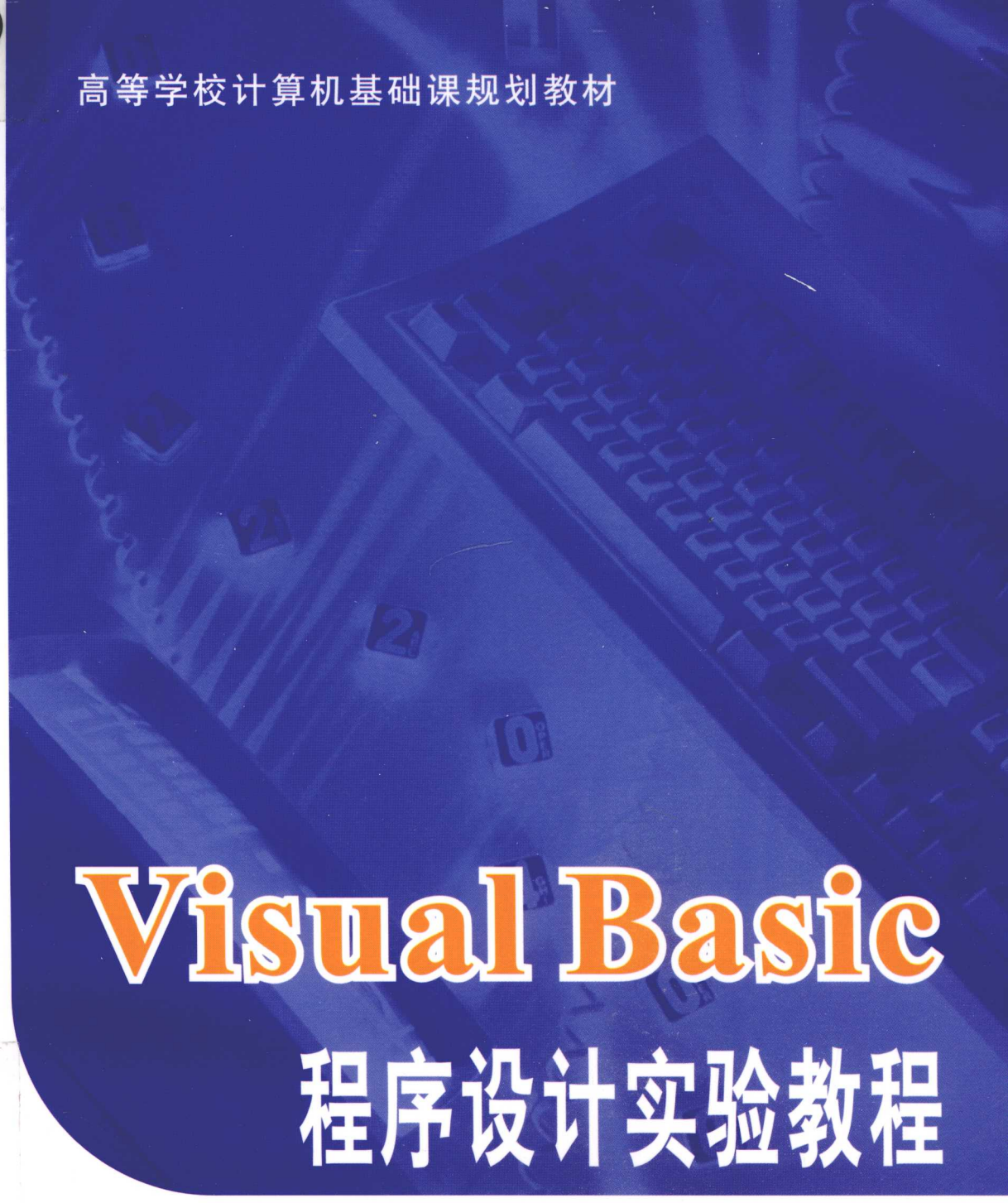
高等学校计算机基础课规划教材



Visual Basic 程序设计实验教程

陈　达　吴长海　主编

科学出版社
www.sciencep.com

高等学校计算机基础课规划教材

Visual Basic 程序设计实验教程

陈　达　吴长海　主编

科学出版社
北京

内 容 简 介

本书是与《Visual Basic 程序设计教程》配套的辅助教材。全书共分 5 个部分：第 1 部分为上机指导，介绍了上机过程遇到的一些问题和上机编写、调试程序的基本方法；第 2 部分为实验内容，编排了 12 个综合实验题目，详细讲述了每一个实验的实验目的、实验手段及实验方法；第 3 部分为习题解答篇，对主教材中各章的习题绘出详细答案；第 4 部分综合本教材学习的内容及要求、并结合全国计算机等级考试的内容形式给出了综合测验题；第 5 部分是上机练习题。书中习题内容解答详细，力求针对性强；实验内容丰富，综合性强，并对各章节的知识点加以适当扩充，实验的应用性相对主教材例题有所提升，有利于学习者知识的掌握和实践能力的提高。

