<<Visual Basic 2005程序设 >

图书基本信息

书名：<<Visual Basic 2005程序设计教程>>

13位ISBN编号：9787302194767

10位ISBN编号：7302194769

出版时间：2009-2

出版时间：清华大学出版社

作者：郭兴峰，廖建军，周明辉 编著

页数：349

前言

Microsoft Visual Basic 2005是从Visual Basic语言演变而来的，它是一种类型安全和面向对象的语言。Visual Basic允许开发人员开发面向Windows、Web和移动设备的程序。
与所有基于Microsoft .NET Framework的语言一样，使用Visual Basic编写的程序都具有安全性和语言互操作性方面的优点。

本书面向Visual Basic初中级用户，全面、系统地介绍了使用Visual Basic 2005开发应用程序的基础知识、基本方法和具体应用。
当然，如果您是一名高手，那么这本书也将是一本极好的参考书。

本书由浅入深，层层递进地讲解了使用Visual Basic 2005开发应用程序的技术。
本书共分为14章，包括Visual Studio 2005开发环境，Visual Basic语法，面向对象基本概念的介绍，Windows窗体和常用控件，菜单、工具栏和状态栏，鼠标和键盘事件，对话框，文件操作等技术的讲解，以及使用Visual Basic开发数据库程序、Web网站、ActiveX的应用、应用程序的调试和错误处理等高级内容，最后介绍了应用程序的安装和部署。

第1章对Visual Basic 2005进行了简要介绍。
首先介绍了.NET Framework，然后介绍了Visual Studio 2005的开发环境，最后通过一个实例介绍了如何使用Visual Basic开发一个Windows应用程序。

第2章讲解了程序设计基础。
首先介绍了数据类型和表达式，然后介绍了程序结构——顺序结构、分支结构、循环结构，最后介绍了子过程和函数。

第3章讲解了面向对象的程序设计的基本思想。
首先讲解了面向对象程序设计的基本概念，然后讲解了类和对象，最后讲解了命名空间和Me关键字。

第4章讲解了Windows窗体。
首先介绍了窗体的属性、方法和事件，然后讲解了多文档和单文档的概念，最后通过一个例子展示了多文档程序的开发过程。

第5章讲解了Windows常用控件。
主要包括Label、TextBox、Button、CheckBox、Radio Button、ListBox、Combobox、ListView、TreeView、TabControl等内容。
通过对这些控件的学习，读者可以达到举一反三的效果。

第6章讲解了菜单、工具栏和状态栏。
首先介绍了主菜单和弹出式菜单，然后介绍了工具栏的使用，最后介绍了状态栏的设计和使用。

第7章讲解了鼠标和键盘事件。
首先介绍了鼠标的常见事件，然后介绍了键盘事件以及检测Shift、Ctrl及Alt键的状态，最后介绍了如何中断后台处理。

第8章讲解了对话框。
首先介绍了通用对话框，包括颜色对话框、字体对话框、打开文件对话框、保存文件对话框以及浏览目录对话框，然后介绍了预定义对话框，最后介绍了自定义对话框。

第9章介绍了文件操作。
首先介绍了文件的基本概念，然后介绍了如何创建文件以及对文件进行删除和读写操作，最后介绍了如何操作文件夹。

第10章介绍了如何开发数据库程序。
首先介绍了ADO.NET的基本概念。
然后介绍了.NET数据提供程序的Connection对象、Command对象、DataReader对象、DataAdapter对象。
最后介绍了DataSet组件的结构和用法。

第11章介绍了Web应用程序开发。
首先介绍了如何建立Web应用程序的运行环境，然后介绍了Visual Basic .NET在开发Web应用程序时的

作用，最后介绍了Visual Basic .NET开发Web程序的常用类和常用控件。

第12章介绍了如何使用ActiveX部件。

主要包括使用ActiveX部件的步骤、释放ActiveX部件、处理ActiveX部件的运行错误和挂起的请求。最后通过使用多媒体控件MMControl详细地讲解了使用ActiveX部件的步骤。

第13章介绍了应用程序的调试和错误处理。

首先介绍了应用程序的调试，然后介绍了如何对应用程序的错误进行处理。

第14章介绍了应用程序的安装和部署。

程序员工作的最终目的是把开发的程序交付给用户使用，这部分内容讲解了如何对Windows应用程序和Web程序进行打包。

为了运行这些代码，读者需要配置一个运行环境，需要安装Visual Basic 2005中文企业版，详细的配置请参考本书具体章节的介绍。

本书由廖建军、周明辉、王艳梅编写，参与本书编写和修改的还有叶明、崔宁、卢宏、汪昔玉、卫平峰、程冬丁、王勤、张锐、汪小锋、李葵、叶浩、肖飞、宋海剑、林勇、朱衡等人员。

