更加专业化，也可减少编程人员所花费的编程时间。

一般情况下，在建立新的工程后，工具栏中不会出现常用对话框控件的图标，可以通过在工具栏中添加新控件的方法把常用对话框控件添加到工具栏中。

为了使用 CommonDialog 控件，应将其添加到窗体上并设置属性。设计时 CommonDialog 控件在窗体上将显示为一个图标，它的大小不能改变，在程序运行时不可见。

双击控件工具栏上的通用对话框控件的图标，即可在窗体上加载该控件。保持该控件的选中状态，在属性窗口中选择【自定义】选项，单击出现的按钮，即可打开属性页。可以看到，属性页的内容包括【打开/另存为】、【颜色】、【字体】、【打印】和【帮助】

5 个选项, 可以根据提示对属性页进行属性设置。

该控件运行时所显示的对话框由控件的 Show 方法决定, 不同的 Show 方法对应的对话框的类型如表 8-6 所示。表中的前 4 种方法比较常用。

表 8-6 Show 方法对应的对话框的类型

方 法	显示的对话框的类型
ShowOpen	显示“打开”对话框
ShowSave	显示“另存为”对话框
ShowColor	显示“颜色”对话框
ShowFont	显示“字体”对话框
ShowPrinter	显示“打印机”或“打印选项”对话框
ShowHelp	调用帮助文件

1. 【打开】对话框

【打开】对话框是在应用程序中显示一个带有驱动器、目录和文件名的选择框, 主要进行打开文件操作。只要使用 CommonDialog 的 ShowOpen 方法就会弹出该对话框。如图 8.5 所示。

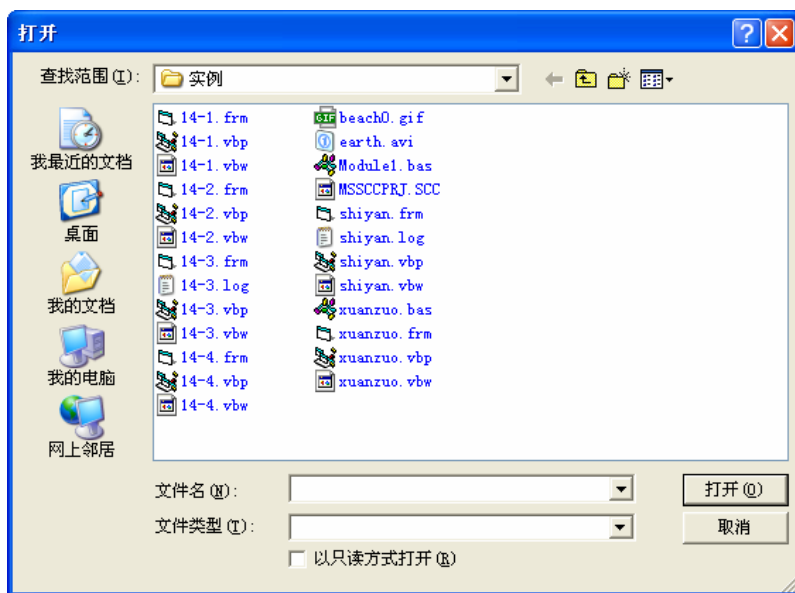


图 8.5 【打开】对话框

下面介绍【打开】对话框的常用属性。

(1) Filter

指定在【文件类型】下拉列表中显示的文件过滤器列表。设置格式:

描述 1 | 过滤器 1 | 描述 2 | 过滤器 2

其中，描述是指列表框中显示的字符串，例如：“位图文件(*.bmp)”，过滤器是指实际的文件类型“*.bmp”，描述和过滤器之间要用“|”符号来分隔。

(2) FileName

用户在对话框中选择打开的文件的路径名。

(3) DialogTitle

设置对话框的标题。

(4) InitDir

设置对话框的初始化文件目录。

(5) CancelError

设置用户选择【取消】按钮时是否发生错误。

以下代码显示【打开】对话框并将用户选定的文件名显示在标签控件中。

```
Private Sub mnuOpenFile_Click()  
With CommonDialog1  
    .CancelError = False  
    .DialogTitle = "打开"  
    .Filter = "所有文件(*.*)|*.*"  
    .ShowOpen  
    Label1.Caption = "打开的文件名为: " & .FileName  
End With  
End Sub
```

【另存为】对话框的属性与【打开】对话框基本一致，这里不再介绍。

2. 【颜色】对话框

在应用程序中需要调用调色板选择颜色或自定义颜色时可调用【颜色】对话框，如图 8.6 所示，用户选定颜色后，可用 Color 属性获取选定的颜色。



图 8.6 【颜色】对话框

以下代码将用户从【颜色】对话框中选择的颜色作为窗体的背景颜色。

```
Private Sub mnuColor_Click()  
With CommonDialog1  
    .Flags = cdlCCRGBInit
```

```

.ShowColor
Form1.BackColor= . Color
End With
End Sub

```

CommonDialog 的【颜色】对话框的 Flag 属性有 4 个标志，cdlCCRGBInit 设置初始颜色值，cdlCCFullOpen 显示所有自定义颜色部分，cdlCCPreviousFullOpen 使【规定自定义颜色】按钮无效，cdlCCShowHelp 将帮助按钮显示在对话框中。

3. 【字体】对话框

使用公用对话框的 ShowFont 方法可显示【字体】对话框，如图 8.7 所示，用户可根据大小、颜色、式样等选择字体。用户选定字体后，编程人员可根据表 8-7 所示的属性获取用户所选的字体信息。

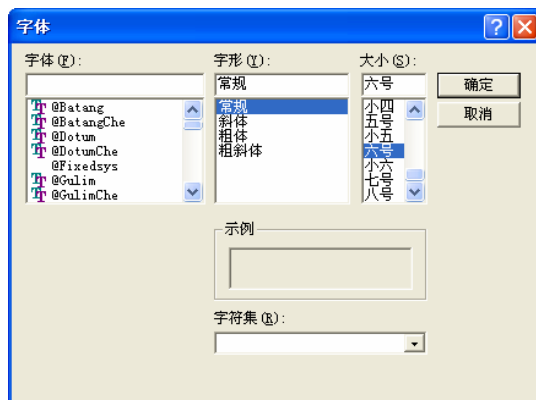


图 8.7 【字体】对话框

表 8-7 【字体】对话框返回的属性

属 性	功 能
Color	选定的字体颜色。要使用此属性必先将 Flags 属性设置为 cdlCFEffects
FontBold	是否设置字体为“粗体”
FontItalic	是否设置字体为“斜体”
FontName	选定字体的名称
FontSize	选定字体的大小
FontStrikeThru	是否设置字体有“删除线”。要使用此属性必先将 Flags 属性设置为 cdlCFEffects
FontUnderLine	是否设置字体有“下划线”。要使用此属性必先将 Flags 属性设置为 cdlCFEffects

在显示【字体】对话框之前，必须将 Flags 属性设置为以下 Visual Basic 常数之一，否则将出现“字体不存在”错误。

- cdlCFScreenFonts—&H1：屏幕字体。
- cdlCFPrinterFonts—&H2：打印机字体。
- cdlCFBoth—&H3：包括屏幕字体和打印机字体。

新建窗体，添加一个标签控件和一个按钮控件。在按钮控件的单击事件中添加如下代码：

```
Private Sub Command1_Click()  
With CommonDialog1  
    .CancelError = True  
    On Error GoTo errhandler  
    .Flags = cdlCFBoth Or cdlCFEffects  
    .ShowFont  
    Label1.FontName = .FontName  
    Label1.FontSize = .FontSize  
    Label1.FontBold = .FontBold  
    Label1.FontItalic = .FontItalic  
    Label1.FontStrikethru = .FontStrikethru  
    Label1.FontUnderline = .FontUnderline  
    Label1.ForeColor = .Color  
End With  
ErrHandddler:  
Exit Sub  
End Sub
```

此例中用用户在【字体】对话框中选择的选项来设置标签控件的字体属性。

4. 【打印】对话框

【打印】对话框由通用对话框的 ShowPrinter 方法调用，如图 8.8 所示。它可设置打印页的范围、打印质量、打印份数等项目。不过该对话框并不将数据直接送到打印机，用户可以指定打印数据的方式，但要编写代码才能真正实现打印。

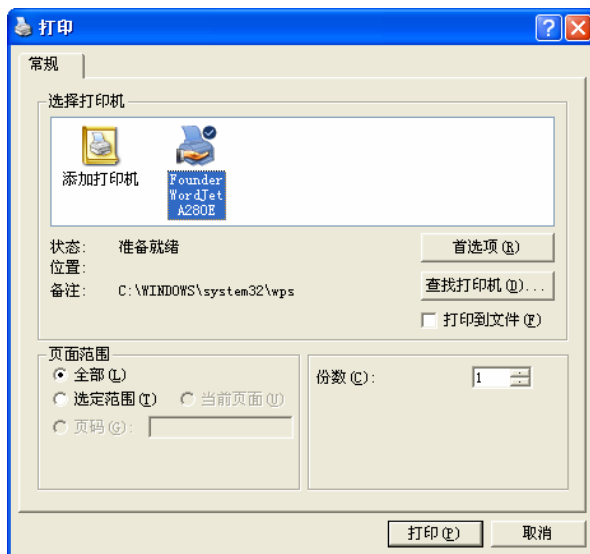


图 8.8 【打印】对话框

用户在对【打印】对话框的各项作出选择后，编程人员可通过表 8-8 所列出的属性得到用户选择的信息。

表 8-8 【打印】对话框返回值的属性

属 性	功 能
Copies	要打印的份数
FromPage	打印的起始页
ToPage	打印的终止页
Hdc	所选打印机的设备描述标识号

【打印】菜单的实现有些复杂，这里只给出了简单框架，有兴趣的读者可参考相关资料。

5. “帮助”文件的显示

使用通用对话框的 ShowHelp 方法可调用 Windows 帮助引擎，显示帮助文件。

首先，设置 HelpFile 和 HelpCommand 属性，然后调用 ShowHelp 方法显示指定的帮助文件。以下代码在单击帮助菜单项时显示指定的帮助文件。

```
Private Sub mnuHelp_Click()  
    With CommonDialog1  
        .HelpFile = "winhlp32.hlp"  
        .HelpCommand = cdlHelpContents  
        .ShowHelp  
    End With  
End Sub
```

8.2.3 通用对话框控件的使用

【例 8.2】调用通用对话框控件的【打开】对话框时，能够同时选中多个文件，并返回所有选中的文件的文件名和路径。将通用对话框控件的 Flags 属性设置为常量值 cdlOFNAllowMultiselect，它指定文件名列表框允许多重选择。运行时，按住 Ctrl 键的同时单击每个文件名称来选择多个文件。完成此操作后，其 FileName 属性就返回一个包含全部所选文件名的字符串。允许选择多文件后，CommonDialog 控件返回的 Filename 是按以下格式的一个字符串：

文件路径 文件名 1 文件名 2 文件名 3 ……

该文件的完整路径是“文件路径+文件名”，所以需要通过对字符串的操作把完整的文件名分离出来。

程序设计步骤如下。

(1) 新建“标准 EXE”工程。

(2) 建立程序用户界面。在窗口上添加 2 个命令按钮，分别修改其 Caption 属性为“打开文件”和“退出”。

(3) 进入代码窗口中, 编写如下事件过程:

```
Private Sub cmdExit_Click()
    Unload Me
End Sub
Private Sub cmdOpen_Click()
    Dim i As Integer
    Dim n As Integer
    Dim z As Integer
    Dim strFile() As String          '存储文件名的数组
    With CommonDialog1
        .Filter = "All Files|*.*"
        .Flags = cdlOFNAllowMultiselect
        .ShowOpen
        .FileName = .FileName & Chr(32) '为文件名字符串尾部添加一个空格
        z = 1
        While i <= Len(.FileName)
            i = InStr(z, .FileName, Chr(32)) '找到空格在文件名字符串中开始的位置
            If i = 0 Then Exit Sub
            ReDim Preserve strFile(n)
            strFile(n) = Mid(.FileName, z, i - z)
            '取得上一次空格和此次空格之间的字符串, 若是第一次则得到文件路径, 否则得到文件名
            n = n + 1
            i = i + 1
            z = i
        Wend
        List1.Clear
        If n = 1 Then
            List1.AddItem strFile(0)
        Else
            For i = 1 To n - 1
                List1.AddItem strFile(0) & "\" & strFile(i)
            Next i
        End If
    End With
End Sub
```

8.3 状态栏的设计

状态栏实际上是一个窗口, 一般分为几个窗格, 每个窗格显示不同的信息。状态栏可以用来显示状态栏提示和 Caps Lock、Num Lock、Scroll Lock 键的状态。

可以利用状态栏的属性页来编辑状态栏。选中窗体上的状态栏, 在鼠标右键菜单中选择“属性 Properties”, 这时会弹出如图 8.9 所示的对话框, 打开【窗格】选项卡。其中插入窗格按钮用来增加状态栏中的窗格数目, 当添加一个窗格后“索引”的值也会自动加 1, 在程序中可以通过窗格的索引值来引用窗格。状态栏的常用属性如表 8-9 所示。



图 8.9 状态栏的属性页

表 8-9 状态栏的常用属性

属 性	说 明
Text	在各个窗格中显示的文字
Alignment	窗格中文本的对齐方式
Style	窗格的样式(0-sbrText 为显示文本，6-sbrDate 为显示日期等)。

状态栏的 Panels 集合提供了 Add 方法，用来在程序中动态添加窗格，其一般格式如下：

```
Object.Panels.Add [Index],[Key],[Text],[Style]
```

其中：

- Object 为状态栏对象
- Index 为可选项。是要添加窗格的索引号。
- Key 为可选项。是要添加窗格的关键字。
- Text 为可选项。是要添加窗格显示的文字。
- Style 为可选项。是要添加窗格的类型，默认状态为普通窗格，取值为 0~6 的整数。分别表示系统时间日期、键盘状态等。

下面通过一个例子来学习状态栏的用法。

【例 8.3】 用状态栏显示当前系统日期时间、键盘状态以及不同窗格样式。程序运行界面如图 8.10 所示。

新建工程，在窗体上添加的控件及其属性设置如表 8-10 所示。



图 8.10 状态栏示例

表 8-10 例 8.3 中状态栏的属性设置

控件名称	属 性	取 值
Label1(0)	Caption	“显示系统信息(系统时间或日期等)”
Label1(1)	Caption	“显示键盘状态(Caps Lock 或 Num Lock)”
Label1(2)	Caption	“显示文本和图片信息”
Label2(0)	Caption	“状态栏的用途包括:”
Label2(1)	Caption	“状态栏的三种式样:”
StatusBar1	SimpleText	“状态栏测试”

程序代码如下:

```

Private Sub Label1_Click(Index As Integer)
With StatusBar1
Select Case Index
Case 0
.Panels.Clear
.Panels.Add , , , 5
.Panels.Add , , "系统日期: ", 6
.Panels.Add
.Panels(1).Alignment = sbrCenter
.Panels(2).Alignment = sbrCenter
Case 1
'.Panels.Clear
.Panels.Add , , "字母大小写", 1
.Panels.Add , , , 2
.Panels.Add , , , 3
.Panels.Add , , , 4
.Panels(1).Alignment = sbrCenter
.Panels(2).Alignment = sbrCenter

```

```
.Panels(3).Alignment = sbrCenter
.Panels(4).Alignment = sbrCenter
Case 2
.Panels.Clear
.Panels.Add
.Panels(1).AutoSize = sbrContents
.Panels(1).Picture = LoadPicture(App.Path & "\203.gif")
.Panels(1).Text = "文本式样"
End Select
End With
End Sub
Private Sub Label2_Click(Index As Integer)
If Index = 1 Then
With StatusBar1
.Panels.Clear
.Panels.Add
.Panels.Add
.Panels.Add
.Panels(1).Bevel = sbrInset
.Panels(2).Bevel = sbrNoBevel
.Panels(3).Bevel = sbrRaised
.Panels(1).Text = "凹陷窗格"
.Panels(2).Text = "平面窗格"
.Panels(3).Text = "凸出窗格"
End With
End If
End Sub
```

8.4 工具栏的设计

工具栏是由工具栏按钮对象组成的集合，在应用程序中，它提供了对最常用命令的快速访问。比如，Visual Basic 工具栏中就提供了诸如【复制】、【粘贴】、【剪切】等常用命令的工具栏按钮。

8.4.1 使用手工方式制作工具栏

在窗体上手工创建工具栏，通常是用 PictureBox 控件作为工具栏容器，用命令按钮控件或 Image 控件作为工具栏按钮。

手工创建工具栏的一般步骤如下。

(1) 在窗体或 MDI 窗体上放置 PictureBox 控件(工具栏按钮的容器)。若是普通窗体，则要将其 Align 属性置为 1，图片框才会自动伸展到窗体长度。若是 MDI 窗体，则不需要这一操作，它会自动伸展。需要注意的是 MDI 窗体上只能用图片框作为工具栏按钮的容器。

(2) 在 PictureBox 控件中放置 CommandButton 或 Image 控件作为工具栏的按钮。为工

具栏上显示的每个控件设置 **Picture** 属性, 指定一个图片。如果需要, 也可通过 **ToolTipText** 属性设置工具提示。

(3) 编写代码。工具栏经常用于提供对其他命令的快捷访问, 大部分时间都是调用对应的菜单项事件过程。

8.4.2 使用工具栏控件制作工具栏

由于工具栏中包含了一系列的图像按钮, 需要图片列表框控件(**ImageList**)来存储图片。如要在设计时设置图片, 与之关联的 **ImageList** 控件需与工具栏控件运行在同一窗体上。可利用 **ImageList** 控件的属性页来为其添加图片, 设置图片大小, 设置图片索引。之后, 要设置工具栏控件的图像列表属性为 **ImageList** 控件的名称, 使之关联。

1. 图片列表框控件

利用 **ImageList** 控件才能在工具栏上显示图片, 下面首先了解一下此控件。**ImageList** 控件包含在 **Microsoft Common Control 6.0** 中, 可为其他 **Windows** 控件保存图片, 也可将该控件的图片赋值给图片框等控件。

ImageList 控件可添加的图片大小任意, 不过在显示时大小都相同。一般来说, 以加入到该控件的第一幅图像的大小为标准。

ImageList 控件利用 **ListImage** 对象集合中的图片, 该对象具有集合对象的标准属性, 如 **Key**、**Index**、**Count**, 可用标准的集合方法如 **Add**、**Remove**、**Clear** 来操作。

也可使用 **ImageList** 控件的属性页(见图 8.11)设置该控件的属性。在将该控件与其他控件相关联之前, 要先向其中添加图片。

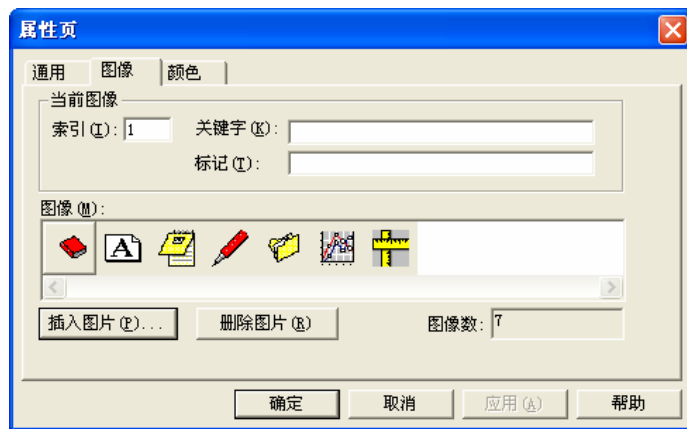


图 8.11 **ImageList** 控件的属性页

2. 工具栏(**ToolBar**)控件

用户使用 **ToolBar** 控件可以方便地在应用程序中创建工具条, 该控件提供了许多属性来定制工具栏, 表 8-11 介绍了工具栏控件的常用属性。

表 8-11 工具栏控件的常用属性

属 性	说 明
Align	决定工具栏在窗体上的位置。具体设置值见表 8-12
Buttons	返回一个 Button 对象的集合，可通过标准集合方法来操作。集合中的每个成员都可用其唯一索引(Index)或键值(Key)来表示
ImageList	返回与工具栏相关联的控件
ShowTips	设置或返回用户将鼠标放置在控件上时是否显示 ToolTips 提示信息。默认值为 True
Style	设置或返回工具栏式样，默认值为 0，即标准式样，工具栏上的按钮一直以 3D 效果显示。也可修改为平面式样，即当鼠标移到按钮上时，按钮才突出显示
TextAlignment	决定按钮文本显示在图片下方(0)，还是右侧(1)。默认值为 0，即文本在图片下方
Wrappable	设置或返回当窗体大小改变时，工具栏是否自动折行。默认值为 True

表 8-12 工具栏控件的 Align 属性设置值

Visual Basic 常数	取 值	说 明
vbAlignNone	0	无对齐方式。可在程序中设置位置，若在 MDI 窗体上，则忽略该设置
vbAlignTop	1	默认值。对象显示在窗体顶部
vbAlignBottom	2	对象显示在窗体底部
vbAlignLeft	3	对象显示在窗体左边
vbAlignRight	4	对象显示在窗体右边

工具栏由按钮组成，Buttons 集合中的 Button 对象组成了工具栏控件。在设计和运行时都可创建按钮对象，表 8-13 介绍了 Buttons 集合的常用属性。

表 8-13 Buttons 集合的常用属性

属 性	说 明
Caption	设置按钮上的文本
Count	返回集合中对象的个数
Enabled	设置按钮对象是否有效，默认值为 True
Image	设置按钮对象使用 ImageList 控件中哪个 ListImage 对象
Index	返回标识集合中对象的数值，从 1 开始
Item	通过 Index 或 Key 属性值来返回特定成员
Key	返回唯一标识集合中对象的字符串
Style	返回或设置按钮对象的式样。它决定了按钮的行为特点。取值为 0~5。默认值为 0—tbrDefault，普通按钮；1—tbrCheck，复选按钮；2—tbrButtonGroup，按钮组，组中一次只能有一个按钮按下；3—tbrSeparator，分隔符；4—tbrPlaceHolder，可变宽度分隔符；5—tbrDropDown，下拉菜单式按钮

续表

属 性	说 明
ToolTipText	返回按钮对象的工具提示信息
Value	返回或设置按钮对象是否处于按下或选中状态。默认值为 0—tbrUnPressed，表示没有被按下或选中；1—tbrPressed，表示按钮被按下或选中
Visible	设置按钮对象是否可见

也可打开工具栏控件的属性页进行设置，如图 8.12 所示。先在通用页将图像列表属性设置为要关联的 ImageList 控件名。然后在按钮页依次添加按钮并设置按钮标题，还要给每个按钮设置关键字，选择式样。

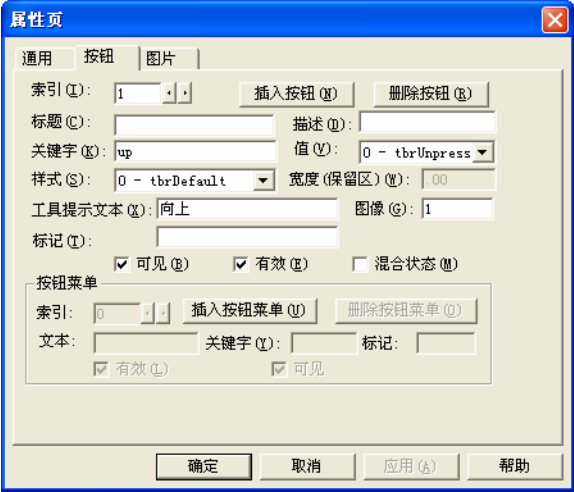


图 8.12 工具栏控件的属性页

ToolBar 控件的 ButtonClick 事件使用率很高，当用户单击按钮时，将激发该事件。此时，使用 Button 对象的索引(Index)或键值(Key)识别被单击的按钮。以下代码是如何编写该事件的一个实例。