本书可与《Visual Basic 程序设计教程》一书配套使用，也可作为全国计算机等级考试 Visual Basic 程序设计的习题参考用书，还可作为相关的技术培训的教材，以及程序设计初学者的自学用书。

图书在版编目(CIP)数据

Visual Basic 程序设计实验教程/陈达，吴长海主编. —北京：科学出版社，2010.2

高等学校计算机基础课规划教材

ISBN 978-7-03-026634-7

Ⅰ. ①V… Ⅱ. ①陈…②吴… Ⅲ. ①BASIC 语言－程序设计－高等学校－教材 Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字(2010)第 020741 号

责任编辑：张颖兵 梅 莹/责任校对：闫 陶

责任印制：彭 超/封面设计：苏 波

科学出版社 出版

北京东黄城根北街 16 号

邮政编码：100717

http://www.sciencep.com

武汉市新华印刷有限责任公司印刷

科学出版社发行 各地新华书店经销

*

2010 年 1 月第 一 版 开本：787×1092 1/16

2010 年 1 月第一次印刷 印张：14 3/4

印数：1—4 000 字数：364 000

定价：29.00 元

（如有印装质量问题，我社负责调换）

《Visual Basic 程序设计实验教程》编委会

前　言

Visual Basic 是 Microsoft 公司推出的一种面向对象的“可视化”Windows 应用程序开发工具，它在语法上继承了 Basic 和 Quick Basic 的优点，具有使用方便、简单易学等特点，且功能强大，与其他开发工具有丰富的接口。因此，深受广大用户的青睐，成为学习开发 Windows 应用程序首选的程序设计语言。

目前，许多高校非计算机专业都开设了“Visual Basic 程序设计”课程，而很多非计算机专业人员也选择使用 Visual Basic 作为学习计算机程序设计的语言。

“Visual Basic 程序设计”是一门实践性很强的课程，我们根据多年从事计算机程序设计教学实践经验，编写了这本与《Visual Basic 程序设计教程》配套的实验教程。编写过程中按照 Visual Basic 程序设计的特点，采用“任务驱动”方式，精心设计每一个实验实例、实验内容，实践证明，使本书更能激发读者学习 Visual Basic 程序设计的兴趣，培养学生的实际编程能力。

全书包括 Visual Basic 上机指导、基础实验、习题参考解答、综合练习题和上机练习题。其中，Visual Basic 上机指导部分主要介绍 Visual Basic 6.0 集成环境的使用与设置，以及 Visual Basic 程序设计的基本概念；基础实验部分共有 12 个实验，是针对程序设计初学者而设计的，主要包括大学非计算机专业“Visual Basic 程序设计”课程的必修教学实验内容。所有实验均具有较强的针对性和实践性，通过实验使读者掌握 Visual Basic 程序设计与调试方法，巩固所学知识，培养实际编程能力。

习题参考解答部分给出了《Visual Basic 程序设计教程》中各章习题的详细分析及解题思路、方法；综合测试题练习题用于读者复习、巩固所学知识，测验自己是否真正掌握相关知识；上机练习题便于检验读者的实际操作能力水平。

本书可以作为各类高等院校、各类高职院校非计算机专业学生的“Visual Basic 程序设计”课程的实验教学用书，也可作为广大计算机爱好者学习 Visual Basic 程序设计语言的参考书。

由于作者水平所限，书中疏漏之处在所难免，恳请广大读者批评指正。

作　者

2009 年 10 月

目　录

第1部分 上机指导

1.1 应用程序调试

在程序编写的过程中，出现错误是难免的，查找并修改错误的过程称为程序调试。

1.1.1 Visual Basic 的工作模式

在进行程序调试时，必须了解应用程序正处在何种工作模式下。Visual Basic(以下简称 VB)提供了设计模式(design mode)、运行模式(run mode)和中断模式(break mode)三种工作模式。

1. 设计模式

启动 VB，即进入设计模式。此时主窗口标题栏显示为**设计**，一个应用程序对应的所有步骤基本上在设计模式下完成，包括程序的界面设计、属性设置和代码编写等，在设计模式下，不能运行程序，也不能使用调试工具但可以设置断点。