在此，编者对他们致以诚挚的谢意！

由于本书涉及的范围比较广泛，作者的经验有限，时间仓促，书中难免有不足之处，敬请广大读者、专家提出宝贵意见。

内容概要

Visual Basic 2005是微软推出的一种功能强大的开发语言，它具有类型安全和完全面向对象的特点。Visual Basic可以用于程序开发的许多领域，如桌面程序、数据库开发、网站开发等，它上手容易、功能强大，越来越受到广大程序员和编程爱好者的青睐。

本书共分14章，系统地介绍了如何使用Visual Basic 2005开发应用程序，具体包括Visual Studio 2005开发环境、Visual Basic的语法、面向对象的概念、Windows窗体的使用、控件的使用、对话框、文件操作、数据库开发、网站开发、使用ActiveX部件、应用程序的调试和错误处理以及应用程序的安装和部署等内容。

本书重点明确，结构合理，语言简明，书中实例均为作者在本领域工作中的真实案例，具有很强的实用性。

本书可作为高等学校计算机相关专业的教材，也可作为Visual Basic初、中级用户的自学用书。

<<Visual Basic 2005程序设 >

书籍目录

章节摘录

第13章 应用程序的调试和错误处理 Visual Basic 2005有着非常丰富的调试工具，读者必须在不断的使用中熟练地掌握它们，这对于编写和开发稍大的应用程序非常重要。
在编写大型应用程序时，会显得更加至关重要。
在发布应用程序给其他用户之前，测试应用程序并尽可能地改正错误是非常重要的。
本章介绍如何使用Visual Basic.NET的调试工具，同时介绍几个技巧将使应用程序出错的可能性降低到最小，并在出错时尽快找出它们。
这些技巧包括：用详细的注释归档代码，使用调试工具等。
无论多么仔细精巧地制作代码，都可能（而且很可能）会出现错误。
例如在运行中，可能会遇到意外而中止程序，或者由于程序的本身缺陷，导致错误。
好的应用程序应该包含错误处理代码，在出现错误的时候，能够对错误进行处理，使程序能够继续运行或安全地退出，减少错误带来的麻烦。
本章在最后将介绍如何在程序中使用错误处理语句处理错误。
本章结束的时候，相信读者已经能够有效地归档、调试程序代码，使程序具有很强的生命力。
本章重点内容： * 归档程序代码 * 熟练使用各种调试工具 * 使用断点中断程序的执行 * 结构化错误处理 13.1 归档程序代码 在深入研究如何使用Visual Basic 2005的调试工具之前，先看一看预防性的措施。
虽然它的效果不一定能明显地表现出来，但清晰、简明、一致的文档是良好代码的基本条件。
技巧： 归档代码时，最重要的是保持一致性。
这意味着使用同样的注释、命名、代码布局样式。
如果学会采用归档代码的方法来积极有效地开发应用程序，将缩短花在调试上的时间。
13.1.1 给程序代码增加注释 用注释作为程序代码文档几乎像程序代码本身一样重要，注释可以用来标识一行代码、一个过程、一个代码模块甚至整个程序段。
如果已经完成了以前的例子，那么就已经看到并使用了注释。
最明显的注释符是单引号。
只要单引号被插入到某一行中，单引号之后的该行中的内容都是注释，并且注释以不同的颜色显示出来（默认的颜色为绿色）。
如何使用注释依赖于读者的风格和习惯，下面将介绍一些有效的归档程序代码的方法。
1. 应用程序级 在开发大的项目时，编程人员被分成若干组去开发项目的不同部分，每个小组使用单独的生成系统（Build System），这个生成系统是一个安装所有第三方控件并且具有各种Visual Basic版本的中央计算机。
组件的需求就显得尤为重要，因为安装应用程序时常常无法知道哪些资源是必需的。
为了解决这个问题，创建一个单独的模块。
这个模块除了具有对项目的注释外，不含其他的信息，它使用如下格式： 应用程序名称 版本 版权 服务器 客户端 作者/公司 用途 最低需求 当上面的注释区被填满后，注释应该看起来如下所示： 应用程序名称 Kill Happy Time Virsus 版本 1.0 版权 2001-免费软件 作者/公司 张三/红软科技公司 用途 杀毒软件 最低需求 CPU：主频1GB以上，内存：512MB以上 客户端 Windows 9x /NT4.0 服务器 MS SQL2000 这个应用程序级的注释文件包含在应用程序中；当其他人检查这个应用程序时，通过这段描述文件就可以知道要想使用这个应用程序正确地编译和运行需要哪些资源。
2. 模块级 描述每个模块的目的比归档应用程序的特殊细节和需要更重要的多，这一点在多个开发人员同时使用一个包含需要重复使用的特定函数时尤为明显。
下面是推荐使用的在模块级注释的代码，它能让要使用该模块的其他编程人员很快了解所有必要的信息，缩短开发时间，并能开发出一致的、标准的应用程序。
文件名 作者 日期 描述 依赖库 更新 下面是填入注释的例子：