【例 8.4】 工具栏控件的使用。运行界面如图 8.13 所示。

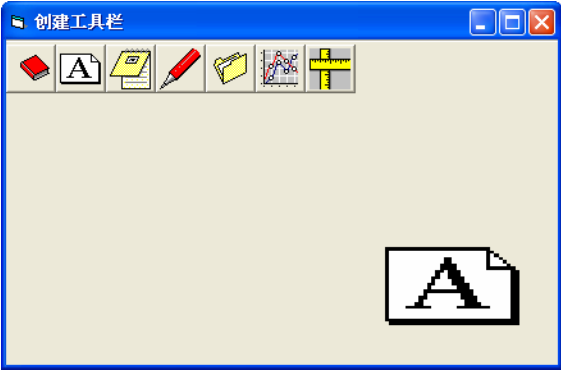


图 8.13 工具栏控件实例界面

新建“标准 EXE”工程。建立程序用户界面。在窗口上添加 1 个工具栏控件、1 个标签控件和 1 个 Image 控件。编写如下事件过程：

```
Private Sub Toolbar1_ButtonClick(ByVal Button As MSComctlLib.Button)
    Set Image1.Picture = ImageList1.ListImages(Button.Index).Picture
End Sub
```

8.4.3 文档编辑器的实现

【例 8.5】设计文档编辑器，此例利用 RichTextBox 控件进行文本编辑，RichTextBox 控件不仅允许输入和编辑文本，同时也提供了 TextBox 控件没有的、更高级的许多功能：字体和段落格式。我们称其为格式文本框。RichTextBox 控件能以 rtf(Rich Text File)格式打开或保存文件，可以直接读写文件，或使用 Visual Basic 的文件功能进行输入输出来打开和保存文件。它常被用来作为不同文字处理系统之间交流的载体。程序运行界面如图 8.14。

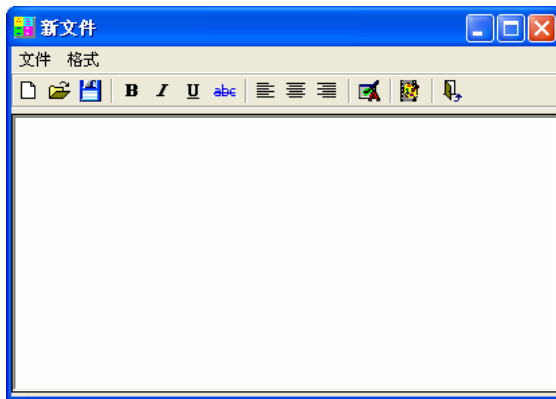


图 8.14 文档编辑器的界面

程序设计步骤如下。

- (1) 新建“标准 EXE”工程。
- (2) 建立程序用户界面。在窗口上添加表 8-14 所示的控件，并参照表 8-14 设置其属性。

表 8-14 文档编辑器的控件设置

对象类型	属 性	取 值
工具条	Name	Toolbar1
RichTextBox	Name	RtfText
图片框	Name	PicRuler
通用对话框	Name	CommonDialog1
图片列表	Name	imlToolbarIcons
CoolBar	Name	CoolBar1

续表

对象类型	属 性	取 值
菜单	Name	mnuFile
	Name	mnuNew
	Name	mnuOpen
	Name	mnuSave
	Name	mnuExit
	Name	mnuFormation
	Name	mnuFont
	Name	mnuColor

(3) 进入代码窗口中，编写如下事件过程：

```

Private Sub tbToolBar_ButtonClick(ByVal Button As MSComctlLib.Button)
    On Error Resume Next
    Select Case Button.Key
        Case "新建"
            mnuNew_Click
        Case "打开"
            mnuOpen_Click
        Case "保存"
            mnuSave_Click
        Case "粗体"
            '设置格式文本框的字体格式，与当前状态相反
            RtfText.SelBold = Not RtfText.SelBold
            Button.value = IIf(RtfText.SelBold, tbrPressed, tbrUnpressed)
        Case "斜体"
            RtfText.SelItalic = Not RtfText.SelItalic
            Button.value = IIf(RtfText.SelItalic, tbrPressed,
tbrUnpressed)
        Case "下划线"
            RtfText.SelUnderline = Not RtfText.SelUnderline
            Button.value = IIf(RtfText.SelUnderline, tbrPressed,
tbrUnpressed)
        Case "删除线"
            RtfText.SelStrikeThru = Not RtfText.SelStrikeThru
            Button.value = IIf(RtfText.SelStrikeThru, tbrPressed,
tbrUnpressed)
        Case "左对齐"
            RtfText.SelAlignment = rtfLeft '设置格式文本框的文字对齐属性
        Case "置中"
            RtfText.SelAlignment = rtfCenter
        Case "右对齐"
            RtfText.SelAlignment = rtfRight
    
```

```

        Case "字体"
            mnuFont_Click
        Case "颜色"
            mnuColor_Click
        Case "关闭"
            mnuExit_Click
    End Select
End Sub

```

以下是各个菜单的单击响应程序:

```

Private Sub mnuColor_Click()
    With CommonDialog1 '利用公共对话框控件打开系统颜色对话框
        .DialogTitle = "颜色"
        .CancelError = False
        .Flags = &H2
        .ShowColor
        RtfText.BackColor = .Color '设置格式文本框背景颜色为用户选中的颜色
    End With
End Sub

Private Sub mnuExit_Click()
    Dim result As Integer
    If Me.Caption = "新文件" And RtfText.Text = "" Then '判断用户是否修改了文件
        Unload Me
    Else '若用户修改了文件,则弹出消息框询问用户是否要保存文件
        result=MsgBox("您已经对文件作了修改,要保存吗?",vbYesNoCancel,"提示")
        Select Case result
            Case 6 '用户单击了 Yes 按钮
                mnuSave_Click
            Case 2 '用户单击了 Cancel 按钮
                Exit Sub
            Case 7 '用户单击了 No 按钮
                Unload Me
        End Select
    End If
End Sub

Private Sub mnuFont_Click()
    CommonDialog1.Flags = cdlCFBoth + cdlCFEffects
    CommonDialog1.ShowFont
    '利用公共对话框控件打开字体对话框,根据用户选择的字体设置格式文本框的字体属性
    With RtfText
        .SelFontName = CommonDialog1.FontName
        .SelFontSize = CommonDialog1.FontSize
        .SelBold = CommonDialog1.FontBold
        .SelItalic = CommonDialog1.FontItalic
        .SelStrikeThru = CommonDialog1.FontStrikethru
        .SelUnderline = CommonDialog1.FontUnderline
    End With
End Sub

```

```

Private Sub mnuNew_Click()
    RtfText.Text = "" '将格式文本框中的内容清空，修改窗体标题栏
    Me.Caption = "新文件"
End Sub

Private Sub mnuOpen_Click()
    Dim sfile1 As String
    With CommonDialog1 '利用公共对话框控件打开系统对话框
        .DialogTitle = "打开"
        .CancelError = False
        .Filter = "Rich Text 文件 (*.rtf)|*.rtf"
        .ShowOpen
        If Len(.FileName) = 0 Then Exit Sub
        sfile1 = .FileName '取得要打开文件的名字
    End With
    RtfText.LoadFile sfile1
    Me.Caption = sfile1
End Sub

Private Sub mnuSave_Click()
    With CommonDialog1 '利用公共对话框控件打开保存对话框
        .DialogTitle = "保存"
        .CancelError = False
        .Filter = "Rich Text 文件 (*.rtf)|*.rtf"
        .ShowSave
    End With
    If Len(.filename) = 0 Then Exit Sub
    sfile2 = .filename '取得保存新文件的名字
    RtfText.SaveFile sfile2
End Sub

```

8.5 多文档界面设计

在前面各章中提及的例子都是在一个窗体中完成的，而复杂一些的应用程序一般需要多个窗体才能完成，这就需要使用多窗体界面和多文档界面。本节主要介绍对多文档用户界面的创建。

8.5.1 多文档界面(MDI)

多文档用户界面允许在单个容器窗口中包含多个子窗口，同时处理多个文档，每个文档都显示在各自的窗口中，比如 Microsoft Excel 就是这样的应用程序，如图 8.15 所示。

MDI 应用程序的子窗体包含在父窗体中，父窗体为应用程序的子窗体提供工作空间，每个子窗体都限制在父窗体范围之内，最小化父窗体时，所有子窗体也被最小化，但只有父窗体的图标显示在任务栏上，子窗体最小化时，它的图标显示在父窗体的工作区内，而不是任务栏中。

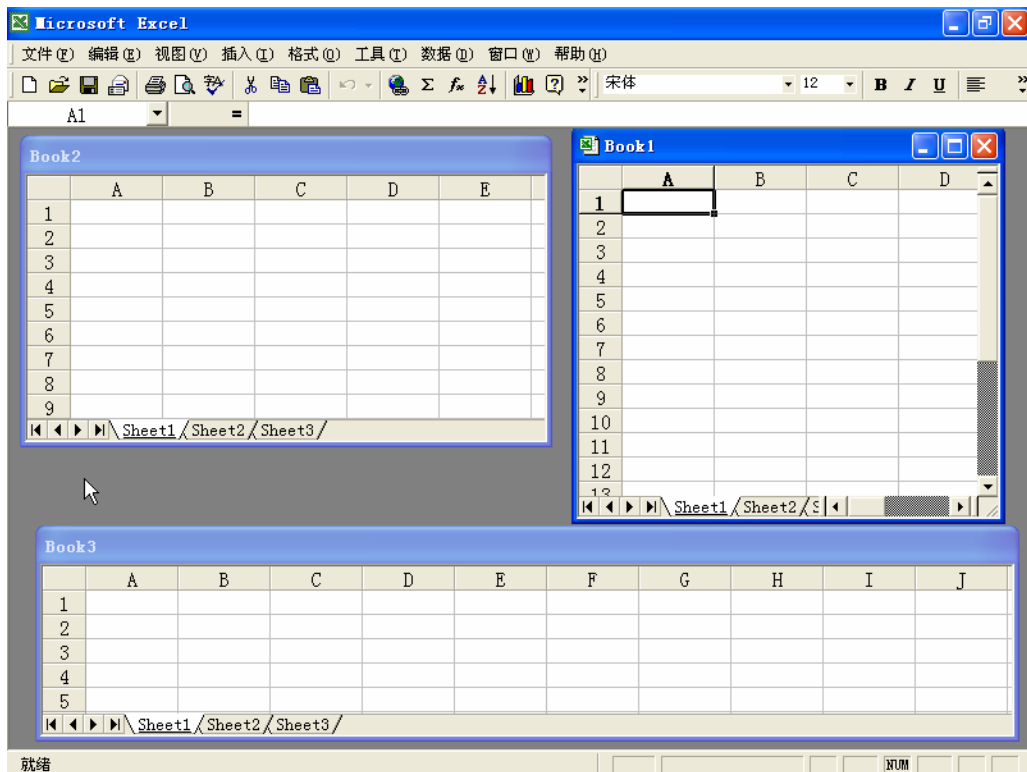



图 8.15 Excel 多文档用户界面

8.5.2 建立多文档界面

1. 创建 MDI 窗体

在工具栏中单击图标 ，在下拉菜单中选择【添加 MDI 窗体】。一个应用程序中只能有一个 MDI 窗体，若已经在工程中添加了 MDI 窗体，则该命令不可用。

2. 创建 MDI 子窗体

首先创建一个普通窗体，然后将其 MDIChild 属性设置为 True，此时该窗体成为一个 MDI 子窗体。在子窗体上可像在普通窗体上一样添加控件，设置属性。

在工程资源管理器中，MDI 窗体和 MDI 子窗体有确定的显示图标，如图 8.16 所示。



图 8.16 工程资源管理器

子窗体能够移动和改变大小,但只能在父窗体的工作空间之内。最大化一个子窗体时,它的标题会与父窗体的标题组合,显示在父窗体的标题栏上。子窗体若有菜单,将显示在父窗体的菜单栏中。

3. 指定活动子窗体

使用 MDI 窗体的 `ActiveForm` 属性,可返回具有焦点或最后激活的子窗体,当然,使用该属性时,应至少有一个子窗体已被加载或可见。

当一个窗体中设置了几个控件时,也可指定活动控件。使用 `ActiveControl` 属性可返回子窗体上具有焦点的控件。

4. 加载 MDI 窗体和子窗体

加载父窗体时,子窗体并不会自动显示,可使用 `AutoShowChildren` 属性加载隐藏状态的子窗体。加载子窗体时父窗体会自动加载。

5. 用 `QueryUnload` 事件卸载 MDI 窗体

`QueryUnload` 事件在 MDI 窗体卸载之前被调用,然后每个打开的子窗体调用该事件。若此事件没有代码,则先卸载子窗体,然后再卸载 MDI 窗体。该事件使得编程人员有机会在窗体卸载之前询问用户是否要保存窗体。例如:子窗体中有一文本框,可设置一个全局标志来记录用户是否修改了文本框中的内容,然后在 `QueryUnload` 事件中检查此标志,若用户修改了文本框中的内容,则询问用户是否保存。

8.5.3 创建 MDI 应用程序的菜单

在 MDI 应用程序中,每一个子窗体的菜单都显示在 MDI 窗体上,而不是在子窗体本身。当子窗体拥有焦点时,该子窗体的菜单(如果有的话)就代替菜单栏上的 MDI 窗体的菜单。如果没有可见的子窗体,或者如果带有焦点的子窗体没有菜单,则显示 MDI 窗体的菜单。

如果子窗口有一组菜单项,需要这些菜单项都出现在父窗口的主菜单中,则可以创建作为子窗口窗体的主菜单。当激活子窗口时,其主菜单与父窗口的主菜单合并。

1. 创建【窗口】菜单

通常,MDI 应用程序有一个【窗口】菜单,使得用户得以通过平铺或层叠排列已经打开的子窗口。【窗口】菜单还使用户得以定位到任何打开的子窗口。

创建【窗口】菜单,首先要将平铺和层叠的菜单项添加到父窗体的【窗口】菜单中。若要在某个菜单上显示所有打开的子窗体标题,则可利用菜单编辑器将其 `WindowList` 属性设置为 `True`,即显示打开的子窗体标题。

2. 排列子窗体

应用程序常包含对打开的 MDI 子窗体进行操作的菜单命令,如【平铺】、【层叠】和【排列】。可以使用 `Arrange` 方法来层叠、平铺和排列子窗体的图标;这些枚举值将子窗体显示为层叠、水平平铺或垂直平铺,或者显示为排列在 MDI 窗体下部的子窗体图标。

假定 MDI 窗体名称为 frmMDI，则以下语句分别实现了不同的子窗体排列。

- 层叠子窗体：frmMDI.Arrange vbCascade。
- 平铺子窗体：frmMDI.Arrange vbTileHorizontal。
- 排列图标子窗体：frmMDI.Arrange vbArrangeIcons。

其中，vbCascade、vbTileHorizontal 和 vbArrangeIcons 是 Visual Basic 的内部常数。

8.6 Visual Basic 的工程结构

Visual Basic 的工程主要包括窗体模块、标准模块、类模块等几个部分。

1. 标准模块

标准模块具有 .bas 文件扩展名，可以包含类型、常数、变量、外部过程和公共过程的公共的或模块级的声明。

2. 窗体模块

窗体模块具有 .frm 文件扩展名，一般包含窗体及其控件的正文描述，包括它们的属性设置。它们也含有窗体级的常数、变量和外部过程的声明、事件过程和一般过程。

3. 类模块

类模块具有 .cls 文件扩展名，与窗体模块类似，只是没有可见的用户界面。可以使用类模块创建含有方法和属性代码的自己的对象。

4. SubMain 过程

在代码模块中使用 SubMain 过程，应用程序会马上开始执行程序，而不必等待窗体上发生事件。一般来说，要在标准模块中创建一个名为 SubMain 的过程，可在其中加入启动应用程序时需执行的代码，然后通过【工程】→【工程属性】→【通用】选项卡→【启动对象】选择 Sub Main。这样在启动应用程序时，就可以首先执行名为 Main 的子过程中的内容。这个子过程不能在窗体模块内，必须在标准模块内创建。

8.7 小 结

在 Windows 环境下运行的应用程序，虽然不是所有的用户界面看上去或操作起来都一样，但是这些界面有许多共同的特点。它们都包括一些基本要素，如菜单、工具栏、状态栏，在应用程序的操作过程中又往往要打开一些对话框。使用 Visual Basic 开发 Windows 环境下的应用程序，同样离不开这些常见的界面元素的设计。本章介绍了如何开发 Windows 应用程序中的一些典型的界面要素，包括菜单、工具栏、状态栏、各种对话框以及多文档界面的设计。

使用 Visual Basic 设计应用程序时，当操作比较简单，且操作命令较少时，可以直接通

过控件来执行；而当要完成比较复杂且命令选项比较多的操作时，使用菜单操作会更为方便。

菜单可以分成两种基本类型：下拉式菜单和弹出式菜单。下拉式菜单通常通过单击菜单标题打开，弹出式菜单通常在菜单区域通过单击鼠标右键打开。

Visual Basic 6.0 通过菜单编辑器来进行菜单的设计，用菜单编辑器可以创建新的菜单和菜单栏。

工具栏是许多基于 Windows 的应用程序的标准功能，它提供了对应用程序中常用的菜单命令的快速访问。在 Visual Basic 中设计工具栏有两种方法：手工设计和使用工具栏控件进行设计。

Visual Basic 为创建工具栏提供了一个 ActiveX 控件——ToolBar 控件，使用该控件创建工具栏更方便、快捷，创建出的工具栏与 Windows 工具栏风格更加统一。

对话框主要起到显示信息和提示用户输入运行程序所必需的数据等功能，主要包括两种：自定义对话框和通用对话框。

多文档用户界面允许创建在单个容器窗口中包含多个窗口，同时处理多个文档，每个文档都显示在各自的窗口中。

8.8 习 题

一、选择题

1. 下列有关子菜单的说法中，错误的是()。
 - A) 每个菜单项都是一个控件，与其他控件一样也有其属性和事件
 - B) 除了 Click 事件之外，菜单项不可以响应其他事件
 - C) 菜单的索引号可以不连续
 - D) 菜单项的索引号必须从 1 开始
2. 以下叙述中错误的是()。
 - A) 下拉式菜单和弹出式菜单都用菜单编辑器建立
 - B) 在多窗体程序中，每个窗体都可以建立自己的菜单系统
 - C) 在程序运行过程中可以增加或减少菜单项
 - D) 如果把一个菜单项的 Enabled 属性设置为 False，则可删除该菜单项
3. 以下说法正确的是()。
 - A) Visual Basic 中的对话框分为两种类型：预定义对话框和自定义对话框
 - B) 预定义对话框是用户在设置程序代码前定义的
 - C) 自定义对话框是由用户根据自己的需要定义的
 - D) 以上说法都不正确
4. 设菜单中有一个菜单项为 Open。若要为该菜单项设计访问键，即按下 Alt 键及字母 O 时，能够执行 Open 命令，则在菜单编辑器中设置 Open 命令的方式是()。
 - A) 把 Caption 属性设置为&Open

- B) 把 Caption 属性设置为 O&pen
- C) 把 Name 属性设置为&Open
- D) 把 Name 属性设置为 O&pen

5. 以下关于多重窗体程序的叙述中, 错误的是()。

- A) 用 Hide 方法不但可以隐藏窗体, 而且能清除内存中的窗体
- B) 在多重窗体程序中, 各窗体的菜单是彼此独立的
- C) 在多重窗体程序中, 可以根据需要指定启动窗体
- D) 在多重窗体程序中, 而且单独保存每个窗体

6. 在窗体上画一个名称为 CommandDialog1 的通用对话框和一个名称为 Command1 的命令按钮。然后编写如下事件过程:

```
Private Sub Command1_Click()  
CommonDialog1.FileName = ""  
CommonDialog1.Filter="All file|*.*|(*.Doc)|*.Doc|(*.Txt)|*.Txt"  
CommonDialog1.FilterIndex=2  
CommonDialog1.DialogTitle="VBTest"  
CommonDialog1.Action=1  
End Sub
```

对于这个程序, 以下叙述中错误的是()。

- A) 该对话框被设置为“打开”对话框
- B) 在该对话框中指定的默认文件名为空
- C) 该对话框的标题为 VBTest
- D) 在该对话框中指定的默认文件类型为文本文件(*.Txt)

7. 在窗体上画一个名称为 CommonDialog1 的通用对话框, 一个名称为 Command1 的命令按钮。要求单击命令按钮时, 打开一个打开文件的通用对话框。该窗口的标题为 Open, 在【文件类型】栏中显示*.bmp。能够满足上述要求的程序是()。

```
A) Private Sub Command1_Click()  
CommonDialog1.Filter = "所有文件|*.*|(*. bmp)|*. bmp "  
CommonDialog1.DialogTitle = " Open "  
CommonDialog1.ShowOpen  
End Sub  
B) Private Sub Command1_Click()  
CommonDialog1.Filter = "所有文件|*.*|(*. bmp)|*. bmp "  
CommonDialog1.DialogTitle = "Save"  
CommonDialog1.ShowSave  
End Sub  
C) Private Sub Command1_Click()  
CommonDialog1.FileName = "Save"  
CommonDialog1.Filter = "所有文件|*.*|(*. bmp)|*. bmp "  
CommonDialog1.DialogTitle = " Open "  
CommonDialog1.ShowOpen  
End Sub  
D) Private Sub Command1_Click()  
CommonDialog1.Filter = "所有文件"
```



```
CommonDialog1.DialogTitle = " Open "  
CommonDialog1.ShowOpen  
End Sub
```

二、填空题

在菜单编辑器中建立一个菜单，菜单项的名称为 mnuColor，Visible 属性为 False，程序运行后，如果用鼠标右键单击窗体，则弹出与 mnuColor 相应的菜单。以下是实现上述功能的程序，请填空。

```
Private Sub Form _ _____ (Button As Integer, Shift As Integer,  
X As Single, Y As Single)  
If Button=2 Then  
_____ mnuColor  
End If  
End Sub
```

三、编程题

1. 设计一个具有弹出式和下拉式菜单栏和工具栏的窗体，单击菜单或工具栏图标能够打开不同窗体。
2. 利用通用对话框控件编写一个应用程序，界面如图 8.17 所示。要求实现以下功能。
 - (1) 单击【打开】按钮，显示标准的打开文件对话框，对话框标题为“请选择文件”，默认路径为“d: \VbTest(或所用机器中的一个已经存在的文件夹)，默认列出的文件扩展名为.txt 文件，选定路径及文件名后，该路径和文件名显示在文本框中。
 - (2) 单击【字体】按钮，显示标准的【字体】对话框，利用该对话框设置文本框中文字的字体、字体样式、大小、效果和颜色。
 - (3) 单击【颜色】按钮打开【颜色】对话框，用于设定文本框的背景颜色。

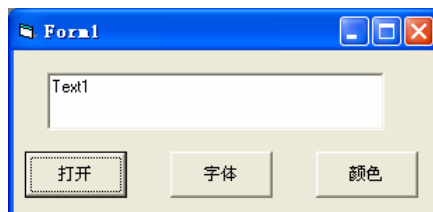


图 8.17 编程题 2 的效果图

第 9 章 Visual Basic 图形设计

教学提示：Visual basic 为程序设计者提供了非常丰富的绘图功能。设计程序时，不仅可以使用 Visual Basic 提供的图形控件来画图，还可以调用图形方法绘制丰富多彩的艺术图形。

教学目标：了解与绘图有关的基础知识，如坐标系、绘图的颜色等，掌握用图形控件、图形函数、语句、方法进行图形设计的基本方法和技巧。

9.1 图形设计基础

9.1.1 坐标系

坐标对图形绘制与文字输出的操作而言是非常重要的。坐标系统包括原点位置、坐标单位以及坐标轴的方向等几方面内容。

在窗体和图片框中绘图时，Visual Basic 提供了 7 种标准坐标系统供选用。此外，还可以使用自定义坐标系统。

1. 标准坐标系统

Visual Basic 的坐标系用于在二维空间定义容器对象(如窗体和图片框)中点的位置。和数学中的坐标系一样，Visual Basic 的坐标系也包含坐标原点、x 坐标轴和 y 坐标轴。Visual Basic 坐标系的默认坐标原点(0, 0)在容器对象的左上角，水平方向的 x 坐标轴向右为正方向，垂直方向的 y 坐标轴向下为正方向，坐标轴的默认刻度单位是缇(Twip)。

除了缇外，Visual Basic 中坐标的常用度量单位还有点(Point)、厘米和英寸等。1 缇大约为 1/1440 英寸，1/20 点。1 厘米约等于 567 缇，1 英寸约等于 1440 缇。72 点等于 1 英寸。用户可以根据实际需要，使用 ScaleMode 属性改变坐标系的刻度单位。ScaleMode 属性的取值如表 9-1 所示。