2. 运行模式

执行**运行**菜单中**启动**命令，也可单击工具栏上**启动**按钮或按 F5 功能键，即进入运行模式。标题栏显示**运行**，在该模式下，不能修改程序代码。若要修改代码，可以执行**运行**菜单中**结束**命令，或单击工具栏上**结束**按钮，回到设计模式进行修改；或者执行**运行**菜单中**中断**命令，也可单击工具栏上**中断**按钮或按 Ctrl+Break 组合键，进入中断模式进行修改。

3. 中断模式

执行**运行**菜单中的**中断**命令，或者单击工具栏上**中断**按钮或按 Ctrl+Break 组合键，以及程序出现运行错误时，进入中断模式，标题栏显示**中断**。在此阶段，程序暂停，可以查看并修改代码，检查数据是否正确，修改结束后，可以继续执行程序或中止程序的运行。

在中断模式下，**运行**菜单中**启动**命令变为**继续**命令。

1.1.2 程序错误种类

VB 程序设计中的常见错误可以分为语法错误(syntax error)、编译错误(compile error)、运行错误(run-time error)和逻辑错误(logical error)4 类。

1. 语法错误

语法错误常常是由于拼错一条命令或使用不正确的语法引起的。例如，关键字拼写不正确，遗漏了某些标点符号，英文的标点符号写成中文的标点符号，函数调用缺少参数，括号不匹配，有 For 没 Next，或把 ElseIf 写成了 Else If 等。语法错误通常在键入程序时发生，是最容易

发现的一类错误，因为 VB 具有自动检测语法错误的功能。该功能可以通过**工具**菜单中**选项**命令，在**编辑器**选项卡中设置。当用户在代码窗口中编辑代码时，VB 会及时检查出语法错误并提示用户纠正。例如以下代码：

```
Private Sub command1_Click( )
 a=9
 b=4
 c=a*b
 print  "c=";c
End Sub
```

在输入时，如果第一行输入为

```
a=
```

按回车键后，就会弹出一错误信息框，如图 1.1 所示，同时该语句行变为红色醒目提示。

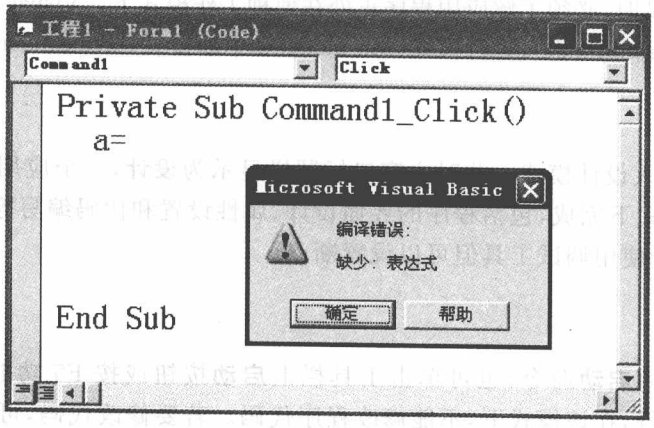


图 1.1　检查语法错误

此时，若想对程序进行修改，可先单击错误信息框的**确定**按钮，或按回车键，关闭该信息框。如果不明白错误信息的含义，可以单击错误信息框中的**帮助**按钮，或按 F1 键，来获取这条错误产生原因及解决办法的帮助信息。若没有安装 MSDN 将会出现提示，如图 1.2 所示。

图 1.2　安装提示

使用 VB 的自动语法检测时，应注意两点：①错误信息框显示的为**编译错误**，实际上也是语法错误；②一般说来，VB 高亮度显示出错单词及语法错误所在的行，但并不能准确给定，需要仔细查找检查。

2. 编译错误

编译错误是指执行**运行**菜单中**启动**命令，或者单击工具栏上**启动**按钮或按 F5 键，以及将程序编译成可执行文件时产生的错误。例如，用户未定义变量、遗漏关键字等。这时，VB 会弹出一个错误信息框，同时出错行高亮显示，如图 1.3 所示。