文件名 MyBrowser 作者 张三 日期 2001-4-25 描述 这是一个动态链接库，包含了创建浏览器所需的方法 依赖库 不需要 更新 2001-4-25创建 2001-5-6 添加刷新函数 技巧： 如果要生成可复用代码、类模块，可以在模块的全局定义中加入注释，以说明整个模块中的过程、需求以及这个模块的独立性，保证其他编程人员确切地了解如何在它们的应用程序中使用这些代码。

3. 过程级 在用户定义的或不明显的过程及函数的开头加入注释是十分有用的，这些注释将帮助读者或其他编程人员确切地理解并记住过程的功能、用法。

例如： 过程: AddUser（） 作者 张三 日期 2001-5-6 描述 这个函数的作用是在域中添加一个全局用户 需求 为了这个函数正常工作，必须首先提供用户ID和密码 提示： 在编写类或标准模块时，可在模块的全局定义中加入注释，用来说明这个模块中的属性、方法和事件以及这个模块的独立性，保证其他编程人员确切地了解如何在它们的程序中使用这些代码。

4. 代码级 注释可以深入到任何一级，可以直接将注释放到代码行中。

当某行代码十分值得注意时，在这行后面加入注释是非常重要的。

在使用注释符时，可以根据自己的习惯对注释符进行扩展，以代表更多的含义。

表13-1是推荐使用的几种注释符。

表13-1 推荐使用的注释符 注释符说明 一般的注释。

它可以用在那些不需特别注意的地方 ?? 有问题的代码，编辑代码的时候，总有一些地方觉得需要改进或存在问题，标出来引起注意，这对于程序的维护和调试是非常有益的 !! 需要引起注意的代码 最基本的注释符是单引号。