表 9-1 ScaleMode 属性的取值

值	常 量	描 述
0	vbUser	自定义坐标系统
1	vbTwips	默认值，坐标单位为缇
2	vbPoints	坐标单位为点(每英寸为 72 点)
3	vbPixels	像素(显示器分辨率的最小单位)

续表

值	常 量	描 述
4	vbCharacters	字符坐标系统(水平每个单位为 120 缇, 垂直每个单位为 240 缇)
5	vbInches	坐标单位为英寸
6	bMillimeters	坐标单位为毫米
7	vbCentimeters	坐标单位为厘米

当 ScaleMode 值为 1~7 时, 是 Visual Basic 已定义好的标准坐标系统, 这 7 种标准坐标系统的原点都在对象绘图区的左上角, 水平坐标的正方向为向右, 垂直坐标的正方向为向下。其中坐标系统 3(ScaleMode 属性的取值为 3)以像素为刻度度量单位, 在不同分辨率的显示器与打印机上输出的结果大小不同, 是设备相关的。其他几种标准坐标系统都是设备无关的。坐标系统 4(ScaleMode 属性的取值为 4)的水平与垂直度量不相同, 使用时应当注意。

2. 自定义坐标系统

当窗体或图片框的 ScaleMode 属性为 0 时, 使用自定义坐标系统, 自定义坐标系统的原点位置和坐标单位由 ScaleHeight、ScaleWidth、ScaleLeft 和 ScaleTop 这 4 个属性决定。

ScaleLeft、ScaleTop 属性分别指定在新坐标系统下对象绘图区左上角的水平和垂直坐标。

ScaleHeight、ScaleWidth 属性决定在新的坐标下窗体或图片框绘画区的高度与宽度。当这两个属性取负值时, 会改变坐标的正方向。在自定义坐标系统下, 坐标的实际单位是由对象的实际大小和 ScaleHeight、ScaleWidth 属性联合决定的。

例如, 假设当前窗体内部显示区域的高度是 2000 缇, 宽度是 3000 缇。此时高度和宽度的刻度单位均为 1 缇。

如果设置 ScaleHeight=500, 则将窗体内部显示区域的高度划分为 500 个单位, 每个单位为 2000/500, 即 4 缇。

如果设置 ScaleWidth=1000, 则将窗体内部显示区域的宽度划分为 1000 个单位, 每个单位为 3000/1000, 即 3 缇。

在使用标准坐标系统时, ScaleHeight、ScaleWidth、ScaleLeft 和 ScaleTop 这 4 个属性也是可用的, 它们分别反映出在当前标准坐标系统下相应的值, ScaleLeft、ScaleTop 的值都为 0, ScaleHeight、ScaleWidth 值为以当前坐标单位为度量标准的绘图区高度与宽度。如果在标准坐标系统下, 改变了这 4 个属性中任意一个的值, 会自动使用自定义坐标, ScaleMode 属性被自动设置为 0。

【例 9.1】 将一个图片(Picture1)的左上角移动到窗体的中央位置。

如果不对窗体重新定义刻度单位, 就需要以缇为单位来计算窗体的中央位置, 如果将窗体的高度和宽度都定义为 2 个刻度单位, 则窗体的中央位置即为坐标为(1, 1)的点。设计界面如图 9.1(a)所示, 运行时单击【移动】按钮, 将图片的左上角移动到窗体的中央位置, 如图 9.1(b)所示。

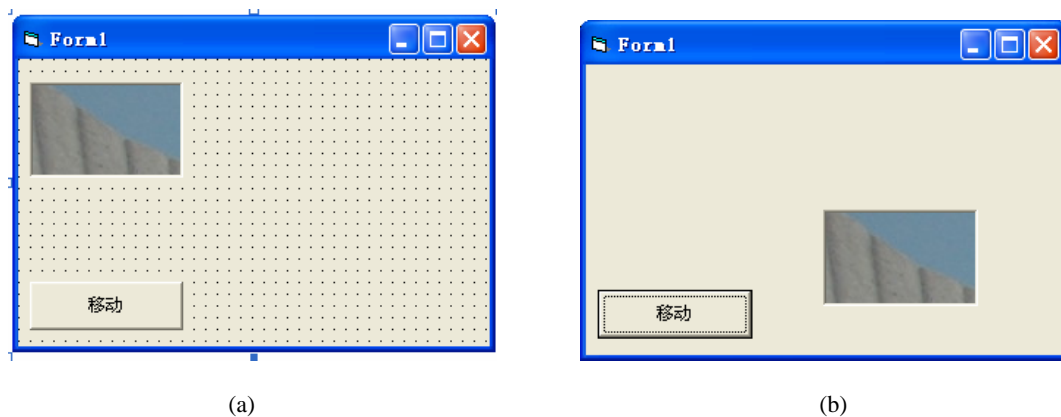


图 9.1 将一个图片的左上角移动到窗体的中央位置

【移动】按钮的 Click 事件过程如下：

```
Private Sub Command1_Click()  
    Form1.ScaleHeight = 2  
    Form1.ScaleWidth = 2  
    Picture1.Top = 1  
    Picture1.Left = 1  
End Sub
```

也可以使用 Scale 方法设置一个自定义坐标系统，其格式如下：

```
[object.] Scale (x1, y1)- (x2, y2)
```

其中：

object 为窗体或图片框对象名。

- x1, y1 为对象绘图区左上角在新的自定义坐标系下的坐标。
- x2, y2 为对象绘图区右下角在新的自定义坐标系下的坐标。
- 对 Scale 方法的调用等价于对前面提到的 4 个属性的设置，关系如下：

```
ScaleLeft = x1  
ScaleTop = y1  
ScaleWidth = x2-x1  
ScaleHeight = y2-y1
```

3. 当前坐标

当在容器中绘制图形或输出结果时，经常要将它们定位在某一希望的位置，这就必须获得某一点的坐标，即当前坐标。Visual Basic 使用 CurrentX 和 CurrentY 属性设置或返回当前坐标的水平坐标和垂直坐标。

9.1.2 颜色

计算机领域中一般采用 RGB 颜色模型，认为任何颜色都是由红(R)、绿(G)、蓝(B)3 种颜色按不同比例混合的结果。因此要设定一种颜色，只要指定其红、绿、蓝分量的大小即可。Visual Basic 中颜色的表示就是基于这个概念。要得到一种颜色有以下几种方法。

1. 使用 RGB 函数

可以使用 Visual Basic 的内部函数 RGB 返回一个颜色值，此函数有 3 个参数，取值范围都是 0~255，分别表示所要颜色中红(R)、绿(G)、蓝(B)分量的大小。如：RGB(0, 0, 0)返回黑色，RGB(255, 0, 0)返回红色，BGB(0, 0, 255)返回蓝色。

2. 使用长整型数

其实，RGB 函数返回的只是一个长整型数。在 Visual Basic 中颜色就是由长整型数表示的，所以可以直接用长整型数来指定一个颜色。

Visual Basic 中表示一个颜色的长整型数中的 4 个字节中，从高位到低位，第 1 个字节的所有位都为 0，第 2 个字节表示的是蓝色(B)分量的大小，第 3 个字节表示的是绿色(G)分量的大小，第 4 个字节表示的是红色(R)分量的大小。每个分量值的十六进制形式都是 &H00~&HFF，十进制为 0~255。

用十六进制的长整型常量表示一个颜色值是很直观的，每个颜色分量占两个十六进制位，其表示格式为：

&H00BBGGRR

例如，下面是一些表示颜色的长整型数。

&H00000000(黑色) &H0FFFFFFF(白色) &H00FF0000(浅蓝)

&H00800000(深蓝) &H0000FFFF(浅黄) &H00008080(深黄)

在源程序中输入长整数时，编辑器会自动删掉前面不必要的 0。

3. 使用系统颜色

如果一个表示颜色的长整数最高位为 1，即它的的第一个字节的值为 &H80 时，则不表示一个具体的 RGB 颜色值，而是一个系统颜色，系统颜色是由用户在 Windows 控制面板的“显示器”属性中设定的各界面元素(如菜单、滚动条、桌面等)的颜色。同一个系统颜色在不同计算机上的具体设置可能不同。系统颜色有 25 个(&H80000000~&H80000018)，它们的具体意义如表 9-2 所示。

表 9-2 系统颜色常量

符号常量	值	描 述
vbScrollBars	&H80000000	滚动条颜色
vbDesktop	&H80000001	桌面颜色
vbActiveTitleBar	&H80000002	活动窗口的标题栏颜色
vbInactiveTitleBar	&H80000003	非活动窗口的标题栏颜色
vbMenuBar	&H80000004	菜单背景颜色
vbWindowBackground	&H80000005	窗口背景颜色
vbWindowFrame	&H80000006	窗口框架颜色
vbMenuText	&H80000007	菜单上文本的颜色
vbWindowText	&H80000008	窗口内文本的颜色

续表

符号常量	值	描 述
vbTitleBarText	&H80000009	标题、调整框和滚动箭头的文本颜色
vbActiveBorder	&H8000000A	活动窗口边框颜色
vbInactiveBorder	&H8000000B	非活动窗口边框颜色
vbApplicationWorkspace	&H8000000C	多文档界面(MDI)应用程序的背景颜色
vbHighlight	&H8000000D	控件内选中项的背景色
vbHighlightText	&H8000000E	控件内选中项的文本颜色
vbButtonFace	&H8000000F	命令按钮表面阴影颜色
vbButtonShadow	&H80000010	命令按钮边缘阴影颜色
vbGrayText	&H80000011	无效文本颜色
vbButtonText	&H80000012	按钮文本颜色
vbInactiveCaptionText	&H80000013	非活动标题内文本颜色
vb3Dhighlight	&H80000014	三维显示元素的高亮颜色
vb3DDKShadow	&H80000015	三维显示元素的最暗阴影颜色
vb3Dlight	&H80000016	低于 vb3Dhighlight 的三维次高亮颜色
vbInfoText	&H80000017	提示窗内文字的颜色
vbInfoBackground	&H80000018	提示窗内背景的颜色

4. 使用颜色常量

Visual Basic 为一些常用颜色定义了内部常量，颜色常量的特点是直观、易于记忆。Visual Basic 中的颜色常量如表 9-3 所示。

表 9-3 Visual Basic 中的常用颜色常量

常 量	值	颜 色	常 量	值	颜 色
vbBlack	&H0	黑色	vbRed	&HFF	红色
vbGreen	&HFF00	绿色	vbYellow	&HFFFF	黄色
vbBlue	&HFF0000	蓝色	vbMagenta	&HFF00FF	洋红
vbCyan	&HFFFF00	青色	vbWhite	&HFFFFFF	白色

5. 使用 QBColor 函数

Visual Basic 保留了 Quick Basic 的 QBColor 函数。该函数用一个整数值对应 RGB 的常用颜色值。其格式如下：

QBColor(颜色值)

其中，“颜色值”的取值范围为 0~15，共可表示 16 种颜色。函数根据参数“颜色值”返回一个表示颜色的长整型数。不同的参数值与返回值之间的对应关系如表 9-4 所示。

表 9-4 QBColor 函数的取值

参 数 值	颜 色	参 数 值	颜 色	参 数 值	颜 色	参 数 值	颜 色
0	黑色	4	红色	8	灰色	12	亮红色
1	蓝色	5	洋红色	9	亮蓝色	13	亮洋红色
2	绿色	6	黄色	10	亮绿色	14	亮黄色
3	青色	7	白色	11	亮青色	15	亮白色

9.2 图 形 控 件

图形控件用于在对象(窗体、图片框)中绘制特定形状的图形,如圆、矩形、直线等。图形控件的属性既可以在设计阶段设置,也可以在运行阶段由程序动态地改变。

9.2.1 Shape 控件

Shape 控件用于在窗体或图片框中绘制常见的几何图形。通过设置 Shape 控件的 Shape 属性可以画出多种图形。Shape 属性具有表 9-5 所示的几种设置值。

表 9-5 Shape 控件的 Shape 属性值

设 置 值	常 量	形 状	设 置 值	常 量	形 状
0	vbShapeRectangle	默认值, 矩形	3	vbShapeCircle	圆形
1	vbShapeSquare	正方形	4	vbShapeRoundedRectangle	圆角矩形
2	vbShapeOval	椭圆形	5	vbShapeRoundedSquare	圆角正方形

【例 9.2】 在窗体中使用 Shape 控件, 画出不同形状的图形。

窗体中设置 1 个 Shape 控件 Shape1 和一个单选按钮 Option1, 将 Option1 的 Index 属性设置为 0, 创建一个只有一个元素的控件数组。程序运行后如图 9.2 所示, 每次单击单选按钮, 能看到对应的形状。

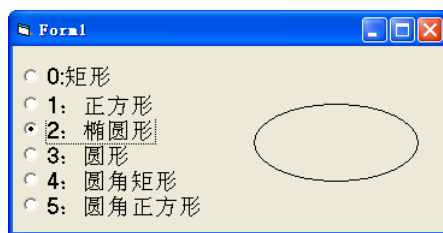


图 9.2 用 Shape 控件画出不同形状的数字

程序代码如下：

```
Private Sub Form_Load()  
    Dim a(5), i  
    a(0) = "矩形"  
    a(1) = "正方形"  
    a(2) = "椭圆形"  
    a(3) = "圆形"  
    a(4) = "圆角矩形"  
    a(5) = "圆角正方形"  
    Option1(0).Caption = 0 & ":" & a(0)  
    For i = 1 To 5                                '创建 Option1 的另外 5 个元素  
        Load Option1(i)  
        '为新的单选按钮设置位置  
        Option1(i).Top = Option1(i - 1).Top + Option1(0).Height + 30  
        '设置单选按钮的标题  
        Option1(i).Caption = i & ":" & a(i)  
        Option1(i).Visible = True  
    Next i  
End Sub  
Private Sub Option1_Click(Index As Integer)  
    Shape1.Shape = Index  
End Sub
```

9.2.2 Line 控件

Line 控件为用户提供了在容器对象中画直线的方法。简单地改变 Line 控件的 BorderStyle 属性即可画出多种线型的直线。表 9-6 列出了 BorderStyle 属性的设置值。

表 9-6 BorderStyle 属性取值

设置值	常 量	形 状
0	vbTransparent	透明，忽略 BorderWidth 属性
1	vbBSSolid	(默认值)实线，边框处于形状边缘的中心
2	vbBDDash	虚线，当 BorderWidth 属性为 1 时有效
3	vbBSDot	点线，当 BorderWidth 属性为 1 时有效
4	bBSDashDot	点划线，当 BorderWidth 属性为 1 时有效
5	vbBSDashDotDot	双点划线，当 BorderWidth 属性为 1 时有效
6	vbBSInsideSolid	内收实线，边框的外边界就是形状的外边缘

【例 9.3】 设计一个简单的秒表。

在窗体上放一个图片框 Picture1，在 Picture1 中用 Shape 控件画一个圆，再画 4 个显示数字的标签，用 Line 控件画一直线 Line1 作为秒针。再在窗体中放置一个定时器控件 Timer1 和两个命令按钮。

运行时，单击【开始】按钮(Command1)后秒针开始旋转，单击【停止】按钮(Command2)后秒针停止旋转，如图 9.3 所示。

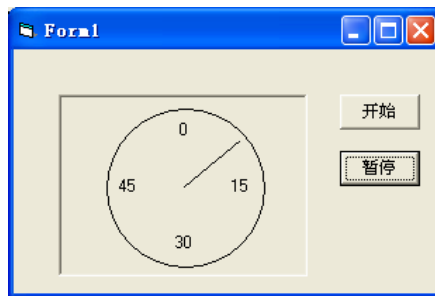


图 9.3 用 Line 控件实现的秒针

程序代码如下：

```
Dim arph '定义秒针旋转角度
Private Sub Command1_Click()
    Timer1.Enabled = True
End Sub
Private Sub Command2_Click()
    Timer1.Enabled = False
End Sub
Private Sub Form_Load()
    Timer1.Enabled = False
    Timer1.Interval = 1000
    Picture1.Scale (-1, 1)-(1, -1) '自定义图片框的坐标系
    Line1.X1 = 0 '设置秒针两端点的坐标，使其指向刻度 0
    Line1.Y1 = 0
    Line1.X2 = 0
    Line1.Y2 = 0.7
    arph = 0 '初始旋转角度为 0
End Sub
Private Sub Timer1_Timer()
    arph = arph + 3.14159 / 30
    Line1.Y2 = 0.7 * Cos(arph)
    Line1.X2 = 0.7 * Sin(arph)
End Sub
```

9.3 绘图方法

9.3.1 画点方法

画点方法(Pset)可以在窗体或图片框的指定位置上使用指定颜色画一个点。PSet 方法的语法如下：

```
[object.]Pset [step] (x, y) [,color]
```

其中，Object 为窗体或图片框的对象名。

[Step](x, y) 指定画点位置的坐标。如没有指定 Step 关键字，则(x, y)指的是绝对坐标

(相对于窗体或图片框绘图区的左上角), 如果指定 Step 关键字, 则(x, y)表示的是相对于 (CurrentX, CurrentY) 点的相对坐标。注意, 把坐标参数(x, y)括起来的小括号是不可少的。

参数 color 用来指定点的颜色, 它可以是长整型数、常量或颜色函数。如果没有指定 color, Pset 将点的颜色设置为前景色(ForeColor)。

PSet 方法执行完后, 对象的 CurrentX、CurrentY 属性值会被自动设置为画点位置的绝对坐标。

【例 9.4】 单击窗体时, 用 Pset 方法在窗体上绘制一条正弦曲线。

为了便于确定正弦曲线中每一点的坐标, 使用 Scale 方法定义窗体的水平坐标从左到右为 0 到 360, 垂直坐标从下到上为 -1~1, 即 Scale(0, 1)-(360, -1)。运行时单击窗体, 显示结果如图 9.4 所示。

窗体的 Click 事件过程如下:

```
Private Sub Form_Click()  
    Scale (0, 1)-(360, -1)  
    DrawWidth = 3           ' 设置所画点的大小  
    For x = 0 To 360  
        y = Sin(x * 3.14159 / 180)  
        PSet (x, y), vbRed  
    Next x  
End Sub
```

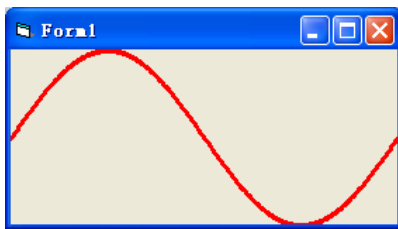


图 9.4 在窗体中绘制的一条正弦曲线

【例 9.5】 用 PSet 方法绘制艺术图案。

在程序中用 ScaleMode 将坐标系的刻度单位设置为像素, 在窗体的(0, 0)~(300, 200)区域内通过 Pset 画点来绘制一幅图案, 点的颜色与点的位置相关, 通过一个椭圆运算确定。运行时单击窗体, 显示结果如图 9.5 所示。

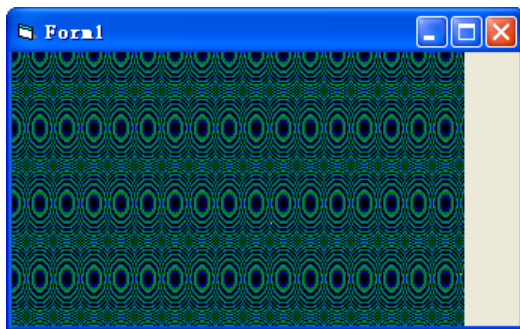


图 9.5 用 PSet 方法绘制艺术图案

窗体的 Click 事件过程如下:

```
Private Sub Form_Click()  
    Dim PColor As Integer  
    ScaleMode = 3  
    For x = 0 To 300  
        For y = 0 To 200  
            PColor = (x * x / 9 + y * y / 25) Mod 4  
            PSet (x, y), QBColor(PColor)  
        Next y  
    Next x  
End Sub
```

9.3.2 画直线、矩形方法(Line)

Line 方法可以在窗体或图片框上绘制一条直线段或一个矩形, 其语法为:

```
[object.]Line [[step] (x1, y1)]- [Step] (x2, y2) [,color] [,B[F]]
```

其中:

- object 为窗体或图片框对象名。
- 参数[step](x1, y1)指定起始坐标, [step](x2, y2)指定终止坐标。如果有参数“B”, 则绘制以给定两点为对角的矩形, 否则画出以给定两点为端点的直线段。
- 参数 color 指定直线或矩形边框的颜色。如果有参数“F”, 则用边框颜色填充矩形。无参数“B”时, 不能使用参数“F”。如果(x2, y2)参数前有 step 关键字, 表示是以起始点绝对坐标为基准的相对坐标。

执行完此方法后, 对象的 CurrentX、CurrentY 属性值即为终点的坐标。

如果在调用 Line 方法时, 省略了参数“[step] (x1, y1)”, 则会把 CurrentX、CurrentY 属性的值作为起始点的坐标, 相当于“step (0,0)”。

另外, 调用一次带参数“B”的 Line 方法只能画出 4 条边是水平或垂直的矩形。如果要绘制任意方向的矩形, 可以调用 4 次 Line 方法画出矩形的 4 条边。

【例 9.6】 使用 Line 方法绘制几何图案。

运行时单击窗体, 显示结果如图 9.6 所示。

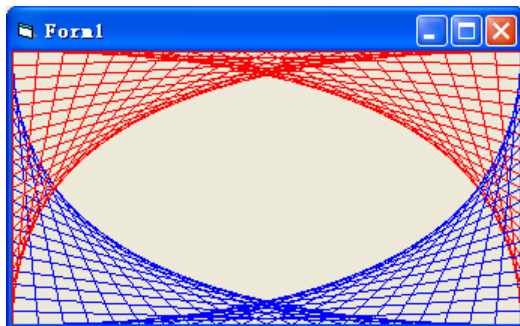


图 9.6 使用 Line 方法绘制的几何图案

窗体的 Click 事件过程如下:

```
Private Sub Form_Click()
    Scale (1, 1)-(400, 180)
    xmax = 400
    ymax = 180
    xmin = 1
    ymin = 1
    s = 20
    xs = (xmax - xmin) / s
    ys = (ymax - ymin) / s
    For i = 0 To s
        Line (xmin + xs * i, ymax)-(xmax, ymax - ys * i), vbBlue
        Line (xmax - xs * i, ymin)-(xmax, ymax - ys * i), vbRed
        Line (xmax - xs * i, ymin)-(xmin, ymin + ys * i), vbRed
        Line (xmin, ymin + ys * i)-(xmin + xs * i, ymax), vbBlue
    Next i
End Sub
```

9.3.3 画圆方法(Circle)

Circle 方法可以在窗体或图片框上绘制圆形、椭圆或弧, 其语法为:

```
[object.]Circle [Step] (x,y),radius [,color][,start] [,end] [,aspect]
```

其中:

- object 为窗体或图片框对象名。
- 参数[Step](x,y) 指定圆心或椭圆中心的坐标,radius 指定圆的半径或椭圆的长半轴,color 为线条颜色。
- start 与 end 指定弧的起止角度(单位是弧度), 如果被省略, 则绘制出完整的圆或椭圆。
- Aspect 指定圆度(垂直半轴与水平半轴长度之比), 当它为 1 或省略时, 绘出的是一个圆, 取其他值时为椭圆。

当 Circle 方法的 start 或 end 参数为负值时, 绘制的是与相应的正值相向的一段弧, 只不过多画出从端点到中心的连线。

此方法执行后, 对象的 CurrentX、CurrentY 属性的值会等于圆心或椭圆中心的坐标。

【例 9.7】 使用 Circle 方法绘制艺术图案。

将一个作为圆心轨迹的圆(该圆的圆心为(x0,y0), 半径为 r)等分为 30 份, 以这 30 个等分点为圆心画 30 个圆, 这些圆的圆心坐标为(x0+r*cos(i),y0 - r*sin(i)), 其中“i”为从水平轴右侧逆时针转动的角度(以弧度为单位)。

运行时单击窗体, 画出如图 9.7 所示的具有艺术效果的图案。

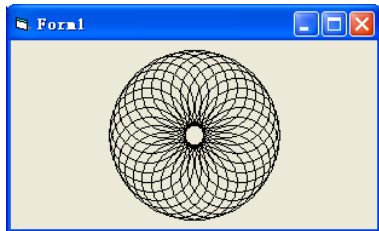


图 9.7 用 Circle 方法绘制的艺术图案

窗体的 Click 事件过程如下：

```
Private Sub Form_Click()
    r = ScaleHeight / 4
    x0 = ScaleWidth / 2
    y0 = ScaleHeight / 2
    pi = 3.14159
    For i = 0 To pi * 2 Step pi / 15
        x = x0 + r * cos(i)
        y = y0 - r * sin(i)
        Circle (x, y), r * 0.8
    Next i
End Sub
```