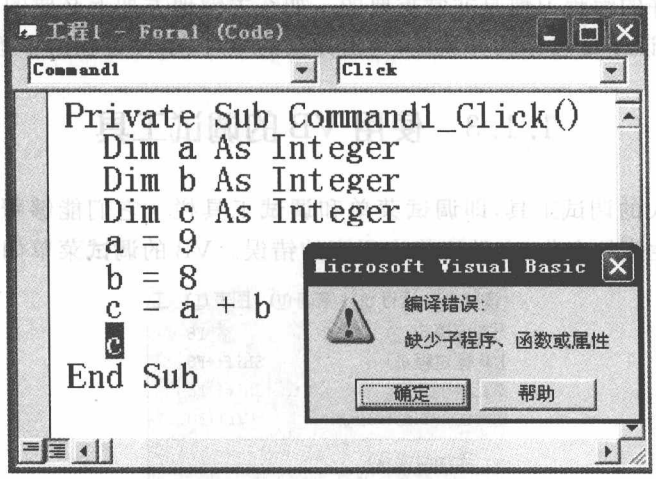


图 1.3　编译错误

这种错误不是语法错误，在输入代码时不会被语法检测发现。出现这类错误后，VB 将停止编译并返回到有错误的代码窗口。

3. 运行错误

运行错误是指程序输入或编译时正确，而在运行代码时发生的错误。这类错误通常是由应用程序在运行期间执行非法操作或某些操作失败所引起的，如类型不匹配、用零作除数、试图打开一个不存在的文件、数组下标越界、磁盘存储空间不足等。例如，变量 a 的类型被定义为单精度型，如果对其赋值的类型是字符串，系统就会发生运行错误，如图 1.4 所示。

图 1.4　运行错误

此时，用户若单击**调试**按钮，则进入中断模式，光标停留在出错行上，可以修改代码；若单击**结束**按钮，则终止程序执行返回到代码窗口；若单击**帮助**按钮，则可以获取有关错误提示的更多信息。这一类错误在设计阶段较难发现，是程序容错检验的重点。

4. 逻辑错误

逻辑错误往往表现为程序运行后得不到设计的预期结果。例如，运算符使用不正确、语句次序不对、循环语句的起始值与终止值不正确等。这类错误一般不产生错误信息，因此是最难查找的一类错误，需要认真分析并借助相应的调试工具才能查出原因并加以改正。为减少逻辑错误的发生，良好的编程习惯是非常重要的。如在关键地方加上必要的注释，强制所有变量必须显式声明以保证变量名称的一致性，列出详细的事件、事件过程清单等。

1.1.3 使用 VB 的调试工具

VB 提供了强大的调试工具，即**调试**菜单和**调试**工具栏。它们能够帮助分析程序运行过程，分析变量和属性值的变化，有助于找出程序的错误。VB 的**调试**菜单如图 1.5 所示。

图 1.5 **调试**菜单

其中，各个命令的作用如下：

(1) 逐语句(step into)，一次执行一条语句；

(2) 逐过程(step over)，一次执行一个过程或语句；

(3) 跳出(step out)：执行当前过程的其他部分，并在调用过程的下一行处中断执行；

(4) 运行到光标处(run to cursor)，通常可以使用这个命令跳过大型循环；

(5) 添加监视(add watch)，该命令可以弹出**添加监视**对话框，可以在该对话框中输入出现在**监视**窗口的监视表达式；

(6) 编辑监视(edit watch)，该命令弹出**编辑监视**对话框，能过此对话框可以编辑或删除监视表达式；

(7) 快速监视(quick watch)，将所选择的表达式值显示在**快速监视**对话框内；

(8) 切换断点(toggle breakpoint)，用于设置或删除当前行上的断点；

(9) 消除所有断点(clear all breakpoints)，消除程序中所有断点；

(10) 设置下一条语句(set next statement)，选定语句作为开始执行；

(11) 显示下一条语句(show next statement),高亮显示下一条将要被执行的语句。

一般说来,**调试**工具栏是隐藏的,选择**视图**菜单中**工具栏**菜单项,在工具栏下的子菜单中执行**调试**命令打开**调试**工具栏,如图 1.6 所示。

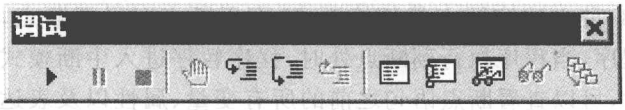


图 1.6 **调试**工具栏

调试工具栏中各个图标所代表的含义从左到右依次为启动、中断、结束、切换断点、逐语句、逐过程、跳出、本地窗口、立即窗口、监视窗口、快速监视和调用堆栈。其中,启动、中断、结束三个按钮控制程序的运行、中断和结束;切换断点、逐语句、逐过程、跳出与调试菜单中相应命令作用相同;本地窗口用于观察当前过程中所有变量的值;立即窗口用于中断状态下显示和改变变量、表达式或对象属性的值;快速监视用于快速查看和选中变量、表达式或对象属性的值;调用堆栈用于显示当前所有活动过程调用的一个列表,该列表中的项目是以分级格式显示的,可以跟踪程序的进程。