无论将这个符号放在一行代码的中间还是字符串的后面，单引号右面的文本都被注释掉了。

运用这一点，就可以发现一些灵活而有用的注释技巧，就像上面列出的那样。

尽管这些技巧是非常有用的，但不是绝对的。

可以创建自己的注释符集合，但是，一定注意，集合的内容不要太多，否则难于记忆，而且要保持所有应用程序一致性，这对于节省调试时间是绝对重要的。

13.1.2 使用命名规则 除了在代码中写注释以外，还应该使用统一的命名方式，即命名规则来为组件命名。

命名规则有助于代码更可读、更明了。

下面介绍Visual Basic中使用的命名规则。

命名Visual Basic应用程序中的元素时，名称的首字符必须为字母字符、数字或下划线。

但是，注意，以下划线开头的名称不符合公共语言规范（CLS）。

以下建议适用于命名： * 名称中各单词首字母均为大写，如FindLastRecord和RedrawMyForm。

* 函数名和方法名以动词开始，如InitNameArray和CloseDialog。

* 类名、结构名、模块名和属性名以名词开始，如EmployeeName或CarAccessory中。

* 接口名称以前缀I开始，后面接一个名词或名词词组（如IComponent），或者接一个描述接口行为的形容词（如IPersistable）。

不要使用下划线，不要过多使用缩写，因为缩写会引起混淆。

* 事件处理程序的名称以一个描述事件类型的名词开始，后面接后缀EventHandler，如MouseEventHandler。

* 事件参数类的名称里要加EventArgs后缀。

* 如果某事件含有“之前”或“之后”的概念，则以现在时或过去时形式使用前缀，如ControlAdd或ControlAdded。

* 对于长项或常用项，可使用缩写使名称长度适中。

例如，可以使用HTML代替HyperText Markup Language。

通常，多于32个字符的变量名在低分辨率的监视器上难以阅读。

同时，应确保缩写在整个应用程序中保持一致。

在项目中随意在HTML和HyperText Markup Language之间切换可能会导致混淆。
* 在内部范围中避免使用与外部范围中的名称相同的名称。
如果访问了错误的变量，则可能会产生错误结果。
若变量与同一名称的关键字冲突，则必须在关键字前加适当的类型库以作标识。
例如，如果有一个名为Date的变量，通过调用 System.DateTime.Date只可以使用内部 Date 函数。
* 常量名最好全部大写，尽量使用全名，使用下划线连接不同单词，如USER_ID。
* 控件和窗体命名使用前缀。
13.1.3 编写结构化的代码 除了注释和命名规则外，另一个归档代码的方法是编写结构化的代码，即使用制表符和空格使代码在代码窗口中以缩进形式排列。
在Visual Basic 2005中默认的制表符是4个空格，但是可以通过IDE中的“ 工具 ”|“ 选项 ”命令，打开“ 选项 ”对话框，选择“ 文本编辑器 ”|Basic|“ 编辑器 ”，来设置制表符的空格个数和缩进量，如图13-1所示。
图13-1 设置制表符 Visual Basic 2005的代码编辑器已经具有了自动缩进功能（可以禁止这些功能），主要是对于过程级代码和控制语句，如下面的代码： Public Sub Button1_Click（ByVal sender As Object, ByVal e As System.EventArgs） Dim a As Integer = 0 If a > 5 Then a = 0 Else a = a + 1 End If End Sub 所有的缩进是由程序自动完成的，不需要自己手动调节。
读者所要做的是设置这些缩进量、在控制语句内部条件不同层次语句之间的缩进量以及添加清晰的注释。
13.2 Visual Studio 2005调试器的新增功能 和原来版本的调试技术相比，Visual Studio 2005增加了许多新的功能。
下面简单介绍一些重要的新增功能。
* Visual Basic的“ 编辑并继续 ”功能：可以在调试应用程序时更改VB代码，同时还可以继续运行应用程序。
此功能让我们能够迅速修复错误、测试新功能和修改现有功能，从而提高工作效率。
* 远程调试更为安全、设置更为简便：通过将单个可执行程序复制到远程计算机来设置远程调试，而无需使用复杂的设置指令或注册。
远程调试现在更为安全和可靠。
此外，现在可以调试64位的托管应用程序和非托管应用程序。
* 增强的调试器数据提示功能：调试器的“ 数据提示 ”得到了改进。
可以在源代码编辑器中直接定位复杂数据结构的内容，还可以从“ 数据提示 ”中打开可视化工具，以直观和自然的格式查看数据。
*“ 仅我的代码 ”调试：此功能使我们可以将注意力集中在自己编写的代码上，而忽略自己不感兴趣的代码。
* 跟踪点和改进的断点用户界面：断点不再仅用于进行中断。
跟踪点是一种使用断点执行自定义操作的新方法。
使用跟踪点，可以输出消息或运行Visual Studio自动化宏，决定当调试器命中跟踪点时是中断还是继续。
用户界面得到了改进，使得设置所有断点更为轻松快捷。
* Visual Basic异常助手：新增的“ 异常助手 ”对话框能够在Visual Basic程序中发生异常时提供更好的信息。
除了上面提到的功能外，Visual Studio 2005还在许多方面进行了改进，这里就不再一一介绍了。
如果使用过早期的Visual Studio软件，就会发现和早期的版本相比，Visual Studio 2005带来了更多便利。
13.3 调 试 工 具 .NET Framework SDK包含名为Visual Debugger的工具，我们可以在运行应用程序时使用该工具来调试代码。
此工具位于%ProgramFiles%\Microsoft Visual Studio 8\SDK\v2.0\GuiDebug\DbgCLR.exe中。
在使用该调试器时，可以通过在执行应用程序时逐句执行每条语句，以及通过查看每个变量中的数据

来准确了解应用程序的工作方式。

使用Visual Debugger可以在运行代码时对代码进行检查，该程序中包含下列可以帮助调试应用程序的功能： * 断点：断点是代码中调试器将要停止应用程序的位置。

可以使用断点查看应用程序的当前数据状态，然后逐句通过每一行代码。

* 单步执行：当在断点处停止后，即可逐行运行代码（称为单步执行代码）。

* 数据查看：Visual Debugger为在运行应用程序时查看和跟踪数据提供了许多不同的选项。

调试器允许开发人员在以中断模式停止应用程序时修改数据，然后使用修改过的数据继续运行应用程序。

注意： Visual Studio?2005的“ 编辑并继续 ”功能对于 Web 应用程序不可用。

调试程序时，需要注意权限的问题。

调试进程比执行该进程需要更多的特权。

因此，除了要为调试配置应用程序外，还必须确保有足够的权限附加到进程，以便调试该进程。

用户有权调试在他们自己的用户本地标识下运行的进程，但不能调试其他用户的进程。

管理员可以调试任何进程。

下面介绍如何使用调试工具。

13.3.1 调试工具栏 在Debug菜单中有所有的调试命令和工具，为了提供更方便的快速访问与使用这些命令和工具，Visual Basic 2005提供了调试工具栏，如图13-2所示。

图13-2 调试工具栏 下面按照从左到右的顺序介绍这些按钮。

* 启动/继续按钮：用来启动当前的工程，进入调试状态。

除了使用这个按钮，还可以使用“ 调试 ”|“ 启动调试 ”命令或者直接按F5键。

说明： 几乎所有的调试工具都有快捷键。

例如，启动/继续是F5键，停止调试是Ctrl+Shift +Break键等，记住快捷键可以提高调试的效率,而且有时候按按钮不方便时（如全屏运行），只能使用快捷键。