【例 9.8】 模拟地球绕太阳旋转。

在窗体中放置 2 个合适大小的形状控件 Shape1、Shape2。将 Shape1 的 shape 属性设置为圆形，显示填充色为红色，用它代表太阳；将 Shape2 的 shape 属性设置为圆形，显示填充色为蓝色，用它代表地球。用定时器控件 Timer1 控制地球运行速度，将 Timer1 的时间间隔设置为 100(0.1 秒)。程序运行情况如图 9.8 所示。

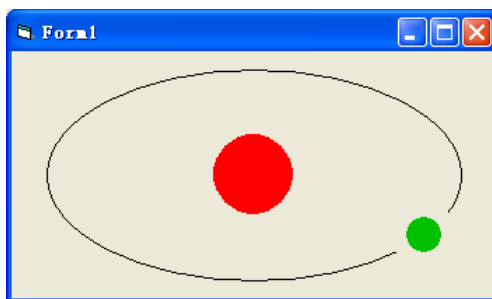


图 9.8 地球绕太阳旋转的模拟图

地球运动的椭圆方程为：

$$\begin{aligned} x &= x_0 + r_x \cdot \cos(\alpha) \\ y &= y_0 + r_y \cdot \sin(\alpha) \end{aligned}$$

其中 x_0 、 y_0 为椭圆圆心坐标， r_x 为水平半径， r_y 为垂直半径， α 为圆心角。

程序代码如下：

```
Dim rx As Single, ry As Single
Dim alfa As Single
Private Sub Form_Load()
    '将表示太阳的 Shape1 放在窗体的正中央
    Shape1.Left = Form1.ScaleWidth / 2 - Shape1.Width / 2
    Shape1.Top = Form1.ScaleHeight / 2 - Shape1.Height / 2
    '计算椭圆轨道的水平半径 rx 和垂直半径 ry
    rx = Form1.ScaleWidth / 2 - Shape2.Width / 2
    ry = Form1.ScaleHeight / 2 - Shape2.Height / 2
    '将表示地球的 Shape2 的初始位置定位在水平轴的 0 刻度位置上
    Shape2.Left = Form1.ScaleWidth / 2 + rx - Shape2.Width / 2
```

```

    Shape2.Top = Form1.ScaleHeight / 2 - Shape2.Height / 2
End Sub
Private Sub Timer1_Timer()
    alfa = alfa + 0.05
    '绘制地球的运行轨迹
    Circle (Form1.ScaleWidth / 2, Form1.ScaleHeight / 2), rx, , , , ry / rx
    x = Form1.ScaleWidth / 2 + rx * cos(alfa)
    y = Form1.ScaleHeight / 2 + ry * sin(alfa)
    Shape2.Left = x - Shape2.Width / 2
    Shape2.Top = y - Shape2.Height / 2
End Sub

```

9.3.4 PaintPicture 方法

PaintPicture 方法用于在窗体或图片框上绘制来自于磁盘文件的图像，在绘制时可以只绘制图像的一部分，并进行缩放。其语法为：

```

object.PaintPicture picture,x1,y1 [,width1] [,height1] [,x2] [,y2]
[,width2] [,height2] [,opcode]

```

其中：

- object 为窗体或图片框的对象名。
- 参数 picture 指定要绘制的图像，可以使用 LoadPicture 函数调入磁盘上的图形文件，也可以是窗体或图片框的 Picture 属性指定的图形文件。图形文件类型包括.bmp、.ico、.wmf、.emf、.cur 和.dib 等。
- x1、y1 参数指定将图像绘制在窗体或图片框的什么位置上(坐标)。
- width1、height1 参数指定把图像绘制在窗体或图片框上区域的高度和宽度，如果是负值，可以把图像翻转显示。
- x2、y2、width2 和 height2 这 4 个参数可以指定只绘制图像的某个矩形区域，x2、y2 为区域左上角的坐标，width2、height2 为绘制区域的宽度和高度。
- opcode 参数指定绘制的方法，即绘制出的图像中每一点使用什么颜色。此颜色由被绘制图像中每一个点的颜色(源像素)、窗体或图片框上现有像素点的颜色(目标像素)和填充颜色(由 FillColor 和 FillStyle 属性决定)计算得到，此计算实质上是表示颜色的长整数进行按位逻辑运算。
- opcode 参数的默认值为 &H00CC0020，使用源像素颜色覆盖目标像素，opcode 参数的取值如表 9-7 所示。

表 9-7 opcode 参数的取值

参 数 值	常 量	描 述
&H005509	vbDstInvert	Not (目标像素)
&H00C000CA	vbMergeCopy	(源像素) And (填充)
&H00BB0226	vbMergePaint	(Not (源像素)) or (目标像素)
&H00330008	vbNotSrcCopy	Not (源像素)
&H001100A6	vbNotSrcErase	Not ((源像素) or (目标像素))

续表

参 数 值	常 量	描 述
&H00F00021	vbPatCopy	只绘制填充
&H005A0049	vbPatInvert	(填充) Xor (目标像素)
&H00FB0A09	vbPatPaint	(Not (源像素) Or (填充)) Or (目标像素)
&H008800C6	vbSrcAnd	(源像素) And (目标像素)
&H00CC0020	vbSrcCopy	使用源像素颜色
&H00440328	vbSrcErase	(源像素) And (Not (目标像素))
&H00660046	vbSrcInvert	(源像素) Xor (目标像素)
&H00EE0086	vbSrcPaint	(源像素) Or (目标像素)

如果省略所有的可选参数,则 PaintPicture 方法会把整个图像按原来的颜色和大小绘制到指定的位置上。

【例 9.9】 使用 PaintPicture 方法对图像进行变形、裁剪和翻转处理

在窗体中放置 4 个图片框(Picture1、Picture2、Picture3 和 Picture4),在图片框 Picture1 中显示源图形文件,单击窗体后,分别在图片框 Picture2~Picture4 中显示该原始图片经过变形、裁剪和翻转后的效果。程序运行情况如图 9.9 所示。



图 9.9 对图形的变形、裁剪和翻转处理效果

程序代码如下:

```
Private Sub Form_Click()  
    '把源图形的宽度拉伸 1 倍,绘制在图片框 Picture2 中  
    Picture2.PaintPicture Picture1.Picture, 0, 0, Picture1.Width * 2,  
    Picture1.Height  
    '裁剪源图形的上半部分,绘制在图片框 Picture3 中  
    Picture3.PaintPicture Picture1.Picture, 0, 0, , , 0, 0, Picture1.Width,  
    Picture1.Height / 2  
    '将源图形旋转 180° 后,绘制在图片框 Picture4 中  
    Picture4.PaintPicture Picture1.Picture, Picture1.Width + 1,  
    Picture1.Height + 1, -Picture1.Width, -Picture1.Height  
    '将源图形垂直翻转后放大,绘制在图片框 Picture4 中  
    Picture4.PaintPicture Picture1.Picture, Picture1.Width + 100,  
    Picture1.Height * 1.5, Picture1.Width * 2, -Picture1.Height * 1.5  
End Sub
```

使用 PaintPicture 方法绘制在窗体或图片框上的图像，与窗体和图片框的 Picture 属性设定的背景图像有着根本的区别。背景图像不会被擦除(除非改变了 Picture 属性的值)，显示的位置不能改变；绘制的图像可以指定位置，可以被擦除。

使用窗体和图片框的绘图方法绘制的图形和图像与 Shape 控件、Image 控件、Line 控件等也有着本质的不同。绘制的图形图像只是临时显示在屏幕上，基本上不占用内存，而各类控件作为对象具有属性、事件和方法，除了在屏幕上显示相应的图形图像之外，还会占用一定的内存。

9.4 与绘图有关的常用属性和事件

9.4.1 清除图形方法

Cls 方法用来清除窗体或图片框上由 Print、PSet、Line、Circle 等方法输出的文字、图形和图像。清除之后 CurrentX 和 CurrentY 属性值都被设为 0。其格式为：

```
[object.] Cls
```

其中，object 是窗体或图片框的对象名。Cls 方法没有参数。

Cls 方法不会清除窗体和图片框上由 Picture 属性设置的背景图像，更不会清除窗体或图片框上的控件对象。

9.4.2 线宽属性和线型属性

DrawWidth 属性设置绘图方法绘制的图形的线条样式，其取值范围从 1~32767，以像素为单位。默认值为 1，即一个像素宽。

如果 DrawWidth 属性值大于 1，画出的图形是实线；如果 DrawWidth 属性值等于 1，可以画出由 DrawStyle 定义的各种线型。

DrawStyle 属性设置绘图方法绘制的图形的线条宽度。此属性的取值为 0~6，具体意义如表 9-8 所示。

表 9-8 DrawStyle 属性值

设置值	常量	线型	设置值	常量	线型
0	VbSolid	实线，默认值	4	vbDashDotDot	双点划线
1	VbDash	虚线	5	vbInvisible	无线
2	VbDot	点线	6	vbInsideSolid	内收实线
3	vbDashDot	点划线			

9.4.3 填充颜色属性和填充样式属性

FillColor 属性设置由 Circle 和 Line 方法生成的圆、矩形等封闭图形的内部填充颜色。

FillStyle 属性设置 Shape 控件所画图形的填充样式，也可以设置由 Circle 和 Line 图形

方法生成的封闭图形的填充样式。此属性的取值为 0~7，具体意义如表 9-9 所示。

FillStyle 属性的默认值为 1(透明)，此时，FillColor 属性的值被忽略。

表 9-9 FillStyle 属性值

设置值	常量	填充样式	设置值	常量	填充样式
0	vbFSSolid	实心	4	vbUpwardDiagonal	上斜对角线
1	vbFSTransparent	透明，默认值	5	vbDownwardDiagonal	下斜对角线
2	vbHorizontalLine	水平直线	6	vbCross	十字线
3	vbVerticalLine	垂直直线	7	vbDiagonalCross	交叉对角线

9.4.4 自动重画属性

应用程序在运行时其窗体经常被移动或被其他窗体覆盖，要想保持窗体中的内容(图形等)不丢失，就要在窗体移动、改变大小或覆盖它的窗体移开后，重新显示(绘制)窗体中的内容。一般来说，Windows 管理和控制窗口、控件的重新显示，而窗体和图片框内图形的重新显示必须由用户的应用程序来控制。AutoRedraw 属性就提供了重新显示窗体和图片框内图形的功能。

当 AutoRedraw 属性设置为 False(默认值)时，对象中的图形不具有持久性，即当窗体或图片框全部或部分被其他窗体遮盖再重新显示时，绘图方法产生的图形不会被自动重画，对象上的图形将丢失。

当 AutoRedraw 属性设置为 True 时，表示对象的自动重画功能有效，使用绘图方法绘制的图形会被保存在内存中，当窗体或图片框的全部或部分被其他窗体遮盖又显示出来后，图形会自动重画。

【例 9.10】 AutoRedraw 属性的使用示例。

在窗体中的右方放置两个命令按钮 Command1、Command2 和一个 PictureBox 控件 Picture1。程序运行时，单击窗体，当 AutoRedraw 属性为 True 时在窗体上画一个绿色实心圆，当 AutoRedraw 属性为 False 时在窗体上画一个红色实心圆。单击【覆盖】按钮(Command1)后，将图片框 Picture1 移去，同时覆盖两种颜色的实心圆；单击【移开】按钮(Command2)，使图片框 Picture1 移开，此时会发现绿色实心圆将完整地再现，而红色实心圆被图片框遮住的部分没有被重画。运行情况如图 9.10 所示。

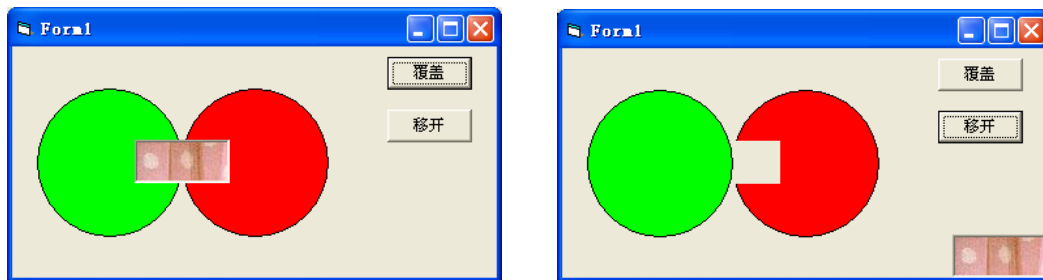


图 9.10 AutoRedraw 属性的使用示例

程序代码如下:

```
Private Sub Form_Click()  
    ScaleWidth = 100  
    AutoRedraw = True  
    FillColor = vbGreen  
    FillStyle = 0  
    Circle (20, ScaleHeight / 2), 15  
    AutoRedraw = False  
    FillColor = vbRed  
    Circle (50, ScaleHeight / 2), 15  
End Sub  
Private Sub Command1_Click()  
    Picture1.Top = ScaleHeight / 2 - Picture1.Height / 2  
    Picture1.Left = 35 - Picture1.Width / 2  
End Sub  
Private Sub Command2_Click()  
    Picture1.Top = ScaleHeight - Picture1.Height  
    Picture1.Left = ScaleWidth - Picture1.Width  
End Sub
```

9.4.5 Paint 事件

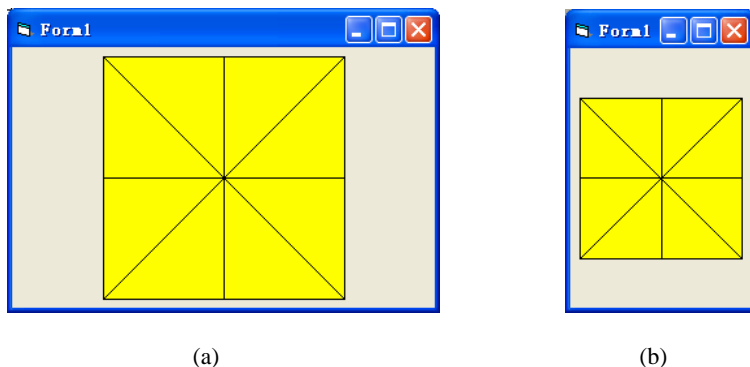
在应用程序运行时, 当一个对象(如窗体或图片框)被移动或改变大小之后, 或当一个覆盖该对象的窗体被移开之后, 如果要保持该对象上所画图形的完整性(重现原来的图形), 可以选择触发 **Paint** 事件来完成图形的重画工作。

窗体与图片框支持 **Paint** 事件。这样当窗体、图片框被遮盖后又显示出来或被缩放时, **Visual Basic** 向窗体或图片框发送 **Paint** 事件, 允许程序(写在其 **Paint** 事件过程中)进行重新绘制。

使用 **Refresh** 方法时, 也将触发 **Paint** 事件。因此, 在改变窗体大小后, 应在 **Resize** 事件过程中调用 **Refresh** 方法, 强制对象通过 **Paint** 事件重画图形。

需要注意的是, 当窗体或图片框的 **AutoRedraw** 属性设为 **True** 时, 不引发 **Paint** 事件, 即不执行 **Paint** 事件过程中的代码。只有当 **AutoRedraw** 属性值为 **False** 时, 才执行 **Paint** 事件过程。

【例 9.11】 在窗体中心画一个黄色米字格, 当窗体的大小改变时, 米字格也随着自动调整, 如图 9.11 所示。



(a)

(b)

图 9.11 例 9.11 的运行界面

程序代码如下：

```
Private Sub Form_Paint()  
    '计算窗体中心坐标  
    CenterX = ScaleLeft + ScaleWidth / 2  
    CenterY = ScaleTop + ScaleHeight / 2  
    '计算米字格的边长  
    If ScaleWidth < ScaleHeight Then  
        HalfWidth = ScaleWidth / 2 - 100  
    Else  
        HalfWidth = ScaleHeight / 2 - 100  
    End If  
    '设置米字格正方形的填充颜色和样式  
    FillColor = vbYellow  
    FillStyle = 0  
    '画米字格正方形  
    Line (CenterX - HalfWidth, CenterY - HalfWidth)-(CenterX + HalfWidth,  
CenterY + HalfWidth), , B  
    '画两条对角线  
    Line (CenterX - HalfWidth, CenterY - HalfWidth)-(CenterX + HalfWidth,  
CenterY + HalfWidth)  
    Line (CenterX - HalfWidth, CenterY + HalfWidth)-(CenterX + HalfWidth,  
CenterY - HalfWidth)  
    '画十字形状  
    Line (CenterX, CenterY - HalfWidth)-(CenterX, CenterY + HalfWidth)  
    Line (CenterX - HalfWidth, CenterY)-(CenterX + HalfWidth, CenterY)  
End Sub  
Private Sub Form_Resize()  
    Refresh  
End Sub
```

9.5 小 结

Visual Basic 为程序设计者提供了非常丰富的绘图功能。设计程序时，不仅可以使
用 Visual Basic 提供的图形控件来画图，还可以调用图形方法绘制丰富多彩的艺术图形。

坐标系统是图形处理的基本概念，它包括原点位置、坐标单位以及坐标轴的方向等几
方面内容。在窗体和图片框中绘图时，Visual Basic 提供了 7 种标准坐标系统供选用。此外，
还可以使用自定义坐标系统。

在 Visual Basic 中，要得到一种颜色有 5 种方法：使用 RGB 函数、使用长整型数、使
用系统颜色、使用颜色常量、使用 QBColor 函数。

图形控件包括 Line 控件和 Shape 控件，用于在对象(窗体、图片框)中绘制特定形状
的图形，如直线、圆、矩形等。图形控件的属性既可以在设计阶段设置，也可以在运行阶段
由程序动态地改变。

Visual Basic 中提供了一些方法，用于在窗体或图片框中直接绘制图形。其中，PSet
方法可以在窗体或图片框的指定位置上使用指定颜色画一个点。PSet 方法的语法为：

```
[object.]Pset [step] (x, y) [,color]
```

Line 方法可以在窗体或图片框上绘制一条直线段或一个矩形，其语法为：

```
[object.]Line [[step] (x1, y1)]- [Step] (x2, y2) [,color] [,B[F]]
```

Circle 方法可以在窗体或图片框上绘制圆形、椭圆或弧，其语法为：

```
[object.]Circle [Step] (x,y),radius [,color][,start] [,end] [,aspect]
```

Visual Basic 中与绘图有关的常用属性有线宽属性 DrawWidth、线型属性 DrawStyle、填充颜色属性 FillColor、填充样式属性 FillStyle 和自动重画属性 AutoReDraw 等。

9.6 习 题

一、选择题

1. 坐标系中默认的刻度单位是缇，可以根据需要，用()属性来改变其刻度单位。

A) DrawStyle 属性

B) ScaleHeight 属性

C) ScaleWidth 属性

D) ScaleMode 属性

2. 执行下列程序段后，窗体 Form1 右下角的坐标为()。

```
Form1.ScaleTop=1
Form1.ScaleLeft=1
Form1.ScaleHeight=-2
Form1.ScaleWidth=2
```

A) (1,1)

B) (1.2)

C) $(-2, 2)$

D) $(3, -1)$

3. 如果用长整数&H00FF0000& 来表示颜色, 则此颜色为()。

A) 红色

B) 黄色

C) 蓝色

D) 绿色

4. 通过设置 Line 控件的()属性,可以绘制虚线、点线、点划线等各种样式的图形。

A) Line

B) Style

C) FillStyle

D) BorderStyle

5. 当使用 **Line** 方法画直线后，当前坐标在()。

A) $(0, 0)$

B) 直线起点

C) 直线终点

D) 容器的中心

6. 执行指令“Line(1200, 1200)–Step(1000, 500)”后, CurrentX=()。

A) 2200

B) 1200

C) 1000

D) 1700

二、阅读下列程序，写出运行结果

- ### 1. Private Sub Form_Click()

```
For i = 1 To 100
    xpos = ScaleWidth / 2
    ypos = ScaleHeight / 2
    r = Int(Rnd * 256)
    g = Int(Rnd * 256)
    b = Int(Rnd * 256)
```

```

    If ScaleWidth < ScaleHeight Then
        radius = Int(ScaleWidth / 2 * Rnd)
    Else
        radius = Int(ScaleHeight / 2 * Rnd)
    End If
    Circle (xpos, ypos), radius, RGB(r, g, b)
Next i
End Sub

```

2. Private Sub Form_Click()

```

    Scale (0, 1)-(360, -1)
    DrawWidth = 2
    pi = 3.14159
    For x = 0 To 360
        y = cos(x * pi / 180)
        PSet (x, y), vbRed
        y = sin(x * pi / 180)
        PSet (x, y), vbBlue
    Next x
End Sub

```

三、编程题

1. 用 Line 方法画图，如图 9.12 所示。要求运行时图形将随着窗体尺寸的改变而自动调整。

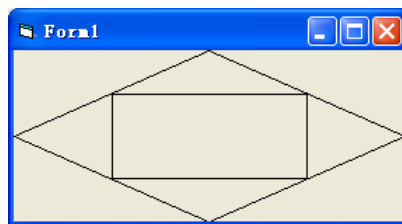


图 9.12 编程题 1 的效果图

2. 输入某班学生某门课程成绩，统计成绩在 90~100 分、80~89 分、70~79 分、60~69 分和不及格等各个分数段的人数，且以饼形图的形式显示各个分数段人数占总人数的比例。

第 10 章 文 件

教学提示：大多数的应用程序都需要进行数据的存储及读取操作，而要在外存上存放数据，必须使用文件。对于文件的操作主要是进行数据的读取、保存及修改等操作。

教学目标：掌握不同访问类型文件的存取操作，尤其是顺序文件及随机文件的使用。熟练掌握文件系统控件的使用。掌握 FileSystemObject(FSO)对象模型的基本特点及使用方法。

10.1 文件的基本概念

计算机中的“文件”一般指具有符号名的数据信息的集合。该符号名是用以标识文件的，称为文件名。文件在计算机中的地位是非常重要的，在日常的实用程序中，经常要与文件打交道，各种应用信息如通信录、职工的工资表、人事档案表、设备表等给予命名后均可作为一个文件。系统中的文件一般分很多类型，不同类型文件的操作方法也有所不同。

10.1.1 文件的类型

为了便于管理和控制文件，通常将文件分成若干类型。按照文件的存取方式及其组成结构可以将文件分为两种类型：顺序文件和随机文件；按照文件的特征属性可以将文件分为系统文件、隐藏文件、只读文件、普通文件和存档文件；按照文件的数据组织形式可以将文件分为 ASCII 文件和二进制文件，其中 ASCII 码文件又称为文本文件；按文件中的数据可将文件分为程序文件和数据文件。

在使用文件时，应根据文件包括什么类型的数据，确定合适的文件访问类型。在 Visual Basic 中，有 3 种文件访问类型，即顺序型文件、随机型文件和二进制型文件。

- **顺序型文件：**是普通的文本文件。文件中每个字节存放的是该字节所表示的字符的 ASCII 码值。
- **随机型文件：**简称随机文件，是由固定长度的记录集合组成。数据一般以二进制方式存储。
- **二进制型文件：**简称二进制文件，是将数据按照内存中的存储格式来表示，允许程序按所需的任何方式组织和访问数据。除了没有数据类型或记录长度的含义以外，它与随机文件很相似。然而，为了能够正确地检索，必须精确地知道数据是如何写到文件中的。

10.1.2 文件访问函数和语句

表 10-1 列出了可用于 3 种文件访问类型的各种文件访问的语句和函数。

表 10-1 可用于 3 种文件访问类型的各种文件访问语句和函数

语句与函数	顺 序 型	随 机 型	二进制型
Open	√	√	√
Close	√	√	√
Input()	√		√
Input#	√		
Line Input#	√		
Print #	√		
Write #	√		
Get		√	√
Put		√	√
Type...End Type		√	

10.2 顺 序 文 件

顺序文件是普通的文本文件，是一种结构比较简单的格式。它的存储方式是顺序存储，即数据是一个接一个地顺序写入文件中的。这种方式只提供了第一个数据的存储位置，其他数据的位置则无法获得，因此，要对文件内部进行修改，就必须将整个文件读到内存中进行，然后再写回文件。这种数据存取方式简单，占用磁盘空间较少，但是无法灵活而随意地存取文件数据，仅适用于不经常修改的数据的存储。顺序文件的打开与关闭、读取和写入在 Visual Basic 中均有相应的语句。