1. 进入中断模式

调试工具只有在应用程序的中断模式下才有效,因此要使用调试工具,首先应该进入中断模式。此模式的作用是暂停应用程序的执行并提供有关应用程序的情况,程序总是从运行模式进入中断模式。

进行程序调试时,常用的中断方法有设置断点和使用 Stop 语句两种。

1) 设置断点

进入中断模式最准确最常用的方法是设置断点。通常在需要程序暂停的地方设置断点,断点可以设置多个,用来缩小错误查找范围,对程序进行分段调试。

断点一般在中断模式或设计模式下设置,设置断点的方法如下:

(1) 打开代码窗口,在希望中断的语句行左边的灰色区域(也称边界标识条)单击鼠标,即完成了断点的设置。此时,该语句行以粗体反相显示,如图 1.7 所示。

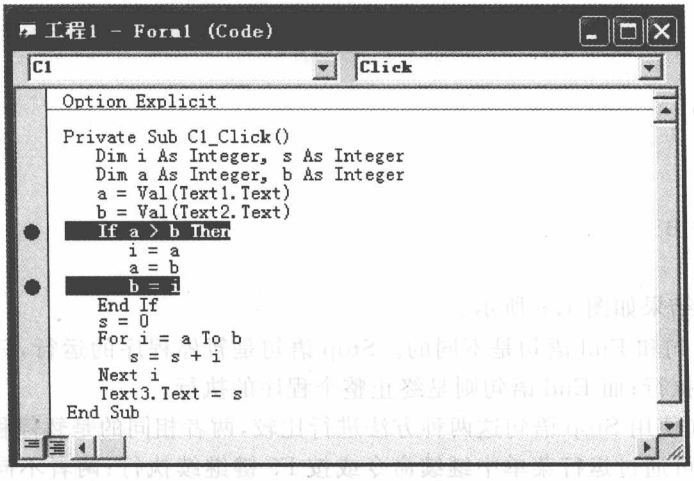


图 1.7 设置断点

(2) 在代码窗口中，将光标移到希望中断的语句上，执行**调试**菜单中**切换断点**命令，即可实现断点的设置。

(3) 单击**调试**工具栏上**切换**按钮或按 Ctrl＋Shift＋F9 组合键，也可设置断点。

上述第(2)、第(3)种方法实际上是切换断点，即无断点时设置断点，有断点时清除断点。设置了断点后运行程序，当程序执行到断点处时，程序暂停进入中断模式，断点语句以黄色高亮度显示出来。此时，可以查看断点语句之前的所有变量、属性以及表达式的值，方法是将鼠标指向所关心的变量处，稍停一下，鼠标下方就会显示该变量的值，如图 1.8 所示。

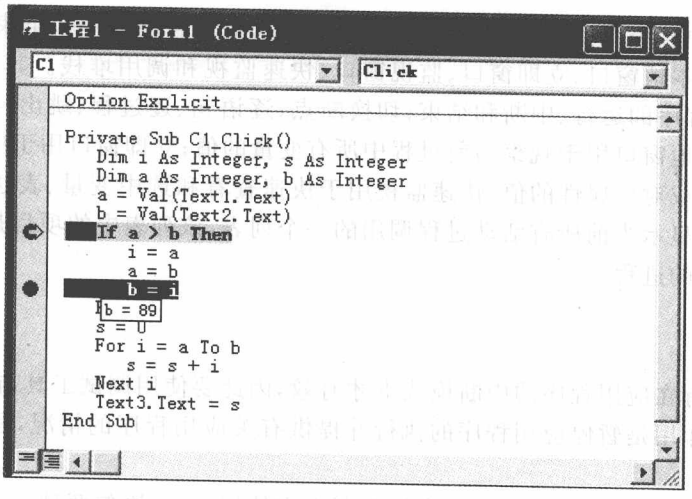


图 1.8　查看变量 b 的值

2) 使用 Stop 语句

进入中断模式的另一种方法是在程序代码中需要暂停的地方加上一个或多个 Stop 语句。当程序遇到 Stop 语句时，就会暂时停止该程序的执行，并进入中断模式以便调试。例如有程序如下：

```
Private Sub command1_Click( )
  a=5
  b=10
  c=a+b
  Stop
  Print "c=";c
  d=a-b
  Stop
  Print "d=";d
  End Sub
```