* 全部中断：调试器将停止所有在调试器下运行的程序。

程序并不退出，可以随时恢复执行。

调试器和应用程序现在处于中断模式。

* 停止调试：如果程序是从Visual Studio启动的，则“ 停止调试 ”终止正调试的进程。

如果程序附加到进程，而不是从Visual Studio启动，则该进程仍继续运行。

* 逐语句按钮：使用逐语句按钮可以每次只执行一行代码。

如果该行包含函数调用，则仅执行调用本身，然后在函数内的第一个代码行处停止。

它的快捷键是F8。

* 逐过程按钮：每次执行一行代码，如果该行包含函数调用，则执行整个函数，然后在函数外的第一行处停止，快捷键是Shift+F8。

* 跳出按钮：在函数内部单击该按钮，一直执行代码，直到函数返回，然后在调用函数中的返回点处中断。

快捷键是Ctrl+Shift+F8。

13.3.2 使用调试工具窗口 有时可以通过运行部分代码来查找问题产生的原因。

但是，经常要做的往往还是分析数据到底发生了什么变化。

可以在有关变量或属性的问题中将不正确的值放到一边，然后确定变量或属性是如何得到不正确的值的，为什么会得到这些值。

在逐步运行应用程序的语句时，可用调试窗口监视表达式和变量的值。

常用的调试窗口有即时窗口、监视窗口、输出窗口、局部变量窗口、调用堆栈窗口。

下面进行介绍。

1. 即时窗口 即时窗口显示代码中正在调试的语句所产生的信息，也可以显示直接往窗口中输入的命令所请求的信息。

即时窗口如图13-3所示。

图13-3 即时窗口 有时，当调试或试验一个应用程序时，可能要执行单个过程、对表达式求

值或者为变量或属性赋予新的值。
可用即时窗口完成这些任务。
可以通过在即时窗口中显示表达式来计算表达式的值。
对于监视表达式，这有多个优点：用户可以得到反馈信息，以获得关于应用程序执行的情况，而不必中断执行。
在运行应用程序时，可看到显示出的数据或其他信息。
由于在一个单独的区域内（即时窗口）显示反馈，所以反馈不会影响用户看到的输出。
又因为可以把这个代码作为窗体的一部分来保存，所以下次处理应用程序时不必重新定义这些语句。
在即时窗口计算表达式或变量很简单，使用问号（?）命令直接把信息显示到即时窗口中。
可在立即窗口中计算任何有效的表达式，包括调用了属性的表达式。
当前活动窗体或模块决定了表达式的范围。
如果在与窗体或类相关联的代码内中止了执行，则可像下面代码这样引用那个窗体（或其控件之一）的属性： ? Button1.Enabled 2. 监视窗口 监视窗口是最常使用的代码调试窗口，如图13-4所示。
监视窗口在运行时监视和观察对象的变量，可以在监视窗口中看到多个属性，单击监视窗口的按钮就可以看到监视窗口。
监视窗口动态地监视变量的更新。
图13-4 监视窗口 3. 局部变量窗口 局部变量窗口显示当前堆栈的所有变量及其值。
局部变量窗口中的值在每次由运行模式到中断模式时或堆栈内容发生改变时会自动更新。
局部变量窗口不仅显示变量的值，而且也显示对象的情况。
局部变量窗口如图13-5所示。
图13-5 局部变量窗口 4. 输出窗口 在调试或运行程序时最先看到的是输出窗口，它显示程序的编译和运行情况，如图13-6所示。
除此以外，输出窗口可以显示程序中正在调试的语句所产生的信息，即使用Debug对象的Write、WriteLine方法时，产生的调试输出结果。
图13-6 输出窗口 5. 调用堆栈窗口 调用堆栈窗口里面显示了已经装入但没有完成的过程，如图13-7所示。
该图显示了Form1窗体的Button1控件的Click事件被装入，而没有运行完成。
有时该窗体也显示其他的一些被装入而未运行的过程。
图13-7 调用堆栈窗口 13.4 断 点 断点就是程序中定义的一个位置，在那里程序可被暂时停止执行，控制权交给调试器。
程序暂停期间，可以检查和修改不同的程序参数，并可以通过单步执行代码或从当前语句继续执行等操作来往下执行。
一般来说，断点可以告诉调试器，在断点之前的所有指令都是正确的，调试器不需要浪费时间去单步执行它们。
Visual Basic中可以设置多种类型的断点，常用的包括位置断点、条件断点、命中次数断点，后面将依次介绍这些断点。
13.4.1 位置断点 位置断点主要是由其位置发挥作用的，即当程序运行到设立断点的地方时程序将会停下来。
位置断点是最简单的设置断点的方式，只要把光标移到要设断点的位置，当然这一行必须包含一条有效的语句，然后从“调试”菜单中选择“切换断点”命令或按F9键或在上下文菜单中选择“断点”|“插入断点”命令，就会在屏幕上看到在这一行的左边出现一个红色的圆点，表示这行设置了一个断点，如图13-8所示。
图13-8 位置断点 若要删除一个断点，只需把光标移到设置断点的代码行处，然后从“调试”菜单中选择“切换断点”命令或按F9键，就会在屏幕上看到在这一行左边出现的红色的圆点消失，表示这行的断点已经删除。
13.4.2 条件断点 条件断点是位置断点的扩展方式，当程序执行到带标记的指令时，如果指定