10.2.1 顺序文件的打开与关闭

要对文件进行操作，第一步就是要先打开文件。Visual Basic 提供了 Open 语句，使用该语句，用户可以用顺序、随机或二进制方式打开文件。Open 语句的语法格式如下：

```
Open FileName [For Mode] [Access access][LockType] As [#] FileNumber
[Len=RecordLenth]
```

其中，FileName 为要操作的文件名；For 子句中的 Mode 用来描述打开模式：Output、Append、Input、Random 或 Binary，其中 Output 模式打开的文件用来写入；Append 模式打开的文件用来在文件尾部追加数据；Input 模式打开的文件用来从文件中读取数据；Random 模式打开的文件用来对文件读取或写入定长记录；Binary 模式打开的文件用来读取或写入，但不要求记录定长。

Access 子句中的 access 用来指定文件的存取权限：Read(只读)、Write(只写)或 ReadWrite(可读或写)；LockType 设置文件打开后其他进程对打开的文件所能进行的操作，可设置为 Shared、Lock、LockRead、LockWrite 和 LockReadWrite 等。

FileNumber 用来标识打开文件的句柄,它必须是 1 到 511 之间的整数,打开文件后就可通过 FileNumber 来读、写文件及关闭文件;在 Len 子句中,RecordLength 用于设置以 Random 方式打开的文件中记录的长度(以字节为基本单位),默认的 RecordLength 为 128。

在对顺序文件进行操作时,可用 Input、Output 和 Append 模式打开文件;在对随机文件进行操作时,可用 Random 模式打开文件;在对二进制类型文件进行操作时,可用 Binary 模式打开文件。

在完成对文件的各种处理后,还需关闭文件以防数据丢失,这由 Close 语句完成。它的语法格式如下:

```
Close [[#]FileNumber]
```

其中,FileNumber 参数是 Open 语句打开文件时指定的句柄。当执行 Close 语句时,与 FileNumber 相对应的文件被关闭,以后可以用相同或不同的文件号重新打开此文件。

【例 10.1】 使用 Open 语句打开文件示例。

(1) 以顺序模式打开 E:\VB\VB.txt 文件,并从中读取数据

```
Open "E:\VB\VB.txt" For Input As #1
```

(2) 以只允许写操作的二进制方式打开 E:\VB\VB1.txt 文件。

```
Open "E:\VB\VB1.txt" For Binary Access Write As #2
```

(3) 以顺序方式打开 E:\VB\VB3.txt 文件进行写操作。

```
Open "E:\VB\VB3.txt" For Output As #3
```

【例 10.2】 使用 Close 语句关闭文件示例。

(1) 关闭 E:\VB\VB.txt 文件。

```
Close #1
```

(2) 关闭 E:\VB\VB1.txt 文件。

```
Close #2
```

(3) 关闭 E:\VB\VB3.txt 文件。

```
Close #3
```

10.2.2 顺序文件的读写

对顺序文件可以用相应的方法进行写入操作,掌握了正确的操作方法就可以灵活使用文件。

1. 顺序文件的读取

Visual Basic 提供了以下过程和函数对文件进行读操作:

```
Input #FileNumber,VarName[,VarName...]  
Input (Length,#FileNumber)  
Line Input #FileNumber,VarName
```

其中 FileNumber 表示用 Open 打开文件时使用的文件句柄,VarName 用来存储从文件

中读出数据的变量名。Input 函数中的 Length 参数用来指明要从文件中读出字符的长度，此函数返回的是一个字符串。

使用 Line Input 读取数据时，会一直读到回车/换行符或文件尾为止。

【例 10.3】 按行读取文件 E:\VB\VB1.txt 中的数据，并将数据显示在屏幕上。

```
Dim FileNumber As Integer
Dim Inbuf As String
Open "E:\VB\VB1.txt" For Input As FileNumber
'以顺序方式打开文件进行读操作
Do While Not EOF(FileNumber)           '用来判断文件是否结束
    Line Input #FileNumber, Inbuf       '从文件中读取一行数据存入 Inbuf
    Print Inbuf                          '在屏幕上输出 Inbuf 中的数据
Loop
Close #FileNumber                       '关闭文件
```

2. 顺序文件的写操作

对顺序文件的写操作比较方便，Visual Basic 提供了 Print #和 Write #语句来完成写操作。

Print #语句的功能是把格式化显示的数据写入顺序文件中，其语法格式如下：

```
Print # FileNumber, [OutputList] [,|;]
```

其中，FileNumber 为一有效的文件号，OutputList 表示要写入文件的表达式，可用逗号将这些表达式分隔开来。“;”和“,”决定下一个字符输出的位置，分号表示下一个字符紧随其前面一个字符打印，逗号表示下一个字符在前一个字符的下一个打印区开始打印，若不加“,”和“;”参数，Print #语句会在字符结束处添加回车/换行符。Print #语句常和 Line Input #语句配合使用。

Write #的语法格式如下：

```
Write # FileNumber, [OutputList]
```

其中，FileNumber 为一有效的文件号，OutputList 表示要写入文件的表达式，用逗号将这些表达式分隔开来。Write #语句常和 Input #语句配合使用。

在使用 Print #语句或 Write #语句对文件进行写操作时，文件必须以 Output 或 Append 模式打开。当以 Append 模式打开文件时，Print #或 Write #语句输出的内容将追加到文件尾部。

【例 10.4】 顺序文件的写操作

```
Private Sub Command1_Click()
Open "E:\VB\MyTest1.txt" For Output As #1 '以顺序方式打开文件进行写操作
Print #1, 2 * 3;                          '(;)表示下一个数据的输出紧随该数据
Print #1,4,                               '(:,)表示下一个数据在下一个输出区输出
Print #1, 3                               '数据后无符号表示下一个数据在下一行输出
Print #1, 5
Close #1
End Sub
```

执行该程序后，打开文件可看到数据的格式如下：

```
6 4          3 5
```

从该格式可看出，4 尾随在 6 后输出，而 3 在 4 的下一输出区，5 则在下一行输出。

10.3 随 机 文 件

随机文件由固定长度的记录组成，每个记录可以设计成若干个字段，各字段的类型可以不同。此类文件中的数据是以二进制形式存放在外存中的。随机文件对文件的读写顺序没有限制，可以随意读写某一条记录，它以记录号来定位记录的位置。随机文件的读写速度快，而且打开后可同时做读、写操作，不像顺序文件那样打开后只能做读或写中的一种。但是随机文件的缺点是占用空间大。

随机文件的建立分 4 个步骤：定义数据类型、打开文件、对文件进行读写和关闭文件。

10.3.1 定义数据类型和变量声明

因为随机文件是由固定长度的记录组成，所以在对随机文件进行操作时，需要先定义记录数据类型并声明变量，然后才能进行读写操作。在 Visual Basic 中，使用语句 `Type...End Type` 来定义记录类型。

【例 10.5】 设计一个通信录记录结构，包括姓名、手机号码和电话号码

```
Type ContactInfo
    Name As String*8
    Mobile As String*11
    Phone As String*12
End Type
```

有了自定义的数据类型，就可以用来声明变量，从而可以处理文件中的数据。下面是定义一个 `ContactInfo` 类型变量的语句。

```
Public Contact As ContactInfo
```

10.3.2 随机文件的打开与关闭

打开随机文件使用的语句同样是 `Open` 语句，但是打开模式必须是 `Random` 方式，同时要指明记录长度，其语法格式为：

```
Open FileName For Random As #FileNumber Len=RecordLen
```

其中，`Random` 是默认的访问类型，是可选项。表达式 `Len=RecordLen` 指定了每个记录的长度。如果 `RecordLen` 比实际长度短，则产生一个错误，如果 `RecordLen` 比实际长度长，则可写入，但是会浪费磁盘空间。因此，在文件处理前，应合理设计记录结构，确定记录长度，可以用 `Len` 函数获得记录的实际长度，如使用 `Len(ContactInfo)` 便可获得通信录记录的实际长度。下面的语句可以打开随机文件 `File.dat` 进行读写操作。

```
Open "File.dat" For Random As #1 Len=Len(ContactInfo)
```

随机文件的关闭同顺序文件一样，使用 `Close #FileNumber` 语句即可。

10.3.3 随机文件的读写操作

Visual Basic 读写随机文件的语句是 `Get #`和 `Put #`。它们的语法格式如下：

```
Get #FileNumber,[RecordNumber],VarName
Put #FileNumber,[RecordNumber],VarName
```

其中，`Get #`语句用于从文件中把 `RecordNumber` 记录号所代表的记录读到对应类型的变量 `VarName` 中；`Put #`语句则把 `VarName` 中的数据写入文件中，形成文件的第 `RecordNumber` 条记录。

【例 10.6】 已知文件 `E:\VB\MyContact.dat` 中存放着 3 条通信录信息，现要将“张三，13504312087,01062620610”这条记录存入该文件中形成第 4 条记录，并将第 2 条记录的信息显示在屏幕上。


```
' 以随机方式打开文件 E:\VB\MyContact.dat
Open "E:\VB\MyContact.dat" For Random As #1 Len=Len(ContactInfo)
' 为通信录类型的变量赋值
Contact.Name="张三"
Contact.Mobile="13504312087"
Contact.Phone="0062620610"
' 将记录写入文件的第 4 条记录处
Put #1,4,Contact
' 读取文件的第 2 条记录
Get #1,2,Contact
' 在屏幕上输出第 2 条记录
Print Contact.Name,Contact.Mobile,Contact.Phone
Close #1
```

其中，`Contact` 的记录类型见例 10-5，记录类型的定义及变量的声明放在模块中。

10.4 文件系统控件

Visual Basic 6.0 中包括 3 个文件系统控件，分别是驱动器列表框、目录列表框和文件列表框控件。利用这 3 个控件可以很方便地实现查看磁盘驱动器、目录及文件的功能，它们常常配合使用，以实现对相关文件的控制。

10.4.1 驱动器列表框控件

驱动器列表框控件(`DriveListBox`)用于设置或显示当前磁盘驱动器的名称，在工具箱中的图标是 。该控件是一个下拉列表框，单击列表框中的下拉箭头，会显示用户系统中所有有效驱动器的列表。如图 10.1 为驱动器列表框控件在运行时的情况。

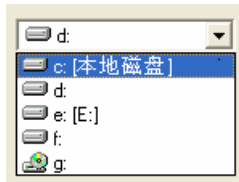


图 10.1 驱动器列表框控件运行示意图

该控件的使用非常简单，只要在窗体上将该控件放上，运行时，即可在下拉列表中显示所在系统的驱动器。一般驱动器列表框控件的名称是 `Drive1`、`Drive2` 等。

驱动器列表框的主要属性是 `Drive` 属性，用来在程序运行时设置或返回选定的驱动器，只在运行阶段有效。

如要设置当前的驱动器为 C 盘，若使用的驱动器列表框控件的名称为 `Drive1`，则可在程序中使用如下语句：

```
Drive1.Drive="C:"
```


或

```
Drive1.Drive="C:\\"
```

驱动器的常用方法主要是 `Refresh` 方法，用于刷新驱动器列表。另外驱动器列表框控件也支持 `SetFocus` 方法和 `Move` 方法。

驱动器列表框控件最重要的事件是 `Change` 事件。当驱动器列表框控件的 `Drive` 属性值发生变化时触发 `Change` 事件，该事件主要是为了与目录列表框配合使用的。

10.4.2 目录列表框控件

目录列表框控件(`DirListBox`)用于显示当前驱动器或指定驱动器上的目录结构。目录列表框控件在工具箱中的图标为 ，该控件一般要与驱动器列表框控件联合使用。目录列表框控件的默认名称一般为 `Dir1`、`Dir2`。目录列表框控件在运行时以所在驱动器的根目录开头，各目录按子目录的层次结构依次缩进。运行时用鼠标双击某个目录，就能打开其下的子目录。图 10.2 为目录列表框控件运行时的示意图。

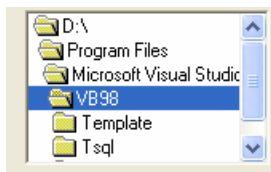


图 10.2 目录列表框控件运行示意图

为了与驱动器列表框控件联合使用，使得目录列表框控件中的根目录与驱动器列表框控件中的驱动器相对应，目录列表框控件提供了相应的属性、方法和事件。

1. 常用属性

目录列表框控件的一个重要属性是 **Path** 属性, 用于设置或返回当前显示的工作目录的路径, 包括驱动器名。该属性只在运行阶段有效。

如果要让目录列表框控件显示目录 C:\ 下的目录结构, 则相应的设置语句为:

```
Dir1.Path="C:\"
```

为了在驱动器列表框控件中的驱动器改变时, 目录列表框随之改变, 则需要在驱动器列表框的 **Change** 事件中设置目录列表框的 **Path** 属性。具体实现过程如下:

```
Private Sub Drive1_Change()  
    Dir1.Path=Drive1.Drive  
End Sub
```


2. 方法

目录列表框控件的常用方法主要有 **Refresh** 方法、**SetFocus** 方法和 **Move** 方法。这些方法可参考驱动器列表框控件的方法。

3. 事件

目录列表框控件最重要的事件是 **Change** 事件。当目录列表框的 **Path** 属性值发生变化时触发 **Change** 事件, 该事件主要是为了与文件列表框配合使用的。

10.4.3 文件列表框控件

文件列表框控件(FileListBox)用来显示指定目录下的文件列表, 该控件在工具箱中的图标为 , 其默认名称为 File1、File2。该控件一般与目录列表框控件配合使用。图 10.3 为文件列表框控件在运行时的示意图。

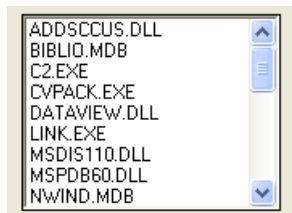


图 10.3 文件列表框控件运行时示意图

文件列表框可以与目录列表框配合使用, 这可以通过它的属性、方法和事件来实现。

1. 常用属性

文件列表框与列表框相似, 也具有 **List**、**ListIndex** 和 **ListCount** 等常用属性, 此外还有文件列表框特有的属性。

(1) Path 属性

该属性用来设置或返回文件列表框所显示的路径全称, 仅在运行阶段有效。一般该属性都是与目录列表框配合使用的, 当目录列表框中的目录发生改变时, 可设置文件列表框

的 Path 属性,使文件列表框显示对应目录中的文件。一般可在目录列表框的 Change 事件中设置,其事件过程如下:

```
Private Sub Dir1_Change()  
    File1.Path=Dir1.Path  
End Sub
```

(2) Pattern 属性

该属性用来设置在文件列表框中要显示的文件类型。通过设置该属性,可对所显示的文件起到过滤效果。该属性可在控件的属性窗口设置,也可在运行阶段设置。例如媒体播放软件如果只播放 AVI 类型文件,则可将 Pattern 属性设置为 "*.avi"。

若要表达的文件类型有多种,各组类型表达式之间应用分号(;)进行分隔。

例如,若仅允许列表框中显示 AVI 文件和 DAT 文件,则相应的设置语句为:

```
File1.Pattern="*.avi ; *.dat"
```

(3) FileName 属性

该属性用于设置或返回文件列表框中被选中的文件名。仅在运行阶段有效。

(4) MultiSelect 属性

该属性用来设定是否允许用户进行多重选择。它的值若为 0 表示只允许单选;若为 1 则表示可直接通过鼠标进行多选;值若为 2 则表示可通过鼠标与 Shift 或 Ctrl 键配合进行多选。

(5) Archive 属性、ReadOnly 属性、System 属性和 Hidden 属性

这些属性分别用于决定是否可显示档案文件、只读文件、系统文件和隐藏文件。它们的类型均为布尔类型。值为 True 或 False。为 True 则可显示对应属性的文件,为 False 则不可显示。

2. 方法

文件列表框的常用方法主要有 Refresh 方法、SetFocus 方法和 Move 方法。

3. 事件

文件列表框可响应很多事件,其中比较特殊的事件有 PathChange 事件和 PatternChange 事件。

(1) PathChange 事件

当文件列表框的 Path 属性值发生变化时触发 PathChange 事件。

(2) PatternChange 事件

当文件列表框的 Pattern 属性值发生变化时触发 PatternChange 事件。

【例 10.7】 设计一个查找文件的程序,可在指定驱动器、指定目录下选择 1 个 TXT 或 DOC 文件,当鼠标双击指定文件时,在文本框控件中显示文件的路径名和文件名。运行界面如图 10.4 所示。

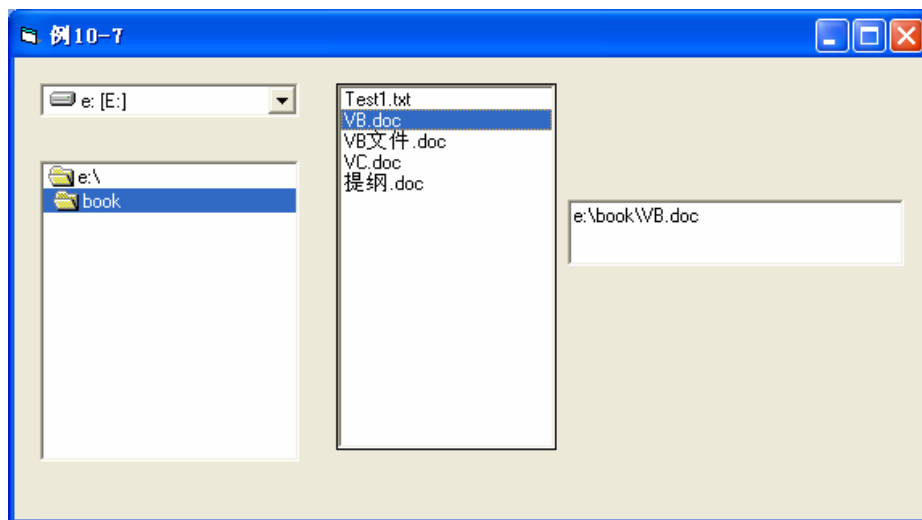


图 10.4 例 10.7 运行界面

程序代码如下：

```
Private Sub Dir1_Change()  
    File1.Path = Dir1.Path          '当目录列表框中的目录发生变化时改变文件列表框  
    Text1.Text = Dir1.Path          '将新的路径显示在文本框中  
End Sub  
Private Sub Drive1_Change()  
    Dir1.Path = Drive1.Drive        '当驱动器列表框中的驱动器发生变化时改变目录列表框  
    Text1.Text = Drive1.Drive       '将新的驱动器显示在文本框中  
End Sub  
'当对选中文件进行双击时触发  
Private Sub File1_DblClick()  
    Text1.Text = File1.Path & "\" & File1.FileName  
'将选中文件的路径和文件名显示在文本框中  
End Sub  
Private Sub Form_Load()  
    File1.Pattern = "*.txt;*.doc"   '设置文件过滤器只显示.txt和.doc文件  
End Sub
```

10.5 文件系统对象模型

10.5.1 文件系统对象模型概述

Visual Basic 的一个新功能是 File System Object(FSO)对象模型。该模型提供了一个基于对象的工具来处理文件夹和文件。这使用户除了使用传统的 Visual Basic 语句和命令及文件系统控件外，还可以使用带有一整套属性、方法和事件的 File System Object 处理文件夹和文件。

FSO 对象模型使应用程序能够创建、改变、移动和删除文件夹，或者检测是否存在指

定的文件夹, 如果存在, 指出在什么地方。FSO 对象模型也能使用户获取关于文件夹的信息, 诸如名称、创建日期或最近修改日期等。

FSO 对象模型使得对文件的处理变得更加简单。使用 FSO 对象, 不需要用户因为文件类型的不同, 而花费太多的精力。

FSO 对象模型编程包括三项主要任务。

(1) 使用 `CreateObject` 方法, 或将一个变量声明为 `FileSystemObject` 对象类型, 来创建一个 `FileSystemObject` 对象。

(2) 对新创建的对象使用适当的方法。

(3) 访问该对象的属性。

FSO 对象模型包含在一个称为 `Scripting` 的类型库中, 此类型库位于 `Scrrun.dll` 文件中。如果还没有引用此文件, 选择【工程】菜单中的【引用】命令, 可打开【选择引用】对话框, 再选择 `Microsoft Scripting Runtime` 项。然后就可以使用对象浏览器来查看其对象、集合、属性、方法以及它的常数。

创建一个 `FileSystemObject` 对象的方法为:

```
Dim fso as new FileSystemObject
```

或

```
Set fso =CreateObject("Scripting.FileSystemObject")
```

其中, `Scripting` 是类型库的名称, 而 `FileSystemObject` 则是要创建一个实例对象的名字。第一种方法在 `Visual Basic` 中有效, 而第二种方法在 `Visual Basic` 或 `VBScript` 中都是可行的。

`FileSystemObject` 提供了管理驱动器、文件夹和文件的方法和属性。

10.5.2 管理驱动器

使用 FSO 可以访问已有的驱动器, 还可获得驱动器的信息。

1. 取得驱动器的名字

要取得一个包含指定路径的驱动器名字的字符串, 可使用 FSO 对象中的 `GetDriveName` 方法, 其语法格式为:

```
object.GetDriveName(path)
```

如果驱动器不能确定, `GetDriveName` 方法返回一个长度为零的字符串("")。`GetDriveName` 方法只对提供的路径字符串起作用。

2. 访问已有的驱动器

要访问一个已有的驱动器, 可以使用 FSO 对象中的 `GetDrive` 方法, 该方法返回指向某个已有驱动器的 `Drive` 对象。其语法格式如下:

```
object.GetDrive DriveSpec
```

其中, `DriveSpec` 为一个驱动器字符加冒号和路径分隔符(如 `"C:\"`)或任何网络共享的说明(如 `"\\computer2\Share1"`)。如果 `DriveSpec` 不符合任何一种可以接受的形式或者不存在, 则发生一个错误。

3. Drive 对象

Drive 对象允许获得一个系统的各个驱动器的信息, 这些信息可以是物理的, 也可以是网络上的。通过该对象的属性可以获得下列信息。

- (1) 以字节表示的驱动器总空间(TotalSize 属性);
- (2) 以字节表示的驱动器可用空间(AvailableSpace 或 FreeSpace 属性);
- (3) 为驱动器指定的符号(DriveLetter 属性)和驱动器类型(诸如可移动的、固定的、网络上的、CD-ROM 或者 RAM 盘);
- (4) 驱动器序列号(SerialNumber 属性);
- (5) 驱动器使用的文件系统类型, 如 FAT、FAT32、NTFS 等(FileSystem 属性);
- (6) 驱动器是否可用(IsReady);
- (7) 共享和/或卷标的名称(ShareName 和 VolumeName 属性);
- (8) 驱动器的路径或根文件夹(Path 和 RootFolder 属性)。

【例 10.8】 使用 Drive 对象收集有关驱动器的信息。

新建一个标准工程, 再添加引用 Microsoft Scripting Runtime, 然后向窗体中添加一个命令按钮, 最后给命令按钮添加如下代码。注意在下面的代码中并没有对实际 Drive 对象的引用, 而是使用 GetDrive 方法获得一个对已有 Drive 对象的引用。

```
Private Sub Command1_Click()  
    ' 定义一个 FSO 对象和 Drive 类型变量及 String 类型变量  
    Dim fso As New FileSystemObject, drv As Drive, s As String  
    ' 取得 c: 驱动器对象  
    Set drv=fso.GetDrive(fso.GetDriveName("c:"))  
    s="驱动器: "&UCase("c:") &"-"  
    ' 取得 c 盘的卷标名称  
    s=s& drv.VolumeName &vbCrLf  
    ' 取得 c 盘总空间数  
    s=s & "总空间:" & FormatNumber(drv.TotalSize /1024,0)  
    s=s & "kb" &vbCrLf  
    ' 取得 c 盘可用空间数  
    s=s & "可用空间:" & FormatNumber(drv.FreeSpace/1024,0)  
    s=s & "kb" &vbCrLf  
    MsgBox s  
End Sub
```

10.5.3 管理文件夹

1. Folder 对象

Folder 对象允许对文件夹进行操作, 该对象可以通过 CreateFolder 方法新建, 如:

```
Dim fso As New FileSystemObject, fdr As Folder  
fdr=fso.CreateFolder("E:\myvb")
```