运行该程序，结果如图 1.9 所示。

注意：Stop 语句和 End 语句是不同的。Stop 语句是暂停程序的运行，只要再选择**继续**命令，程序就可继续执行；而 End 语句则是终止整个程序的执行。

将设置断点和使用 Stop 语句这两种方法进行比较，两者相同的是暂停程序执行并进入中断模式，中断后都可通过**运行**菜单中**继续**命令或按 F5 键继续执行；两者不同的是设置断点较使用 Stop 语句方便，因为无需修改代码，但使用 Stop 语句比设置断点灵活，因为它可以使程序在一定条件下暂停。如程序段：

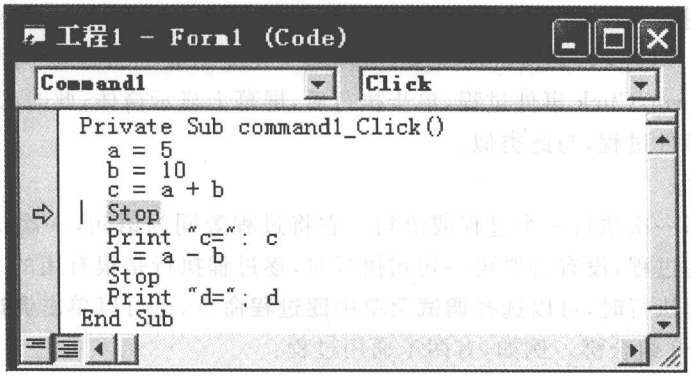


图 1.9　使用 stop 语句

```
If a+b<c Or a+c<b Or b+c<a Then
   Stop
End If
```

2. 程序跟踪

利用断点功能，只能查出错误大致发生的程序段，要确定是哪一条语句发生问题，可以使用程序跟踪方法。VB 调试工具提供逐语句执行、逐过程执行、跳跃执行和运行到光标处 4 种跟踪方式。

1）逐语句执行

逐语句执行，就是每次只执行一条语句，根据输出结果来判断执行的语句是否正确，也称为单步执行。将设置断点和逐语句跟踪相结合，是初学者调试程序最简捷的方式。要进行逐语句执行，可以选择**调试**菜单中**逐语句**命令，也可以单击**调试**工具栏中**逐语句**按钮或按 F8 键。

执行**逐语句**命令后，程序进入运行模式。每执行完一条语句，切换到中断模式并高亮显示下一条语句，如图 1.10 所示。因此，可以知道程序语句的执行顺序及当前的执行位置，从而查找出错误语句。

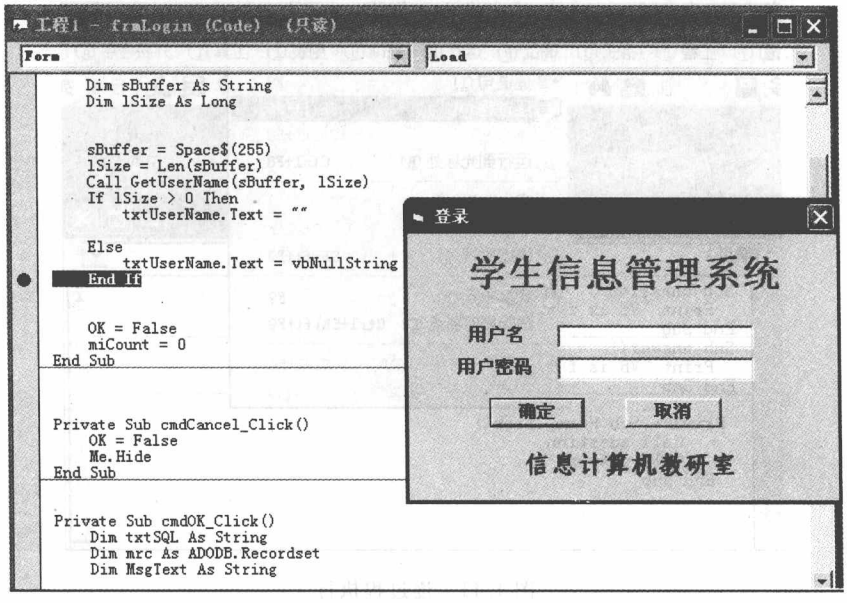


图 1.10　逐语句执行

注意：逐语句执行以语句为单位，不是以行为单位，因此当一行中有多条语句时，每次只有一条语句被执行。另外，如果执行的是事件过程，一定要触发事件才能执行事件过程中的代码。例如，执行 Form_Click 事件过程，单步执行后，屏幕上显示窗体，此时必须单击窗体，才能开始执行。其他事件过程，与此类似。