的条件是真，调试器就会响应条件断点。
在同一指令执行数百次的循环中，设置在该循环中的位置断点在每次重复过程中都会停止执行，这是我们所不希望的。
条件断点仅在某些情况发生时中断指令，例如当循环次数达到100时。
设置条件断点的方法为：首先设置好位置断点，然后在位置断点上单击鼠标右键，在弹出的快捷菜单中选择“条件”命令，弹出如图13-9所示的对话框。
图13-9 设置条件断点 在“条件”文本框中输入中断时的判断条件，可以选择在满足条件时中断，也可以在条件改变时中断。
13.4.3 命中次数断点 默认情况下，每次命中断点时执行都中断。
使用命中次数断点，可以选择： * 总是中断（默认设置）。
* 命中次数等于指定值时中断。
* 命中次数等于指定值的倍数时中断。
* 命中次数大于或等于指定值时中断。
如果要跟踪断点的命中次数但不中断执行，可以将命中次数设置为一个很高的值以便永不命中断点。
需要注意的是，仅为调试会话期间保留指定的命中次数。
在调试会话结束时，命中次数将重置为零。
设置命中次数断点的方法为：首先设置好位置断点，然后在位置断点上单击鼠标右键，在弹出的快捷菜单中选择“命中次数”命令，弹出如图13-10所示的对话框。
图13-10 命中次数断点 如果不选择“命中断点时总是中断”，则下拉列表框中会出现一个文本框，用于让用户输入断点命中多少次后中断。
下面通过一个例子进一步了解如何使用断点。
步骤如下： （1） 创建一个名为chap13的解决方案，然后在该解决方案中添加一个名为DebugTest的项目。
（2） 在该项目默认创建的窗体Form1上添加一个Button控件。
双击该控件，创建该控件的Click事件，在该事件中添加如下代码： Private Sub Button1_Click（ByVal sender As System.Object, ByVal e As System.EventArgs） Handles Button1.Click For i As Integer = 0 To 100 Label1.Text = i.ToString（） Next End Sub 这段代码十分简单，执行的功能为循环100次，每次把i的值显示到Label控件上。
我们主要是通过这段代码来演示如何设置断点。
（3） 把光标移到Label1.Text = i.ToString（）所在的行，按F9键，设置断点，如图13-11所示。
图13-11 设置位置断点 （4） 在断点处单击鼠标右键，从弹出的快捷菜单中选择“命令次数”，在“断点命中次数”对话框的“命中断点时下拉列表框中选择“中断，条件是命中次数等于”，然后在下拉列表框右侧的文本框中输入50，如图13-12所示。
图13-12 设置命中次数 （5） 单击“确定”按钮关闭该对话框，然后按F5键开始运行程序。
单击Button1按钮，发现程序运行到i=49时中断。
因为i是从0开始执行的，所以i等于49时正好运行了50次。
此时可以使用前面介绍的调试工具窗口来查看变量的信息，这里就不再介绍了。
读者可以自己使用调试工具窗口进行查看。
13.5 结构化错误处理 就理想的情况而言，Visual Basic过程根本不需要错误处理代码。
遗憾的是，我们并没有生活在一个理想世界里。
有时文件会被误删除、磁盘驱动器空间会溢出、网络驱动器会意外地分离。
这些可能发生的事情都能在代码中引起运行时错误。
为了处理这些错误，需要将错误处理代码添加到过程中。
有时，错误也可能出现在代码内部，通常称这类错误为缺陷。
例如，像光标不听指挥之类的小缺陷就足以令人感到沮丧并深感不便。
更严重的缺陷可能还会使应用程序不再对命令做出响应，这时，可能要重新启动应用程序，从而使已