也可以通过 FSO 对象的 GetFolder 获得已存在的文件夹对象, 如:

```
fdr=fso.GetFolder("E:\vb")
```

2. 文件夹的操作

有关文件夹的操作和方法如表 10-2 所示。

表 10-2 文件夹的操作方法

操 作	方 法
创建一个文件夹	FSO 对象的 CreateFolder 方法
删除一个文件夹	Folder 对象的 Delete 或 FSO 对象的 DeleteFolder 方法
移动一个文件夹	Folder 对象的 Move 或 FSO 对象的 MoveFolder 方法
复制一个文件夹	Folder 对象的 Copy 或 FSO 对象的 CopyFolder 方法
检索文件夹的名称	Folder 对象的 Name 属性
查找一个文件夹是否在驱动器上	FSO 对象的 FolderExists 方法
获取已有 Folder 对象的一个实例	FSO 对象的 GetFolder 方法
获取一个文件夹的父文件夹的名称	FSO 对象的 GetParentFolderName
获取系统文件夹的路径	FSO 对象的 GetSpecialFolder 方法

下面通过例 10-9 来阐述使用 FSO 对象的方法及其对文件夹进行的操作。

【例 10.9】 操作文件夹并获取其信息。

```
Private Sub Command2_Click()  
    ' 定义一个 FSO 对象和 Folder 类型变量及 String 类型变量  
    Dim fso as New FileSystemObject, fdr as folder, s as String  
    Set fdr=fso.GetFolder("E:") ' 获得 Folder 对象  
    Print "父文件夹名是: " & fdr  
    Print "文件夹所在驱动器名称: " & fdr.drive  
    if fdr.IsRootFolder=true then ' 若该文件夹为根文件夹  
        Print "该文件夹是根文件夹"  
    end if  
    fso.CreateFolder("E:\MyVB") ' 创建文件夹 E:\MyVB  
    Print("创建了文件夹 E:\MyVB")  
    fdr=fso.GetFolder("E:\MyVB")  
    fdr.Delete ' 使用 Folder 对象删除文件夹  
End Sub
```

10.5.4 管理文件

1. File 对象

File 对象用来对文件进行相应的操作。它可通过 FSO 对象的 GetFile 方法来创建,例如:

```
Dim fso As new FileSystemObject, fil as File  
Set fil=fso.GetFile("E:\VB\Test.txt")
```

2. TextStream 对象

该对象可对文本文件进行读写操作,它可通过 FSO 对象的 CreateTextFile 方法来创建一个空的文本文件,并返回 TextStream 对象。

```
Dim fso As new FileSystemObject, fil as File
Set fil=fso.CreateTextFile("E:\VB\Test.txt",True)
```

本方法的第 2 个参数为布尔型值，若为真表示对已存在的文件进行覆盖操作。

3. 文件的操作

(1) 添加数据到文本文件

文件一经创建，就可以分 3 步向其中加入数据。

① 打开文件，可以使用下面两种方法中的任一种：File 对象的 `OpenAsTextStream` 方法或 FSO 对象的 `OpenTextFile` 方法来创建一个 `TextStream` 对象。

② 向打开的文本文件中写入数据，可以使用 `TextStream` 对象的 `Write` 或 `WriteLine` 方法。它们之间的唯一区别是 `WriteLine` 在指定的字符串末尾添加换行符。如果向文本文件中添加一个空行，则使用 `WriteBlankLines` 方法。

③ 使用 `TextStream` 对象的 `Close` 方法，关闭一个已打开的文件。

(2) 读取文件

要从一个文本文件中读取数据，可使用 `TextStream` 对象的 `Read`、`ReadLine` 或 `ReadAll` 方法。从一个文件中读取指定数量的字符用 `Read` 方法，读取一整行用 `ReadLine` 方法，读取一个文本文件的所有内容用 `ReadAll` 方法。如果使用 `Read` 或 `ReadLine` 方法并且要跳过数据的某些部分，可以使用 `Skip` 或 `SkipLine` 方法。这些读取方法产生的文本被存储在一个字符串中，而这个字符串可以在控件中显示，也可以被字符串操作符分解或合并。

(3) 移动、复制和删除文件

可使用 File 对象的方法和 FSO 对象的方法对文件进行移动、复制和删除操作。

File 对象的方法 `Move` 用来移动文件，方法 `Copy` 用来复制文件，方法 `Delete` 用来删除文件。

FSO 对象的方法 `MoveFile` 用来移动文件，方法 `CopyFile` 用来复制文件，方法 `Delete` 用来删除文件。

【例 10.10】 文件操作的使用示例。

本例首先在 E 盘的根目录下创建一个文本文件，并向其中写入一些信息，再将该文件移至“E:\VB”目录下，同时复制到“E:\MyVB”目录中，并打开“E:\MyVB”目录下的文件，读取后显示在消息框中，最后将这两个目录中的复制文件删除。

新建一个标准工程，再添加引用 `Microsoft Scripting Runtime`，然后向窗体中添加一个命令按钮，最后给命令按钮添加如下代码。

```
Sub Command1_Click()
    ' 定义一个 FSO 对象和 TextStream 类型变量及 File 类型变量
    Dim fso As new FileSystemObject, txtfile as TextStream, fl as File
    Set txtFile=fso.CreateTextFile("E:\Test.txt",true)
    ' 创建一个空文件并返回 TextStream 对象
    txtFile.Write("这是对文件的一个操作") ' 使用 TextStream 对象的 Write 写入数据
    txtfile.Close ' 关闭文件
    Set fl=fso.GetFile("E:\Test.txt") ' 取得 TextStream 对象
```

```

fso.CopyFile("E:\Test.txt", "E:\MyVB\Test.txt")
'使用 FSO 对象进行文件复制
fl.Move("E:\VB\Test.txt")           '使用 File 对象进行文件移动
Set fl=fso.GetFile("E:\VB\Test.txt") '取得 File 对象
Set txtFile=fl.OpenAsTextStream      '通过 fl 取得 TextStream 对象
MsgBox txtFile.ReadAll               '读取文件的全部内容, 在消息框中显示
txtFile.Close
fl.Delete                            '使用 File 的 Delete 方法删除文件
fso.DeleteFile("E:\MyVB\Test.txt")   '使用 fso 的方法删除文件
End Sub

```

注意在使用该代码时要求文件夹 E:\VB 和 E:\MyVB 必须存在。

10.6 文件应用举例

下面利用文件的功能实现一个简单的通信录程序, 该程序可添加新的通信录信息, 也可在列表中选择某一个人的姓名, 并得到其手机号码和住宅电话。

该程序的运行效果如图 10.5 所示。

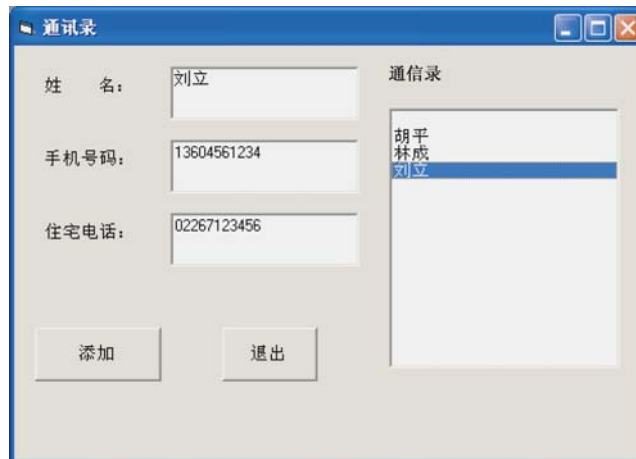


图 10.5 通信录程序运行界面

该程序窗体上有 3 个标签控件和 3 个文本框控件、1 个列表框控件和 2 个命令按钮控件。除了 Caption 属性外, 其他属性均取默认值。当单击【添加】按钮时, 可将 3 个文本框中的信息加入到列表框及文件中。为了方便查找, 在读取文件时, 将文件中所有记录读入到对应的数组中。

该通信录的程序代码如下:

```

Dim FileNumber As Integer
Dim Length As Long
Dim N As Long
Private Sub Command1_Click()
    '记录记录号
    '添加按钮的单击事件
    '将新信息存入数组第 N-1 个元素中

```

```

    Contact(N - 1).Name = Trim(Text1.Text)
    Contact(N - 1).Mobile = Trim(Text2.Text)
    Contact(N - 1).Phone = Trim(Text3.Text)
    '将新信息存入到文件第N条记录处
    Put #FileNumber, N, Contact(N)
    '将新信息的姓名添加到列表框中
    List1.AddItem Contact(N - 1).Name, N - 1
    N = N + 1
    '清空文本框中的信息
    Text1.Text = ""
    Text2.Text = ""
    Text3.Text = ""
End Sub
Private Sub Command2_Click()      '关闭按钮的单击事件
    Close #FileNumber
    Unload Me
End Sub
Private Sub Form_Load()
    FileNumber = FreeFile          '取得空闲文件号
    Length = Len(ContactInfo)
    N = 1
    Open "E:\contact.dat" For Random As #FileNumber '打开随机文件
    RefreshList                    '取得已有的通信录
End Sub
Private Sub RefreshList()
Do While Not EOF(FileNumber)
    '将文件中的通信录信息依次读入数组中
    Get #FileNumber, N, Contact(N - 1)
    '将每条记录的姓名填入列表框中
    List1.AddItem Contact(N - 1).Name, N - 1
    N = N + 1
Loop
End Sub
'单击列表框中的姓名, 将其对应的电话信息显示在文本框中
Private Sub List1_Click()
    I = List1.ListIndex
    Text1.Text = Contact(I).Name
    Text2.Text = Contact(I).Mobile
    Text3.Text = Contact(I).Phone
End Sub
'下面的代码要放到模块中
Type ContactInfo
    Name As String * 8
    Mobile As String * 11
    Phone As String * 12
End Type
Public Contact(100) As ContactInfo

```

该程序只实现了有限的功能, 感兴趣的同学可增加删除功能、修改功能及控制信息的输入格式和不允许重名功能。

10.7 小 结

文件是具有相同类型数据的集合，保存在外存储器上，且以文件名作为标识。

文件的基本操作有打开文件、读/写文件和关闭文件。

打开文件的作用是指定操作文件名、操作方式和建立缓冲区。打开文件用 **Open** 语句。

读/写文件是通过对文件指针的操作，在内存和文件之间传输数据。读是指把文件中的数据送入内存，写是指把内存中的数据写入文件。

文件读写完成后，需要关闭文件。关闭文件包括处理缓冲区、释放内存以及终止程序与文件的联系。关闭文件使用 **Close** 语句。

在 **Visual Basic** 中，有 3 种文件访问类型的文件，即顺序型文件、随机型文件和二进制型文件。

DriveListBox、**DirListBox** 和 **FileListBox** 是 3 个关于文件操作的控件。

Visual Basic 的 **File System Object(FSO)**对象模型提供了一个基于对象的工具来处理文件夹和文件。**FSO** 对象模型使得对文件的处理变得更加简单。使用 **FSO** 对象，不会因为文件类型的不同，而花费用户太多的精力。

FSO 对象模型编程包括三项主要任务。

- (1) 使用 **CreateObject** 方法，或将一个变量声明为 **FSO** 对象类型，来创建一个 **FSO** 对象。
- (2) 对新创建的对象使用适当的方法。
- (3) 访问该对象的属性。

10.8 习 题

一、填空题

1. 按照文件的特征属性，可将文件分为_____、_____、_____和_____。
2. 可以对顺序文件进行格式化写操作的语句是_____。
3. 一个文件在使用前必须先_____，使用后必须_____。
4. **Visual Basic 6.0** 中包括 3 个文件系统控件，分别是_____、_____和_____。
5. 若想要使用 **FSO** 对象，必须在工程菜单中的引用对话框中选择_____项。
6. 文件的移动可使用 **FSO** 对象的_____方法或者 **File** 对象的_____方法。

二、选择题

1. 下列哪些语句和函数可用于随机文件的操作？()
A) Input # B) Type...End Type C) Write # D) Print #

2. 如果使用 Open 语句打开随机文件进行操作, 则 For 子句后应使用的打开模式应该为? ()。

- A) Append B) Input C) Random D) Output

3. 若要设置文件列表控件的路径, 则需要使用属性()

- A) FileName B) Pattern C) Path D) Drive

4. 使用文件系统对象模型对文件进行操作时, 如果向打开的文本文件添加一个空行, 需要使用 TextStream 对象的()方法。

- A) WriteBlankLines B) Write C) WriteLine

三、简答题

1. 文件列表框的 FileName 属性是否包含路径?
2. 随机文件和二进制文件的读写操作有何不同?
3. 如何用 FSO 对象模型来管理文件?
4. 如何将文件列表框控件和目录列表框控件配合使用?

四、编程题

1. 向一个顺序文件中写入多个字符串, 然后将每个字符串中的所有小写字母都改为大写字母。

2. 编写一个程序, 输入一个班所有学生的 3 门课程的成绩信息, 其中每位学生的成绩信息包括学号、姓名、高等数学成绩、大学英语成绩、程序设计成绩等, 将输入的学生成绩信息存放到成绩文件 Score.dat 中。

第 11 章 数据库程序设计

教学提示：数据库程序设计是计算机应用技术中的一个重要组成部分，对于大量的数据，使用数据库来存储管理可以降低数据的冗余度，提高数据存储效率。Visual Basic 中提供了多种访问数据库的方法，可以访问的数据库类型有 dBase、FoxPro 和 Access 等本地数据库。另外，还可以通过 ODBC 方式访问 MS SQL Server、Oracle SQL Server 和 Sybase SQL Server，并以客户/服务器方式存取数据库中的数据。Visual Basic 提供的数据库访问方法主要有：使用数据(Data)控件或 ADO 数据控件访问数据库。通过 ODBC 方式访问远程数据库以及采用 ADO 对象模型访问数据库等。

教学目标：掌握可视化数据管理器的使用方法，能熟练地建立数据库表，对数据进行必要的管理；熟练使用数据控件、数据绑定控件及 ADO 控件和 ADO 对象模型访问本地数据库。

11.1 数据库和 SQL 语言基础

11.1.1 数据库简介

数据库(Database)是指一组排列成易于处理或读取的相关数据的有序集合，它是由一个或多个表对象组成的集合。它类似于 Excel 的工作簿和工作表。

按数据组织形式可以将数据库分为层次型、网状型和关系型结构，其中最常用的是关系型数据库。关系型数据库模型把数据用表的集合来表示，通过建立简单表之间的关系来定义结构，而不是根据数据的物理存储方式来建立数据中的关系。不管表在数据库文件中的物理存储方式如何，都可以把它看成一组行和列。下面介绍几个关系型数据库的相关概念。

1. 表

表是一种按行与列排列的相关信息的逻辑组，类似于工作表。例如，一张表可以包含一个班级学生的一系列信息，如姓名、性别、年龄、地址、相片、每门功课的成绩等。

2. 字段

数据库表中的每一列称为一个字段。表是由其包含的各种字段定义的，每个字段描述了它所包含的数据。创建数据库时，需要为每个字段分配一个数据类型、最大长度和其他属性。字段可包括各种字符、数字甚至图形。

3. 记录

表中的每一行称为一条记录。一般来说，数据库表的记录必须惟一，即数据库表的记录创建时任意两行都不能完全相同。例如，学生档案中不能有两个名字、性别、地址等字段完全相同的学生记录。

4. 记录集

记录集是表或查询中的记录集合。Visual Basic 使用记录集来检索和显示数据库中的数据。记录集通常有 3 种类型：表(Table)、动态集(Dynaset)和快照(Snapshot)。

Table 类型的记录集代表了数据库中单个的表。它的 RecordSet 对象是当前数据库中真实的数据表，因此对于 Table 类型的记录集的处理速度比其他记录集类型都快，但它需要大量的内存空间。

Dynaset 类型的记录集是一个动态记录集合的本地副本。Dynaset 类型的记录集的记录字段不但可以取自同一个表，还可以取自多个表，这使得它与 Table 类型的记录集相比具有更大的灵活性。尽管 Dynaset 类型的记录集是一个本地副本，它存储于本地内存中，但是，Dynaset 类型的记录集是和原始数据库中的表相关联的，因此，用户仍然可以对 Dynaset 类型的记录集进行添加、删除、修改等操作，并且和 Table 类型的记录集一样，这些操作都会被保存到相应的数据库中。

Snapshot 类型的记录集同 Dynaset 类型的记录集相似，但它是一个静态的本地副本。也就是说，用户不能通过 Snapshot 类型的记录集对原始表中的记录进行修改，而只能以只读方式浏览。与 Dynaset 类型的记录集一样，Table 类型的记录集也是存储在本地内存中的。

5. 索引

为了便于查找，可以在数据库中建立索引来加快查找速度，大多数数据库都使用索引。数据库表的索引是比表搜索更快的排序列表。每个索引输入项指向其相关的数据库行。当与 Data 控件一起使用表类型的记录集时，表的主索引可用于加速检索操作。建立数据库时，可以设置任意多个索引字段。虽然索引表是不可见的，却同样会占据磁盘空间。选择索引字段时，应该尽量选择惟一字段。

11.1.2 结构化查询语言

结构化查询语言 SQL 是操作数据库的工业标准。SQL 语言进行数据操作，只要提出“做什么”，而无须指明“怎么做”，SQL 语句的操作过程由系统自动完成。SQL 中最经常使用的是从数据库中获取数据，从数据库中获取数据称为查询数据库，查询数据库通过使用 SELECT 语句实现。常见的 SELECT 语句包含六部分，其语法形式为：

```
SELECT 字段列表
FROM 表名
[WHERE 查询条件]
GROUP BY 分组字段
        HAVING 分组条件
ORDER BY 字段 [ASC|DESC] ]
```

其中：

(1) 字段列表部分包含了查询结果要显示的字段清单, 字段之间用逗号隔开。要选择表中的所有字段, 可以用“*”代替。如果要对选定的字段进行更名, 可在该字段后用“AS[新名]”来实现。如果需要查询的某字段在数据表中没有, 而是通过若干字段进行运算的结果, 则可以用 SQL 提供的基本的运算符(“+”、“-”、“*”、“/”), 以及重命名关键字“AS”来实现。例如: 在学生成绩数据表中通过已有的“期末考试成绩”、“平时成绩”来计算学生的“综合成绩”, 则可以写成:

```
SELECT 期末考试成绩*0.7+平时成绩*0.3 AS 综合成绩
```

(2) WHERE 子句用于限制记录的选择, 构造查询条件可使用大多数 Visual Basic 内部函数和运算符(“<”、“<=”、“>”、“>=”、“<>”、“+”、“-”、“*”、“/”等)以及 SQL 特有的运算符(“Between”、“Like”和“In”运算符等)构成表达式。

其中, 比较运算符“<”、“<=”、“>”、“>=”、“<>”主要与数值类型的字段一起使用, 比较运算只能在数据类型相似的字段、列之间进行; 比较运算符“=”和“<>”也可以用于文本和算术数据类型的字段, 查询的值为文本时, 用单引号将文本字段值括起来; 算术运算符“+”、“-”、“*”、“/”用于执行算术运算。

“Between”运算符用于数值或日期类型的字段, 语法形式为: field_fieldname Between Value1 and Value2, 要求在日期的数值两边加上标志#, 如: 报到日期 Between #2005-08-23 and 2005-09-3#。

“Like”运算符用来找出符合指定条件的记录可以指定完整的值(例如, Like“字段值”), 或使用通配符来指定查询的范围。语法形式为: field_fieldname like “字段值”, 如: 姓名 Like “*dia” 表示在“姓名”字段中所有以“dia”结尾的记录, 如 Like “李*”, 则查询返回该字段中所有姓“李”的学生。

“In”运算符用于判断表达式的数值是否等于指定列表中几个数值中的一个。例如: 地区 in (“华东”、“华南”、“华北”) 表示在“地区”字段值等于“华东”、“华南”、“华北”其中之一的所有记录。

(3) GROUP BY 和 HAVING 子句用于分组和分组过滤处理。它能把指定字段中有相同值的记录合并成一条记录。如果在 SELECT 语句中含有 SQL 合计函数, 例如 SUM 或者 COUNT 函数, 那么就为每条记录创建摘要值。在 GROUP BY 字段中的 NULL 值会被分组, 并不省略。但是, 在任何 SQL 合计函数中都计算 NULL 值。可用 WHERE 子句来排除不想分组的行, 将记录分组后, 也可以用 HAVING 子句进行筛选。一旦 GROUP BY 子句完成了记录分组, HAVING 子句就显示由 GROUP BY 子句分组的并且满足 HAVING 子句条件的所有记录。

(4) ORDER BY 子句决定了查找出来的记录的排列顺序, 在 ORDER BY 子句中可以指定一个或者多个字段作为排序关键字, ASC 选项代表升序, DESC 选项代表降序。

在上述 SQL 语句中, SELECT 和 FROM 子句是必须要有的, 它指明 Visual Basic 从何处寻找想要的数据库, 通过 SELECT 语句可返回一个记录集。

可在 SELECT 子句内使用合计函数对记录进行操作, 它返回一组记录的单一值。例如, AVG 函数可以返回记录集的特定字段中所有值的平均数。表 11-1 列出了合计函数。

表 11-1 合计函数

合计函数	描 述
AVG	用来获取特定字段中的值的平均数
COUNT	用来返回选定记录的个数
SUM	用来返回特定字段中所有值的总和
MAX	用来返回指定字段中的最大值
MIN	用来返回指定字段中的最小值

11.2 可视化数据管理器

Visual Basic 所支持的不同类型的数据库可以通过相关的数据库管理系统来建立。可视化数据管理器(VisData)是 Visual Basic 的数据库设计工具,使用这个工具不但能够创建多种格式的数据库,而且可对它们进行浏览和其他操作。除此之外,VisData 还有其他功能,如修复和压缩 MDB 数据库。

11.2.1 启动可视化数据管理器

打开可视化数据管理器的方法是单击【外接程序】主菜单的【可视化数据管理器】选项,进入可视化数据管理器(图 11.1),在其中对数据库进行各种操作。通过交互方式可以实现如下数据库管理器的基本功能。

- (1) 打开已经存在的数据库;
- (2) 新建一个指定类型的数据库;
- (3) 建立、新增及删除数据表;
- (4) 建立、增加字段;
- (5) 建立、增加、更正、删除和查询记录;
- (6) 建立、增加和删除索引;
- (7) 整理数据库。

可视化数据管理器的缺点是无法修改原有数据库的字段名称、类型和字段的长度。



图 11.1 可视化数据管理器

11.2.2 新建数据库

在 Visual Basic 中, 利用可视化数据库管理器可以创建多种类型的数据库, 如: Access、FoxPro、dBASE、Excel 等格式。其中 Access 数据库又包括 Access 2.0 和 Access 7.0 两种版本。另外, FoxPro、dBASE 和 Paradox 也包含不同版本。具体的创建步骤如下。

单击【外接程序】主菜单的【可视化数据管理器】选项, 打开可视化数据管理器(见图 11.1)。可视化数据管理器【文件】菜单中的选项功能描述如表 11-2 所示。

可视化数据管理器【实用程序】菜单中的选项功能描述如表 11-3 所示。

表 11-2 【文件】菜单中的选项功能

选 项	功能描述
打开数据库	打开指定的数据库
新建	根据所选类型建立新数据库
导入/导出	从其他数据库中导入数据表, 或导出数据表及 SQL 查询结果
工作空间	显示注册对话框, 注册新工作空间
压缩 MDB	压缩指定的 Access 数据库
修复 MDB	修复指定的 Access 数据库

表 11-3 【实用程序】菜单中的选项功能

选 项	功能描述
查询生成器	建立、查看、执行和存储 SQL 查询
数据窗口设计器	创建数据窗体并将其添加到 VB 工程中
全局替换	创建 SQL 表达式并更新所选数据表中满足条件的记录
附加	显示当前 Access 数据库中所有附加数据表及连接条件
用户组/用户	查看和修改用户组、用户、权限等设置
System.mda	创建 System.mda 文件, 以便为每个文件设置安全机制
首选项	设置查询或登录超时值以及启动时是否打开上一次使用的数据库文件

单击可视化数据管理器【文件】菜单中的【新建】选项, 出现图 11.2 所示的对话框, 用户指定一个数据库文件名(可以包括路径), 就会出现如图 11.3 所示的窗口。