2）逐过程执行

逐过程执行指一次执行一个过程或语句。它将过程等同为语句，一次执行完毕。当面对一个已经调试过的过程，没有必要再一句句执行时，逐过程执行是很有用的。

要进行逐过程执行时，可以选择**调试**菜单中**逐过程**命令，也可以单击**调试**工具栏中**逐过程**按钮或按 Shift＋F8 组合键。例如，有两个通用过程：

```
Sub question( )
  Print "VB is fun?"
End Sub
Sub answer( )
  Print "VB is fun!"
End Sub
```

在 Form_Click 事件过程中调用上述过程：

```
Private Sub Form_Click( )
    Call question
    Call answer
End Sub
```

执行**逐过程**命令后，屏幕显示窗体，单击窗体后，执行窗体单击事件过程，Private Sub Form_Click()高亮显示；再次启动**逐过程**，Call question 高亮显示，准备执行，如图 1.11 所示，再按一次 Shift＋F8 组合键，窗体上显示 **VB is fun ?**，同时 Call answer 高亮显示，准备执行，继续按 Shift＋F8 组合键，过程执行，窗体上显示 **VB is fun!**，同时 **End Sub** 高亮显示，程序执行结束。

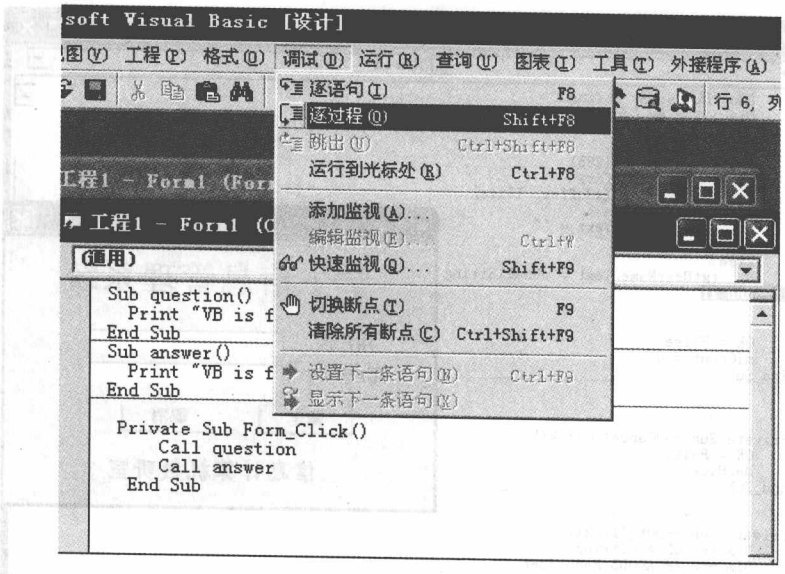


图 1.11　逐过程执行

3）跳跃执行

逐语句执行和逐过程执行都只能按顺序一次执行一条语句或一个过程，如果想暂时避开程序的某一部分调试其他部分，或对程序修改后再回过头来执行，则可以通过跳跃执行来实现。

4）运行到光标处

设计阶段中，在代码窗口中将光标移到某一行上，然后执行**调试**菜单中**运行到光标处**命令，或按 Ctrl+F8 组合键，程序将会在光标所在行停止运行，并在边界标识条中显示相应的标记，如图 1.12 所示。

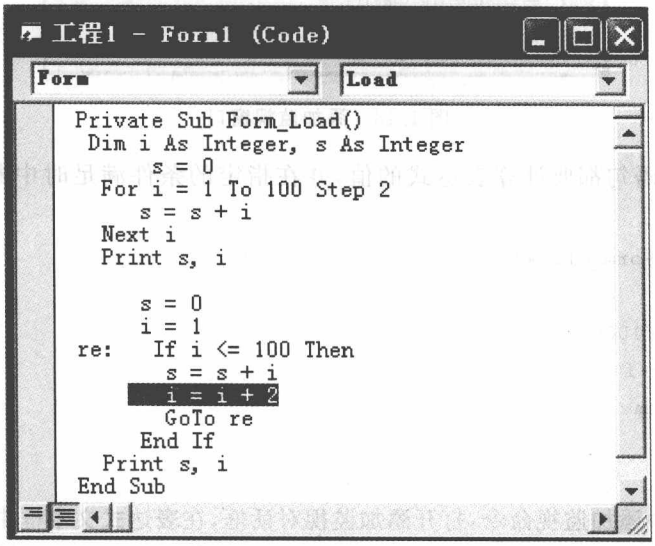


图 1.12　运行到光标处

注意：光标所在行必须在程序的执行流程中，有**运行到光标处**命令可以跳过大型循环。

3. 调试窗口

在中断模式下，除了用光标指向要观察的变量、属性和表达式，直接显示其值外，还可以通过调试窗口来观察变量、属性和表达式的值。VB 主要有**监视**窗口、**立即**窗口和**本地**窗口这三种窗口提供调试，可单击**视图**菜单中相应命令或**调试**工具栏中相应按钮来打开这些窗口。