经完成但尚未存储的工作丢失。
因而在程序中编写错误处理代码是非常重要的。
13.5.1 On Error语句 Visual Basic 6中已经使用了On Error Goto语句来捕获和处理错误，在Visual Basic 2005中该功能被保留下来。
下面简单介绍它的使用，Visual Basic 6的高手可以略过这段内容。
使用一个错误处理的例子来说明如何使用On Error语句。
步骤如下： （1） 在该解决方案chap13中添加一个名为OnErrorTest的项目。
（2） 修改该项目的默认窗体Form1的Text属性为演示OnError，并在Form1中增加一个Timer控件，设置其Enabled属性为True，创建该控件的Tick事件，并在该事件中增加如下代码： Private Sub Timer1_Tick（ByVal sender As System.Object, ByVal e As System.EventArgs） Handles Timer1.Tick Dim msg As Integer Dim counter As Integer 如果检测到任何一个错误，则开始捕获错误，并响应错误处理程序 Label1.Text = "我们邂逅了：" & counter.ToString & "秒！" On Error GoTo check counter = Integer.MaxValue counter += 1 Exit Sub 语句标号，以下是错误处理语句 check: Timer1.Stop（） 从消息框中选择重试还是放弃 msg = MsgBox（"Error Find ", Microsoft.VisualBasic.MsgBoxStyle.RetryCancel, "Error"） 如果选择重试，返回到出错的行，重新执行 If msg = 4 Then Resume 如果放弃则退出应用程序 Else End End If End Sub 执行该程序，程序开始执行后，很快弹出错误对话框，如图13-13所示。
如果单击“ 重试 ”按钮，则继续弹出该对话框。
如果单击“ 取消 ”按钮，则关闭该程序。
现在简单解释一下错误处理语句。
* On Error语句：用来捕获程序中的错误。
* Goto语句：跳转语句，一般用来调转到错误处理程序段。
* Resume语句：重新启动，当错误处理完成后，重新回到出错的语句，继续向下执行。
上面代码中，定时器启动时每隔0.1秒执行Tick事件，因为把counter赋为最大整数，然后再对counter加1会产生溢出错误。
因此会进入On Error语句中，程序执行直接跳转到check处。
在check处，程序首先弹出“ 错误 ”对话框，如果单击“ 重试 ”按钮，则程序继续从发生错误处执行，因为该错误还存在，所以会继续弹出“ 错误 ”对话框。
如果单击“ 取消 ”按钮，则End语句被调用，该程序自动退出。
13.5.2 使用Try…Catch…Finally…End Try块 使用On Error错误处理代码的最大缺点是不灵活，不稳定，不符合结构化的程序设计（有Goto语句）。
在Visual Basic 2005中增加了新的结构化错误处理语句，来代替这些错误处理语句，它们是Try…Catch…Finally…End Try。
通过这些语句可以集中处理错误。
下面介绍它的使用。
Try…Catch…Finally…End Try语句的语法如下： Try statements Catch statements Catch statements … Finally statements End Try 将可能出错的语句放在Try后面，由Try语句创建错误句柄；由Catch语句获得错误，并执行Catch后面的错误处理语句，进行错误处理。
程序中可以有多个Catch语句，如果第一个Catch捕获不到，将由后面的Catch继续捕获错误。
无论有无错误，程序都将执行Finally后面的语句。
下面举一个简单的例子说明如何在程序中使用该结构捕获和处理错误。
这段程序的作用是当单击Button1按钮时，在TextBox中显示x整除y的结果。
程序中加入了错误处理语句，步骤如下： （1） 在解决方案chap13中增加一个名为TryCatchTest的项目。
（2） 在该项目的默认窗体Form1中添加一个Button控件、三个Label控件、三个TextBox控件，按照图13-14所示设置控件的Text属性和位置。