图 11.3 左边的数据库窗口用于列出指定数据库所包含的表名及结构(但这个数据库是新建的, 没有用户表, 只是显示数据库的一些基本属性)。

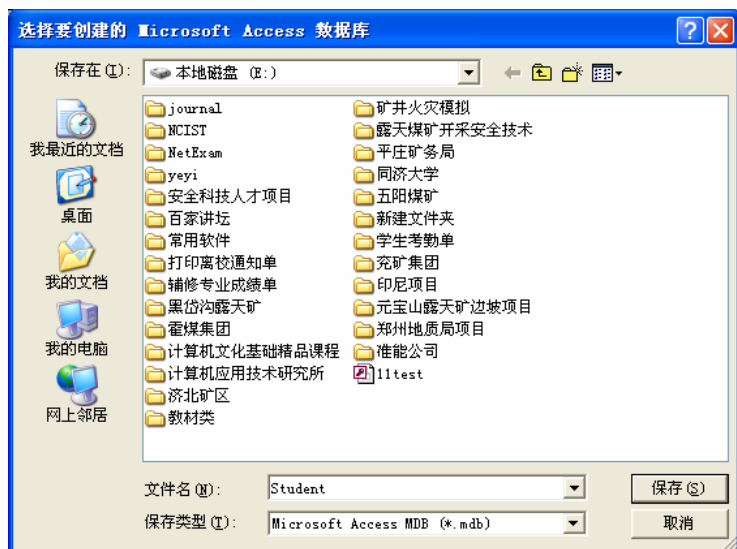


图 11.2 指定新建一个 Access 数据库的名称和路径

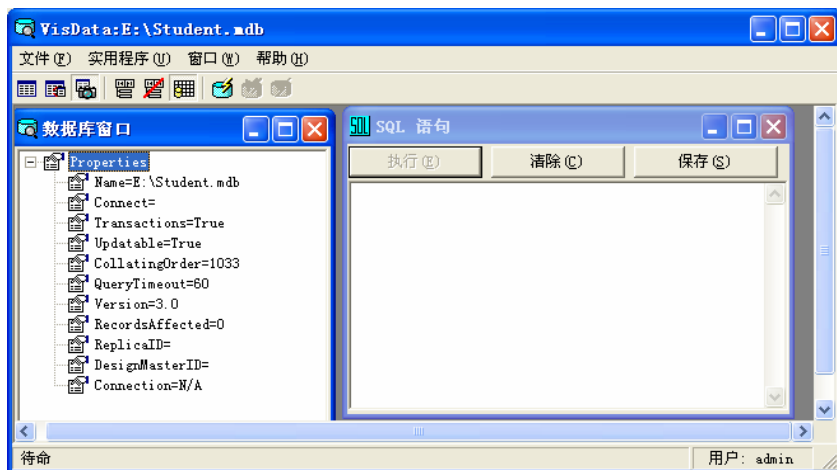


图 11.3 新建一个 Access 数据库

11.2.3 打开数据库

打开数据库的操作和新建数据库的操作基本类似，只是需要保证打开的数据必须在硬盘上存在。具体的创建步骤如下。

单击【外接程序】主菜单的【可视化数据管理器】选项，打开可视化数据管理器(图 11.1)。使用可视化数据管理器【文件】菜单中的【打开】选项，选择需要打开的数据库类型，然后指定数据库的全路径即可。

11.2.4 添加数据表

下面以学生基本情况表来说明使用数据管理器建立 Student.mdb 数据库的过程。学生基本情况表结构如表 11-4 所示。

表 11-4 学生基本情况表

字段名	类型	宽度	字段名	类型	宽度
学号	Text	12	专业	Text	20
姓名	Text	10	出生年月	Date	8
性别	Text	2	照片	Binary	

在 VisData 中,单击【文件】菜单下的【新建】选项,出现数据库类型选择菜单。单击菜单中的 Microsoft Access,将出现版本子菜单,在版本菜单中选择要创建的数据库版本,如: Microsoft Access 7.0 版本。

选定要创建的数据类型及版本后,出现【新建数据库】对话框,在文件名框中输入要创建的数据库文件名,如:“E:\Student”,单击【保存】按钮(图 11.2)。

在可视化数据库管理器窗口中出现【数据库窗口】和【SQL 语句】窗口,如图 11.3 所示。在数据库窗口中只有 Properties 一项,单击其前面的“+”可以以树型结构显示库的具体属性。

【SQL 语句】窗口主要用来执行数据查询语句,并且在该窗口中可以执行和清除 SQL 语句,如需要保存 SQL 语句可以单击【保存】按钮。

另外,在数据库窗口中单击鼠标右键打开快捷菜单,在此菜单中有【新建表】和【刷新列表】等选项。当要创建新表时,单击【新建表】选项,即可打开如图 11.4 所示的【表结构】对话框,在该对话框中可以进行建立一个新表的基本操作,如输入新表的名称、添加字段、删除字段等。

图 11.4 【表结构】对话框

其中，【表名称】文本框用于输入数据库的表名，在本例中输入“学生基本情况表”，单击【添加字段】按钮，打开如图 11.5 所示的【添加字段】对话框，在该对话框中可为数据库表添加字段。【添加字段】对话框内各选项功能描述见表 11-5。首先在【名称】框中输入添加字段的名称，然后在【类型】项中进行合适的选择。选定类型后，【大小】文本框中出现该字段类型的默认长度，也可根据需要输入字段长度。另外，还可在【添加字段】对话框中设置其他项。各项设置完后，单击【确定】按钮，可以添加下一个字段。最后，单击【关闭】按钮，返回【表结构】对话框。

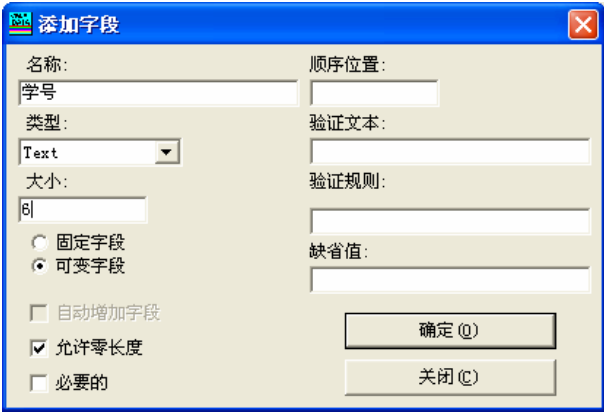


图 11.5 【添加字段】对话框

表 11-5 【添加字段】对话框中各选项说明

选 项 名	描 述
名称	字段名称
类型	指定字段的类型
大小	字段宽度
固定字段	字段宽度固定不变
可变字段	字段宽度可变
允许零长度	表示空字符串可作为有效的字段值
必要的	表示该字段值不可缺少
顺序位置	字段在表中的顺序位置
验证文本	当向表中输入无效值时所显示的提示
验证规则	验证输入字段的简单规则
默认值	在输入时设置的字段初始值

此时，在【表结构】对话框中的【字段列表】中显示了所添加的所有字段，单击各个字段，可以浏览该字段属性的各项设置，如图 11.6 所示。

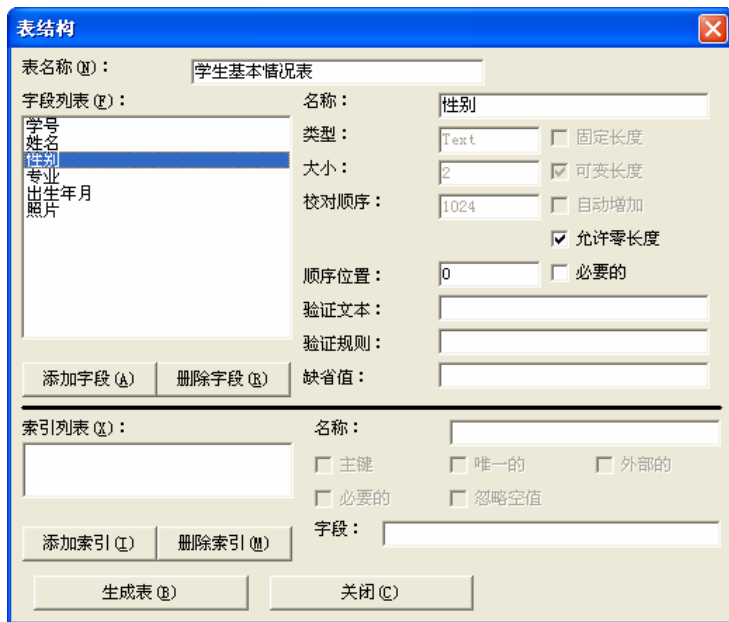


图 11.6 添加字段后的表结构图

在该对话框中单击【添加索引】按钮后，出现添加索引对话框，如图 11.7 所示。在该对话框中的【名称】框中输入建立索引表的名称，其中在【可用字段】列表中，列出了所有可以用做索引的字段，以供用户选择。单击【确定】按钮，将当前索引表加入表结构中。同时还可以再单击【添加索引】按钮新建其他索引表。所有索引表建立后，单击【关闭】按钮，关闭添加索引对话框。通过添加索引对话框可以将数据表中的某些字段设置为索引，以加快查找速度。

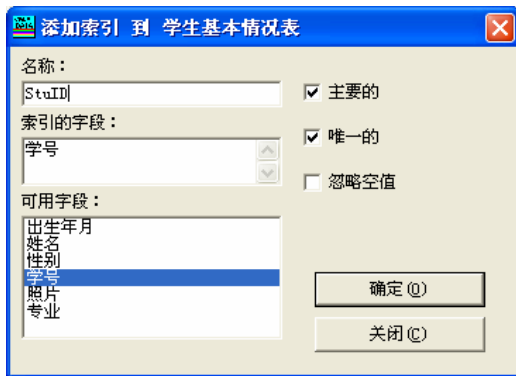


图 11.7 添加索引对话框

在图 11.6 所示的对话框中，单击【生成表】按钮，就会出现如图 11.8 所示的学生基本情况表。双击“学生基本情况表”前面的“+”、“-”号可以将其展开或者折叠。

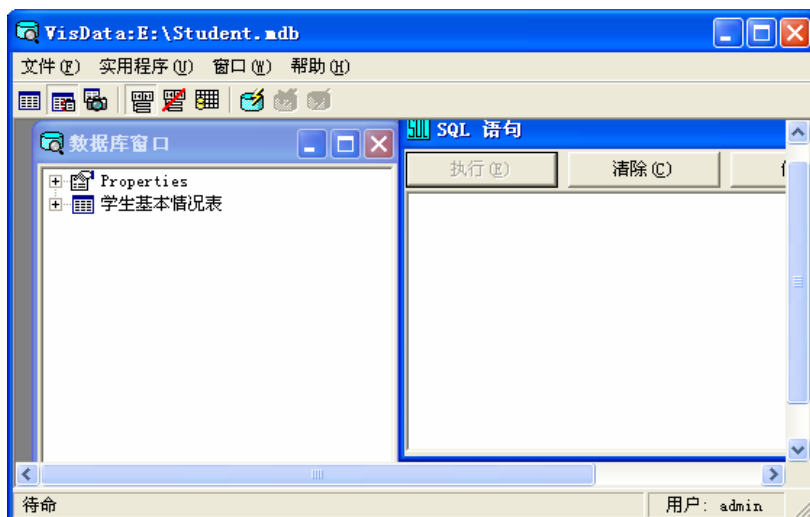


图 11.8 表结构建立后的数据库管理器

11.2.5 数据的增加、删除和修改

在图 11.8 所示的数据库窗口中，应注意在工具栏中使【动态集类型记录集】和【在新窗体上使用 Data 控件】按钮被选中，右击“学生基本情况表”，在快捷菜单中选择【打开】选项，即可向表中添加记录，如图 11.9 所示。为了便于后面的使用，请输入若干条记录。若要修改该表的结构，可在快捷菜单中选择【设计】选项，便可以返回到图 11.4 进行修改。

当表建好后，用鼠标双击数据库窗口中的学生基本情况表，打开如图 11.9 所示的对话框，选择对应的选项进行添加、编辑和删除操作。图 11.9 中各按钮功能描述如表 11-6 所示。



图 11.9 添加记录对话框

表 11-6 图 11.9 中各按钮功能描述

按钮名称	描 述
添加	数据区清空，等待数据输入
更新	保存新输入的数据，更新数据库
删除	删除选中的当前记录
查找	设置查找条件，查找数据
刷新	取消上次更新以后的改变
关闭	关闭当前窗口

11.2.6 数据的查询

“查询生成器”是一个用来构造 SQL 查询的表达式生成器，可用来生成、查看、执行和保存 SQL 查询。生成的查询条件将作为数据库的一部分保存。使用查询生成器建立查询的步骤如下。

(1) 打开可视化数据管理器和要建立查询的数据库。从【实用程序】菜单中选择【查询生成器】选项，打开【查询生成器】对话框(图 11.10)。在【表】列表对话框中列出了该数据库中包含的所有数据表。

(2) 在【表】列表框中，单击要查询的表，该表中的所有字段将出现在【要显示的字段】列表框中，在该框中选中所有查询时要显示的列的字段名，使它们处于选中状态。查询还可以在多个表之间进行，选择多个表后，可以选择不同表的字段组成查询结果的列，通常还要建立表间的连接。

(3) 在对话框上部有【字段名称】、【运算符】和【值】3 个下拉列表框，它们用来构成一个查询条件表达式。【字段名称】中列出了选中表的所有字段，【运算符】中列出了选中的字段可以参加的关系运算，主要有“>”、“>=”、“<”、“<=”、“<>”和“Like”等，在【值】下拉列表框中可以输入条件值，也可以单击【列出可能的值】按钮，选择字段的取值添加到【值】列表中。

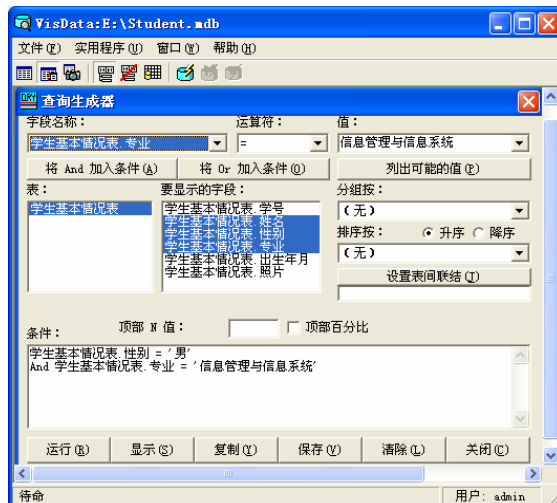


图 11.10 【查询生成器】对话框

(4) 单击【将 And 加入条件】或【将 Or 加入条件】按钮, 可以将条件依次添加到【条件】列表框中。查询条件表达式可以是由多个条件组合而成的逻辑表达式。单击【清除】按钮可以删除条件。

(5) 要查看查询条件, 可以单击【显示】按钮, 打开【SQL 查询】对话框, 如图 11.11 所示。单击【复制】按钮, 可以将 SQL 查询复制到 SQL 语句窗口。

(6) 查询条件设定后, 可以单击【运行】按钮查看结果。这时将弹出如图 11.12 所示的对话框。单击【否】按钮后, 将显示查询结果, 如图 11.13 所示。单击【关闭】按钮可以结束查询。

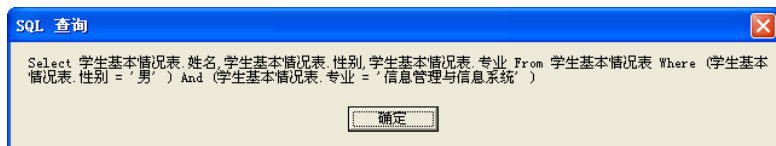


图 11.11 【SQL 查询】对话框

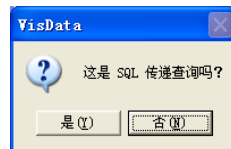


图 11.12 确认对话框

(7) 如果要保存创建的查询, 则在【查询生成器】对话框中单击【保存】按钮, 即可打开保存对话框, 如图 11.14 所示, 输入查询名称, 将查询保存到数据库中。

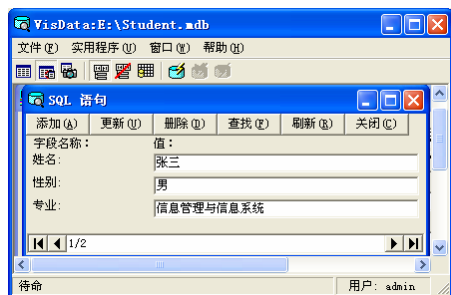


图 11.13 SQL 查询结果窗口

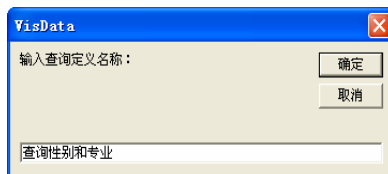


图 11.14 输入查询名

保存查询后, 在数据库窗口中就可以看到刚才建立的查询。以后要执行该查询, 只需在数据库窗口中双击该查询名。同时, 可以选择该查询右键快捷菜单中的【设计】选项, 在【SQL 语句】窗口中对该查询条件进行修改, 如图 11.15 所示。

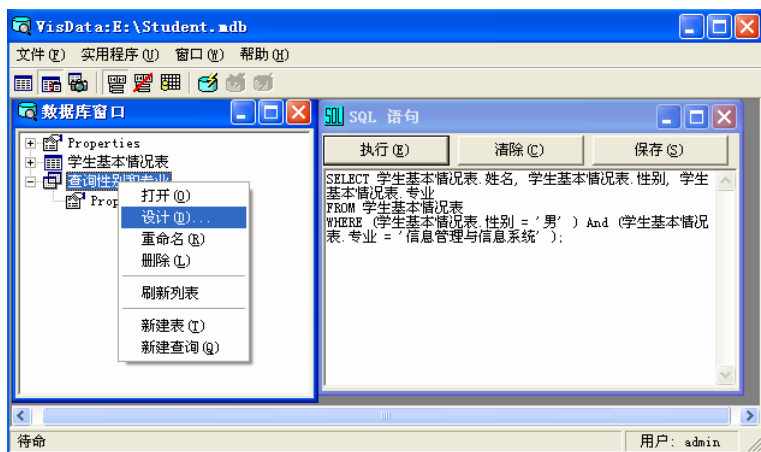


图 11.15 SQL 语句窗口

11.2.7 数据窗体设计器

数据窗体设计器可以根据数据库中建立的表或查询快速生成一个窗体，并添加到工程中。使用数据窗体设计器建立数据库操作窗体的步骤如下。

(1) 新建一个“标准 EXE”工程。

(2) 在可视化数据管理器中，从【实用程序】菜单中选择【数据窗体设计器】选项，打开【数据窗体设计器】对话框，如图 11.16 所示。

在【窗体名称(为带扩展名)】框中输入要添加到 VB 工程中的窗体的名称。从【记录源】列表选择一个存在的查询或表，或者输入一个新的 SQL 语句。记录源是用于窗体或报表的基本数据源(表、查询或 SQL 语句)。

【可用字段】框中列出了选中记录源的所有字段。【包括字段】框中列出了要在窗体出现的字段。在【可用字段】框中选择出现在窗体上的字段，单击【>】按钮将该字段移到【包括字段】框中。单击【<】按钮可以进行相反的操作。如果是全部选中或者全部移除，则可以使用【>>】或【<<】按钮。单击对话框右侧的上下箭头按钮，可以重新排列字段在窗体中出现的次序。

(3) 单击【生成窗体】按钮，可以在工程中添加一个数据窗体。

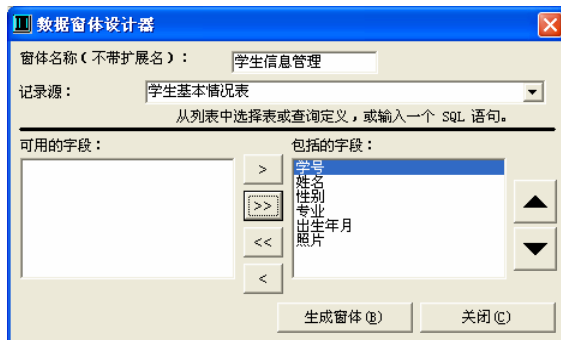


图 11.16 【数据窗体设计器】对话框

11.3 数据控件和数据绑定控件

数据控件(Data)是 Visual Basic 访问数据库的一种利器，添加数据控件并对其某些属性进行设置后，就可对数据库进行操作。但是，它只是负责数据库和工程之间的数据交换，允许将数据从一个记录移动到另一个记录，它本身并不能独立显示数据。如果要显示数据库的访问结果，或往数据库中加入新的数据，还需要文本框或标签等控件。当这些控件和数据控件捆绑在一起时称为数据绑定控件。常用的数据绑定控件有文本框(TextBox)、标签(Label)、复选框(CheckBox)、图像框(Image)、图片框(PictureBox)、列表框(ListBox)和组合框(ComboBox)等。数据绑定控件必须与数据控件在同一窗体中。

11.3.1 数据控件

数据控件(Data)是 Visual Basic 的标准控件之一,可直接从工具箱中加入窗体。它的某些属性必须在【属性】窗体中设置。其他属性既可在【属性】窗体中设置,也可在程序运行时用代码来设置或改变。表 11-7、表 11-8、表 11-9 分别列出了 Data 数据控件的常用属性、方法和事件。

表 11-7 数据控件的常用属性

属 性	描 述
Connect	指定数据控件所要连接的数据库类型
DatabaseName	指定被访问数据库的名称和所在路径
RecordSource	设置或返回数据库中所要访问的数据库表名或查询名
RecordsetType	设置记录集的类型: Dynaset、Snapshot 或 Table 类型
ReadOnly	决定数据库是否可以编辑(True, 可以; False, 不可以)
Exclusive	表示当前用户是否对所访问的数据库独自享用(True, 单用户; False, 多用户)

表 11-8 数据控件的常用方法

方 法	描 述
Refresh	用来建立或重新显示与数据控件相连接的数据库记录集
UpdateRecord	将数据绑定控件中的当前内容写入到数据库表中

表 11-9 数据控件的常用事件

事 件	描 述
Error	主要用来处理不能被任何应用程序捕获的错误
Reposition	当用户单击数据控件上某个箭头按钮, 或者在代码中使用了某个 Move 或 Find 方法使某条新记录成为当前记录时, 将激发 Reposition 事件
Validate	在一条不同的记录成为当前记录之前, 在 Update 方法之前(用 UpdateRecord 方法保存数据时除外)以及 Delete、Unload 或 Close 操作之前会发生该事件

11.3.2 Recordset 对象的属性和方法

在 Visual Basic 中数据库表是不能直接被访问的, Visual Basic 6.0 通过 Microsoft Jet 3.51 数据库引擎(Access 97 以前的版本, 从 Access 2000 开始数据库引擎改为 Microsoft Jet 4.0)提供的 Recordset(记录集)对象来检索和显示数据库记录。一个 Recordset 对象表示一个或多个数据库表中的对象集合的多个对象, 或运行一次查询所得到的记录结果。一个记录集对象相当于一个变量, 与数据库表相似, 记录集也是由行和列组成的, 但不同的是记录集可以同时包含多个表中的数据。表 11-10、表 11-11 分别列出了 Recordset 对象的常用属性和方法。

表 11-10 Recordset 对象的常用属性

属 性	描 述
ActiveConnection	返回 Recordset 对象所属的 Connection 对象
Source	返回或设置 Recordset 对象的生成方式
RecordCount	返回 Recordset 对象中的记录个数
BOF 和 EOF	指示记录指针是否指向了第一条记录之前和最后一条记录之后
AbsolutePosition	返回当前记录的序号
Bookmark	返回或设置当前记录集指针的书签
CursorType	设置或返回 Recordset 对象使用的光标类型
Filter	设置 Recordset 对象中的数据筛选条件
Sort	设置排序字段

表 11-11 Recordset 对象的常用方法

方 法	描 述
Open	打开代表数据表、查询结果等 Recordset 对象中记录的光标
Move	移动记录的指针到指定位置
MoveFirst	使记录集中的第一条记录成为当前记录
MoveLast	使记录集中的最后一条记录成为当前记录
MoveNext	下移一条记录，使下一条记录成为当前记录
MovePrevious	上移一条记录，使上一条记录成为当前记录
AddNew	为可更新的 Recordset 对象添加一条空记录
Requery	重新执行生成的 Recordset 对象的查询，更新其中数据
Update	保存对当前记录的更改
CancelUpdate	取消在调用 Update 方法之前对记录的更改
Delete	删除当前记录或记录组