1）**监视**窗口

在**监视**窗口中只能被动地显示变量或表达式的值。在此之前，必须在设计模式下利用**调试**菜单中**添加监视命令**或**快速监视**命令添加监视表达式以及设置监视类型。程序运行时，**监视**窗口根据所设置的监视类型进行相应显示。执行**调试**菜单中**添加监视**命令后，屏幕上弹出一个对话框，如图 1.13 所示。

该对话框分为**表达式**、**上下文**栏和**监视类型**栏三部分。**表达式**文本框用来输入监视表达式，可以是自述表达式、关系表达式、逻辑表达式等。上下文指定要监视的过程和模块。监视类型部分包括三个单选按钮，可根据需要进行选择来设置监视类型。依据监视类型，可将监视分为监视点和监视表达式。当选择类型为**当监视值为真时中断**或**当监视值改变时中断**时，添加的监视称为监视点。监视点实际上是一个特殊的表达式，当该表达式为 True(非 0)，监视点的值改变时，程序执行中断，其作用与断点类似，但它是有条件中断。对于监视点，启动程序

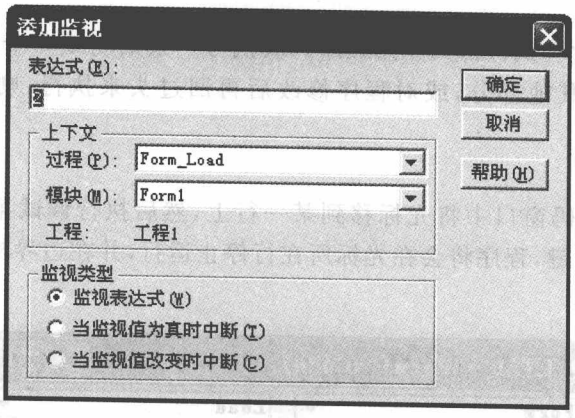


图 1.13　**添加监视**窗口

后，VB 每执行一条语句都要计算表达式的值，并在指定的条件满足时中断程序执行。例如，有以下一段程序：

```
Private Sub Form_Click( )
  Sum=0
  For i=1 To 1000
    Sum=Sum+i
    Print Sum
  Next i
End Sub
```

执行**调试**菜单中**添加监视**命令，打开**添加监视**对话框，在**表达式**文本框中输入 **Sum＞100**，在监视类型部分选择**当监视值为真时中断**。然后运行程序，当变量 Sum 的值超过 100 时，程序执行中断，如图 1.14 所示。可以在**监视**窗口看到，此时表达式 **Sum＞100** 的值已变为 True。

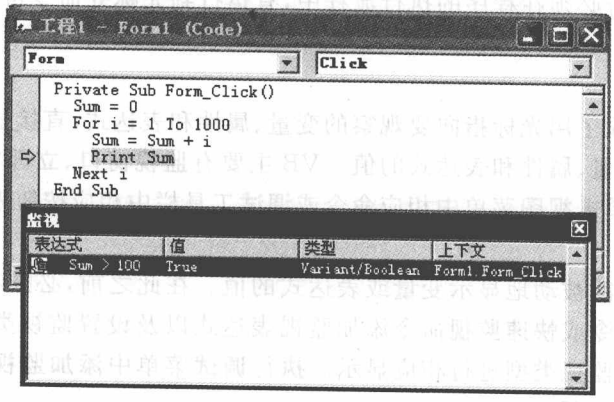


图 1.14　当监视值为真时中断

设置监视点后，由于不断地检查表达式的值，会使程序的执行变慢。因此，在实际调试程序时，宜将监视点和断点结合起来使用，在有可能发生错误的地方设置断点，程序以正常速度运行到断点处中断后，再设置一个或多个监视点，以较慢速度执行程序。

当监视类型选择为监视表达式时，添加的监视称为监视表达式，利用它可以跟踪多个变量或表达式的变化轨迹。例如，对于上面那段程序，如果想观察变量 Sum 的变化，可以在**添加监**