（3） 双击“ 计算 ”按钮，创建该控件的Click事件。
在该事件中添加如下代码： Private Sub Button1_Click（ByVal sender As System.Object, ByVal e As System.EventArgs） Handles Button1.Click Dim x As Integer x = Integer.Parse（TextBox1.Text） Dim y As Integer y = Integer.Parse（TextBox2.Text） 创建结构化的错误句柄 Try 除出错 x = x \ y TextBox3.Text = x.ToString 捕获错误 Catch err As Exception 错误处理，将错误转化为字符串显示 MsgBox（err.ToString） If y = 0 Then y = 1 跳出错误处理过程 Finally Beep（） End Try End Sub （4） 运行该程序，被除数输入1，除数输入0，单击“ 计算 ”按钮，弹出如图13-15所示的对话框。
图13-15 捕获异常 程序中有一个漏洞：没有注意y是否未零。
如果y为零，除零必然会产生错误，因而程序中将Try语句放在“ x=x\y ”语句之前，进行捕获错误。
当发生错误时，由Catch捕捉，并将错误原因用消息框显示出来，然后跳出错误处理程序。
错误对象详细地指明了错误的原因、出错的位置，当开发人员看到捕获的错误时，将很快地根据错误提示改正程序中的错误。
使用Try...Catch...Finally...End Try结构可以方便地捕获和处理错误，大大增加了程序的可靠性和应变能力。
如果希望在调试运行过程中将程序中捕获的错误显示在即时窗口中，可以使用Visual Basic 2005的Debug对象的Write和WriteLine方法来实现。
Debug.WriteLine将一行文字输出到即时窗口；Debug.Write 方法将文字输出到即时窗口，并且没有换行符。
将上个例子中的MsgBox（…）语句替换为： Debug.Write（err.ToString） 这时按F5键运行，可以在即时窗口看到捕获的错误以及原因，如图13-16所示。
图13-16 即时窗口错误信息 使用Debug对象可以方便调试，但在正式发布的工程中一定要去掉所有Debug语句和前面的MsgBox消息框，而应该使应用程序根据可能的错误进行处理，使程序可以继续运行或者能够安全退出，使用户的损失最小。
13.6 习 题 13.6.1 填空题 1. 调试程序时，需要注意权限的问题。
比执行该进程需要更多的特权。
因此，除了要为调试配置应用程序外，还必须确保有足够的 附加到进程，以便调试该进程。
2. 在逐步运行应用程序的语句时，可用调试窗口监视表达式和变量的值。
常用的调试窗口有 、 、 、 、 。
3. 在即时窗口计算表达式或变量很简单，使用 命令直接把信息显示到即时窗口中。
4. 断点就是程序中定义的一个位置，在那里程序可被暂时停止执行，控制权交给 。
程序暂停期间，可以检查和修改不同的 ，并可以通过 代码或从当前语句 等操作来往下执行。
5. 如果希望在调试运行过程中将程序中捕获的错误显示在即时窗口中，可以使用Visual Basic 2005的Debug对象的 和 方法来实现。
13.6.2 选择题 1. 使用异常时，将可能出错的语句放在 后面，由 语句创建错误句柄；由 语句获得错误，并执行 后面的错误处理语句，进行错误处理。
无论有无错误，程序都将执行 后面的语句。
A. Try B. Catch C. Finally D. On Error 2. 窗口显示当前堆栈的所有变量及其
窗口中值在每次由运行模式到中断模式时或堆栈内容发生改变时会自动更新。
A. 局部变量 B. 即时 C. 监视 D. 调用堆栈 3. 在调试或运行程序时最先看到的是窗口，它显示程序的编译和运行情况。
除此以外， 窗口可以显示程序中正在调试的语句所产生的信息。
A. 局部变量 B. 即时 C. 监视 D. 输出 4. 断点是 断点的扩展方式，当程
行到带标记的指令时，如果指定的条件是真，调试器就会响应 断点。
A. 条件 B. 命中次数 C. 输出 D. 位置 5. Visual Basic中可以设置多种类型的断点，常用的包括 断点、 断点、 断点。

A. 条件 B. 命中次数 C. 输出 D. 位置 13.6.3 问答题 1. 简述如何使用即时窗口试程序。

2. 简述如何使用断点调试程序。

3. 简述On Error语句和Try…Catch…Finally…End Try块的区别。

13.6.4 上机操作题 1. 使用Try…Catch…Finally…End Try块改写OnErrorTest项目，弹出的异常如图13-17所示。

图13-17 错误对话框 2. 修改DebugTest项目，使用条件断点，使得当i等于50时中断程序。

书摘插图 第1章 Visual Basic 2005简介 Microsoft Visual Basic 2005是从Visual Basic语言演变而来，它是一种类型安全和面向对象的程序设计语言。

Visual Basic允许开发人员开发面向Windows、Web和移动设备的程序。

与所有面向Microsoft.NET Framework的语言一样，使用Visual Basic编写的程序都具有安全性和语言互操作性方面的优点。

本章重点内容： .NET Framework的基本概念 Visual Studi0 2005开发环境 程序开发的基本流程 1.1.NET Framework 2.0概述 .NET Framework是一种Windows内部组件，它支持生成和运行下一代应用程序，同时也支持XML Web Services。

.NET Framework具有两个主要组件：公共语言运行库和.NETFramework类库。

公共语言运行库是.NET Framework的基础。

可以将运行库看作一个在执行时管理代码的代理，它提供内存管理、线程管理和远程处理等核心服务，并且还强制实施严格的类型安全。

事实上，代码管理的概念是运行库的基本原则。

以运行库为目标的代码称为托管代码，而不以运行库为目标的代码称为非托管代码。

编辑推荐

本书系统地介绍了使用Visual Basic 2005开发应用程序的方法和技巧，具体内容包括Visual Studio 2005开发环境、Visual Basic语法、面向对象的概念、Windows窗体和控件的使用、对话框和文件操作、数据库开发、网站开发、使用Active×部件、应用程序的调试和错误处理以及应用程序的安装和部署等内容。

读者对象　　本书可作为高等学校计算机相关专业的教材，也可作为Visual Basic初、中级用户的参考书。

本书特色　　针对高校学生和初、中级用户，基础知识与实践相结合，详细介绍Visual Basic 2005程序设计相关知识。

内容全面，实例丰富，可操作性强，切实提高读者的实际编程能力。

对编程实例中的重点步骤给予特别说明，仔细剖析技术要点，加深读者印象。

章前给出重点内容，章后附有针对性的练习，使读者加深对知识点的理解和掌握，并能举一反三。

版权说明