11.3.3 数据绑定控件

数据绑定控件是用于访问数据库信息的数据识别控件。为创建一个简单的数据库应用程序，使文本框等控件与数据控件捆绑在一起，成为数据控件的绑定控件。创建一个 Data 控件后，通过设定相应的属性，可把数据绑定控件与 Data 控件联系起来。绑定控件的常用属性如表 11-12 所示。

表 11-12 数据绑定控件的常用属性

属 性	描 述
DataChanged	显示于绑定控件里的数据是否被修改
DataSource	用来设置与数据绑定控件捆绑在一起的数据控件
DataField	设置绑定控件所连接数据库中的当前字段的名称

【例 11.1】通过手工方式建立一个简单的学生信息管理界面，如图 11.17 所示。



图 11.17 一个简单的学生信息管理界面

设计步骤如下。

(1) 建立应用程序用户界面与设置对象属性。

单击【新建工程】命令，进入窗体设计器。首先在窗体上增加一个数据控件 Data1，并将其 Align 属性改为：2-Align Bottom，使之位于窗体的下端。然后在窗体上增加 1 个标签控件数组 lblLabels、1 个文本控件数组 txtFields 和 4 个命令按钮，按照表 11-13 进行属性的设置。

表 11-13 属性设置

对 象	属 性	属 性 值	说 明
Data1	Align	2-Align Bottom	设置位置
	DataBaseName	E:\Student.mdb	与数据源连接
	RecordSource	学生基本情况表	与数据表连接
LblLabels(0)~ LblLabels(5)	Caption	学号、姓名、性别、专业、出生年月、照片	
txtFields	DataSource	Data1	设置数据源
	DataField	学号、姓名、性别、专业、出生年月、照片	分别绑定到表中的字段
Command1	Name	cmdAdd	
Command2	Name	cmdDelete	
Command3	Name	cmdRefresh	
Command4	Name	cmdUpdate	
Command5	Name	cmdClose	

(2) 编写代码。

```
Private Sub cmdAdd_Click()  
    Data1.Recordset.AddNew  
End Sub  
Private Sub cmdDelete_Click()  
    Data1.Recordset.Delete  
    Data1.Recordset.MoveNext  
End Sub  
Private Sub cmdRefresh_Click()  
    Data1.Refresh  
End Sub  
Private Sub cmdUpdate_Click()  
    Data1.UpdateRecord  
    Data1.Recordset.Bookmark = Data1.Recordset.LastModified  
End Sub  
Private Sub cmdClose_Click()  
    Unload Me  
End Sub  
Private Sub Data1_Error(DataErr As Integer, Response As Integer)  
    MsgBox "数据错误事件命中错误: " & Error$(DataErr)  
    Response = 0          '忽略错误  
End Sub  
Private Sub Data1_Reposition()  
    Screen.MousePointer = vbDefault  
    On Error Resume Next  
    Data1.Caption = "记录: " & (Data1.Recordset.AbsolutePosition + 1)  
End Sub
```

11.4 使用 ADO 访问数据

ADO(ActiveX Data Object, 数据访问接口)是 Microsoft 公司提出的数据库访问策略,它采用了被称为 OLE DB 的数据访问模式,是数据访问对象 DAO、远程数据对象 RDO 和开放数据库互连 ODBC 这 3 种方式的扩展。Visual Basic 6.0 可以很好地支持 ADO 和 OLE DB 数据访问模式。用户可以使用 ADO 快速建立数据库连接,并通过它方便地操作数据库。

11.4.1 ADO 对象模型

ADO 对象模型定义了一个可以编程的分层对象集合,主要由 3 个对象成员 Connection、Command 和 Recordset 对象,以及几个集合对象 Errors、Parameters 和 Fields 等组成。图 11.18 示意了这些对象彼此之间的关系,表 11-14 是这些对象分工的描述。

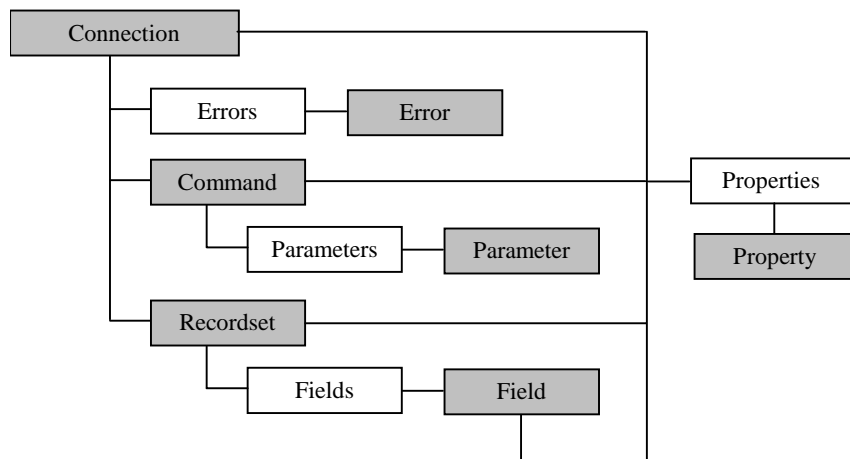


图 11.18 ADO 对象模型

表 11-14 ADO 对象的分工描述

对象名	描述
Connection	连接数据源
Command	从数据源获取所需数据的命令信息
Recordset	所获取的一组记录组成的记录集
Error	在访问数据时，由数据源返回的错误信息
Parameter	与命令对象相关的参数
Field	包含了记录集中某个字段的信息
Property	包含由提供者定义的 ADO 对象的动态特征

要想在程序中使用 ADO 对象，必须先为当前工程引用 ADO 的对象库，引用方式是执行【工程】菜单的【引用】命令，启动【引用】对话框，在列表中选择 Microsoft ActiveX Data Object 2.0 Library 选项。

11.4.2 ADODC 控件

在使用 ADODC 数据控件前，必须先通过【工程】→【部件】菜单选择 Microsoft ADO Data Control 6.0(OLE DB)选项，将 ADO 数据控件添加到工具箱。ADO 数据控件与 Visual Basic 的内部数据控件很相似，它允许使用 ADO 数据控件的基本属性快速地创建与数据库的连接。

1. ADODC 控件的基本属性、方法和事件

ADODC 控件的基本属性如表 11-15 所示，它的方法和事件与 Data 控件对应的方法和事件基本一致，此处不再介绍。ADODC 的 ConnectionString 属性可以带 4 个参数，其参数说明如表 11-16 所示。

表 11-15 ADODC 控件的基本属性

属 性	描 述
ConnectionString	连接数据源，它包含了用于与数据源建立连接的相关信息
RecordSource	设置或返回数据库中所要访问的数据库表名或查询名
ConnectionTimeout	设置数据连接的超时
MaxRecords	设置一个查询中最多能返回的记录数

表 11-16 ConnectionString 属性的参数说明

参 数	说 明
Provider	指定用于连接的数据源名称
File Name	指定基于数据源的文件名
Remote Provider	指定在远程数据服务器打开一个客户端时所用的数据源名称
Remote Server	指定在远程数据服务器打开一个服务器端时所用的数据源名称

2. 设置 ADO 数据控件的属性

下面通过使用 ADO 数据控件连接“Student.mdb”数据库来说明 ADODC 控件属性的设置过程：

(1) 在窗体上添加 ADO 数据控件，控件名采用默认名“Adodcl”。

(2) 右击 ADO 控件，在弹出的快捷菜单中选择【ADODC 属性】选项，将打开如图 11.19 所示的【属性页】对话框。

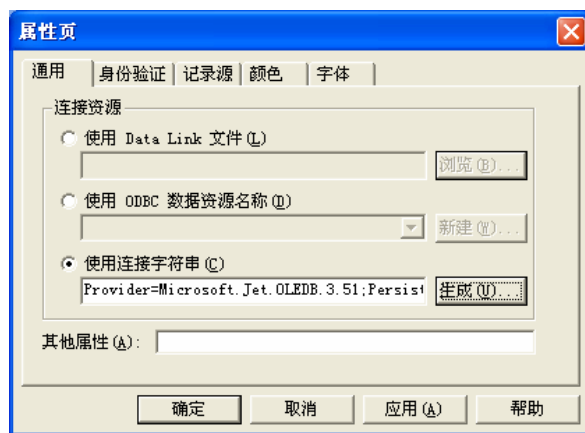


图 11.19 【属性页】对话框

(3) 在【通用】选项卡中选择【使用连接字符串】选项，单击【生成】按钮，打开如图 11.20 所示的【数据链接属性】对话框，在其中可以设置 ADO 控件的窗口中的 ConnectionString 属性。在【数据链接属性】对话框的【提供程序】选项卡中，选择适当的 OLE DB 的提供者(本例选择 Microsoft Jet 3.51 OLE DB Provider 项)，单击【下一步】按钮，进入【数据链接属性】对话框的【连接】选项卡，如图 11.21 所示。在其中选择需要使用

的数据库文件(本例选择 Student.mdb)，然后单击【测试连接】按钮，验证连接的正确性。最后单击【确定】按钮完成 Connectionstring 属性的设置。



图 11.20 【数据链接属性】对话框

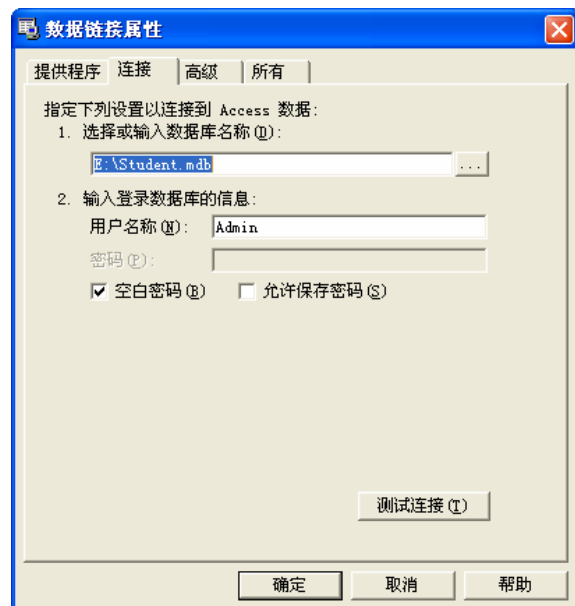


图 11.21 【连接】选项卡

(4) 打开 ADO 控件属性窗口中的【记录源】选项卡，设置记录源的命令类型为 2(表类型)，设置表或存储过程的名称为“学生基本情况表”，如图 11.22 所示。关闭记录源属性页。此时，已完成了 ADO 数据控件的连接工作。

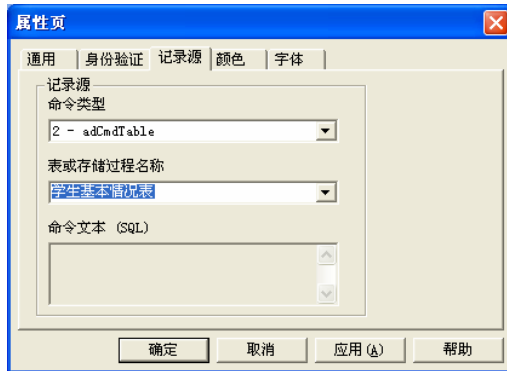


图 11.22 【记录源】选项卡

11.5 应用示例

【例 11.2】 使用 ADODC 控件建立数据查看窗体，完成例 11-1 实现的功能。

设计步骤如下。

(1) 建立应用程序用户界面与设置对象属性。单击【新建工程】命令，进入窗体设计器。首先在窗体上增加一个 ADODC 控件 Adodc1，并将其 Align 属性改为: 2-vbAlignBottom，使之位于窗体的下端。然后在窗体上增加 1 个标签控件数组 lblLabels、1 个文本控件数组 txtFields 和 4 个命令按钮，按照表 11-17 进行相关属性的设置。

(2) 按照 11.4 小节中“设置 ADO 数据控件的属性”提供的步骤，对 Adodc1 控件进行属性设置，使其连接上数据库和数据表。

表 11-17 属性设置

对 象	属 性	属 性 值	说 明
Adodc1	ConnectionString	Provider=Microsoft.Jet.OLEDB.3.51; Persist Security Info=False; Data Source=E:\Student.mdb	与数据源连接
	RecordSource	学生基本情况表	与数据表连接
LblLabels(0)~ LblLabels(5)	Caption	学号、姓名、性别、专业、出生年月、照片	
txtFields	DataSource	Adodc1	设置数据源
	DataField	学号、姓名、性别、专业、出生年月、照片	分别绑定到表中的字段
Command1	Name	cmdAdd	
Command2	Name	cmdDelete	
Command3	Name	cmdRefresh	
Command4	Name	cmdUpdate	
Command5	Name	cmdClose	

(3) 编写程序。

```
Private Sub Adodc1_MoveComplete(ByVal adReason As ADODB.EventReasonEnum,
ByVal pError As ADODB.Error, adStatus As ADODB.EventStatusEnum, ByVal
pRecordset As ADODB.Recordset)
    On Error Resume Next
    Adodc1.Caption = "记录: " & Adodc1.Recordset.AbsolutePosition
End Sub
Private Sub cmdAdd_Click()
    Adodc1.Recordset.AddNew
End Sub
Private Sub cmdDelete_Click()
    Adodc1.Recordset.Delete
    Adodc1.Recordset.MoveNext
End Sub
Private Sub cmdRefresh_Click()
    Adodc1.Refresh
End Sub
Private Sub cmdUpdate_Click()
    Adodc1.Recordset.Update
    Adodc1.Recordset.MoveFirst
End Sub
Private Sub cmdClose_Click()
    Unload Me
End Sub
```

【例 11.3】 利用 ADO 对象编程模型建立数据查看窗体(图 11.23)。



图 11.23 使用 ADO 对象编程模型

设计步骤如下。

(1) 建立应用程序用户界面与设置对象属性。单击【新建工程】命令，进入窗体设计器。在窗体上增加 1 个标签控件数组 lblLabels、1 个文本控件数组 txtFields 和 1 个命令按钮控件数组 Command1。

(2) 将数组 txtFields 的 DataField 属性依次设置为：学号、姓名、性别、专业、出生年月。其他控件的属性设置参见图 11.23。

(3) 编写代码。

首先在窗体的通用声明段声明对象变量：

```
Private DBS As New ADODB.Connection
Private RST As New ADODB.Recordset
```

然后在窗体的 Load 事件中建立 ADO 的 Connection 对象和 Recordset 对象：

```
Private Sub Form_Load()
With DBS
    If .State = adStateOpen Then
        .Close
    End If
    .Provider = "Microsoft.Jet.OLEDB.4.0"
'Access 2000 及以上版本创建 MDB 数据库
    .Provider = "Microsoft.Jet.OLEDB.3.51"
'Access 97 及以下版本创建 MDB 数据库
    .ConnectionString = "E:\Student.mdb"
'可以和公用对话框结合起来使用
    .Open
End With
With RST
    If .State = adStateOpen Then
        .Close
    End If
    .LockType = adLockOptimistic
    .ActiveConnection = DBS
    .CursorLocation = adUseClient
    .CursorType = adOpenKeyset
    .Open "学生基本情况表", Options:=adCmdTable
End With
For each aa in txtfields '绑定所有文本框到记录源
    Set aa.DataSource = RST
Next
End Sub
```

最后编写命令按钮控件数组 Command1()的 Click 事件代码：

```
Private Sub Command1_Click(Index As Integer)
    n = Index
    Select Case n
        Case 0
            RST.MoveFirst
        Case 1
            RST.MovePrevious
            If RST.BOF Then RST.MoveFirst
        Case 2
            RST.MoveNext
            If RST.EOF Then RST.MoveLast
        Case 3
            RST.MoveLast
    End Select
End Sub
```

11.6 小 结

本章介绍了数据库的基本概念和在 Visual Basic 中的数据库操作。

数据库(Database)是指一组排列成易于处理或读取的相关数据的有序集合,它是由一个或多个表对象组成的集合。按数据组织形式可以将数据库分为层次型、网状型和关系型结构,其中最常用的是关系型数据库。关系数据库是二维表的集合,SQL 是最重要的关系数据库操作语言。

Visual Basic 中可以访问的数据库类型有 dBase、FoxPro、Access、MS SQL Server、Oracle SQL Server 和 Sybase SQL Server 等。

Visual Basic 所支持的不同类型的数据库可以通过相关的数据库管理系统来建立。可视化数据管理器(VisData)是 Visual Basic 的数据库设计工具,使用这个工具不但能够创建多种格式的数据库,而且可对它们进行浏览和其他操作。

Visual Basic 提供的数据库访问方法主要有:使用数据(Data)控件或 ADO 数据控件访问数据库,通过 ODBC 方式访问远程数据库以及采用 ADO 对象模型访问数据库等。

11.7 习 题

一、填空题

1. 按数据组织形式可以将数据库分为:_____、_____和_____。
2. 关系型数据库模型是把数据库用_____来表示。
3. 数据控件的 Connect 属性的默认属性是连接_____数据库。
4. VB 中记录集有_____、_____、_____三种类型。
5. 数据控件通过它的 3 个基本属性_____、_____、_____设置要访问的数据资源。
6. 常用的数据绑定控件有:_____、_____、_____、_____、_____和_____等。

二、上机操作题

1. 使用可视化的数据库管理器建立一个 Access 数据库 Mydb.mdb,包含两张表。一张是学生基本信息表(Student),表结构如表 11-18 所示,一张是课程表(Class),表结构如表 11-19 所示。

表 11-18 学生基本信息表结构

字 段 名	类 型	宽 度	字 段 名	类 型	宽 度
学号	Text	12	专业	Text	20
姓名	Text	10	出生年月	Date	8
性别	Logical		家庭住址	Text	30
照片	Binary		备注	Memo	

表 11-19 学生课程表结构

字 段 名	类 型	宽 度	字 段 名	类 型	宽 度
学号	Text	12	成绩	Simple	
课程名	Text	20	考核方式	Text	6
教师	Text	12	学期	Int	

当数据库建立后，使用数据库管理器在各表中输入若干条记录。

2. 设计 2 个窗体，通过文本框、标签、图像框等数据绑定控件分别显示 Student 或 Class 表内的记录，显示界面自定。对数据控件属性进行设置，使之可以对记录集直接进行增加、修改操作。(分别使用 Data 控件、ADODC 控件和 ADO 对象模型等 3 种方式来完成。)

3. 设计一个窗体，通过使用数据控件和数据网格控件浏览 Student 表内的记录。

参 考 答 案

第 1 章

一、选择题

- | | | | |
|------|------|------|------|
| 1. C | 2. B | 3. B | 4. C |
| 5. B | 6. B | 7. D | 8. A |

二、填空题

1. .vbp .frm .bas
2. 文件(File) 退出(Exit)
3. Click
4. 工程资源管理
5. 窗体布局

三、编程题(略)

第 2 章

一、选择题

- | | | | |
|------|------|------|--------|
| 1. A | 2. A | 3. D | 4. C |
| 5. B | 6. D | 7. A | 8. B C |

二、填空题

- | | |
|--------------------------|-------------------|
| 1. 289 | 2. 双引号 # |
| 3. ABCDEFGHI bcdefghi | 4. BCD123 |

第 3 章

一、选择题

- | | | | | |
|------|------|------|------|------|
| 1. C | 2. C | 3. B | 4. C | 5. B |
|------|------|------|------|------|

二、填空题

1. Rnd() 100 x mod 3 x Next I
2. InputBox("输入 a 值: ") InputBox("输入 b 值: ") InputBox("输入 c 值: ")
- T = a: a = b: b = T a < c a c

三、编程题(略)

第 4 章

一、选择题

- | | | | | | |
|-------|-------|-------|-------|-------|-------|
| 1. C | 2. B | 3. C | 4. D | 5. C | 6. B |
| 7. B | 8. D | 9. C | 10. D | 11. D | 12. D |
| 13. A | 14. B | 15. A | | | |

二、填空题

1. 文本框 列表框
2. Clear
3. Value
4. 文本框中输入的字符数没有限制
5. (1) List1.AddItem str(i)
- (2) List1.Count-1
- (3) Val(List1.List(i))
6. (1) Text1.SelText
- (2) Text1.SelText
- (3) Text1.SelText = " "
- (4) Text1.Text = Clipboard.GetText
- (5) Text1.SelText = " "
- (6) " "

三、编程题(略)

第 5 章

一、选择题

- | | | | |
|------|------|------|------|
| 1. C | 2. C | 3. C | 4. C |
| 5. B | 6. C | 7. B | 8. A |

二、填空题

1. Name Index
2. 变体类型 一
3. (1) arr1(1)
(2) min=arr1(i)
4. (1) i+j-1
(2) 7-i
(3) print
5. (1) val(inputbox("请输入一个数据"))
(2) k=i
(3) k=j
(4) a(i)=a(k)

三、编程题(略)

第 6 章

一、选择题

- | | | | |
|------|------|------|------|
| 1. A | 2. C | 3. B | 4. D |
| 5. D | 6. A | 7. A | 8. A |

二、填空题

1. 9
2. 30 30 10
3. 30 70
4. 21
41

三、编程题(略)

第 7 章

一、选择题

- | | | | | |
|------|------|------|------|---------------|
| 1. A | 2. C | 3. D | 4. C | 5. D |
| 6. A | 7. D | 8. B | 9. C | 10. (1)C (2)A |

二、填空题

1. ASC II 码 键代码
2. 按下鼠标拖动后释放鼠标或者双击鼠标
3. MousePointer 99 MouseIcon
4. DragMode 1 0
5. CTRL 键 ALT 键 CTRL 键+ ALT 键
6. 分三行输出 F65、G66 和 H67
7. DragOver DragDrop
8. DragIcon
9. A a

三、编程题(略)

第 8 章

一、选择题

- | | | | |
|------|------|------|------|
| 1. D | 2. D | 3. D | 4. A |
| 5. A | 6. D | 7. A | |

二、填空题

MouseDown PopupMenu

三、编程题(略)

第 9 章

一、选择题

- | | | | | | |
|------|------|------|------|------|------|
| 1. D | 2. D | 3. C | 4. D | 5. C | 6. A |
|------|------|------|------|------|------|

二、阅读程序，写出运行结果

1. 以当前窗体中心为圆心，半径随机的 100 个不超出窗体范围的同心圆
2. 窗体中显示一条正弦曲线和一条余弦曲线

三、编程题(略)

第 10 章

一、填空题

1. 系统文件 隐藏文件 只读文件 普通文件和存档文件
2. Print 和 Write
3. 打开 关闭
4. 驱动器列表框 目录列表框 文件列表框
5. Microsoft Scripting Runtime
6. MoveFile Move

二、选择题

1. B 2. C 3. C 4. A

三、简答题(略)

四、编程题(略)

第 11 章

一、填空题

1. 层次型 网状型 关系型
2. 表的集合
3. Access
4. Dynaset Snapshot Table
5. Connect DatabaseName RecordSource
6. 文本框(TextBox) 标签(Label) 复选框(CheckBox) 图像框(Image)
- 图片框(PictureBox) 列表框(ListBox) 组合框(ComboBox)

二、上机操作题(略)

参 考 文 献

- 1 郑阿奇. Visual Basic 实用教程(第 2 版). 北京: 电子工业出版社, 2004
- 2 刘炳文. Visual Basic 程序设计. 北京: 机械工业出版社, 2004
- 3 教育部考试中心. 全国计算机等级考试二级教程: Visual Basic 语言程序设计(修订版). 北京: 高等教育出版社, 2003
- 4 谭浩强, 袁玫, 薛淑斌. Visual Basic 程序设计(第 2 版). 北京: 清华大学出版社, 2004
- 5 钱雪忠等. 新编 Visual Basic 程序设计实用教程. 北京: 机械工业出版社, 2004
- 6 张瑾, 刘开南. 计算机技术基础(上册). 北京: 中国矿业大学出版社, 2005
- 7 刘瑞新, 王远征. Visual Basic 程序设计教程. 北京: 机械工业出版社, 2003
- 8 李振亭. Visual Basic 程序设计教程. 北京: 北方交通大学出版社, 2001
- 9 谢步瀛, 龚沛曾. Visual Basic 计算机绘图实用技术. 北京: 电子工业出版社, 2004
- 10 蒋金丹. Visual Basic 数据库应用. 北京: 科学出版社, 2004