第 1 篇　VB.NET 语言基础知识

第 1 章　VB.NET 语言综述

VB.NET 语言是一种简洁、类型安全的面向对象的编程语言，主要用来构建在.NET Framework 上运行的各种安全、可靠的应用程序。

本章要点

- VB.NET 语言及其特点；
- VB.NET 语言的编译和运行环境；
- VB.NET 程序的创建、编译和运行；
- VB.NET 程序的基本结构；
- 命名空间；
- Main 方法与命令行参数；
- VB.NET 注释与 XML 文档注释；
- 控制台输入和输出。

1.1　VB.NET 语言概述

1.1.1　VB.NET 语言简介

1964 年美国 Dartmouth 学院的 J. Kemeny 和 T. Kurtz 教授共同设计了 BASIC（Beginners All-purpose Symbolic Instruction Code，初学者通用的符号指令代码）语言。随后 BASIC 语言不断发展，已经历了基本 BASIC 语言、BASIC 语言（MS-BASIC 和 GS-BASIC）、结构化 BASIC 语言（Turbo BASIC 和 QBASIC）和 Visual Basic 语言 4 个发展阶段。BASIC 语言是一种容易学习、功能强、效率高的编程语言，被广泛用于各种编程环境。

1991 年微软公司推出 Visual Basic 1.0（VB），VB 支持可视化界面设计、以事件驱动为运行机制，是 Windows 环境下广泛使用的编程语言之一。随后经历多次版本升级，提供更多功能更强的用户控件；增强网络等功能。1998 年微软公司推出 Visual Basic 6.0。

2000 年微软公司推出了.NET 开发平台。Visual Basic.NET 是在.NET 平台上编程的一种高级语言。由于 Visual Basic.NET 是从 Visual Basic 6.0 发展而来，因此也可称其为 Visual Basic 7.0。

VB.NET 作为微软.NET Framework 的主要语言，其主要发展历史如表 1-1 所示。

表 1-1　VB.NET 主要发展历史

发布时间	开发工具	开发平台	CLR 版本	VB.NET 版本
2002/02	Visual Studio .NET 2002	.NET Framework 1.0	1.0	7.0
2003/04	Visual Studio .NET 2003	.NET Framework 1.1	1.1	7.1
2005/11	Visual Studio 2005	.NET Framework 2.0	2.0	8.0
2006/11	Visual Studio 2005+Extension	.NET Framework 3.0	2.0	8.0
2007/11	Visual Studio 2008	.NET Framework 3.5	2.0	9.0

续表

发 布 时 间	开 发 工 具	开 发 平 台	CLR 版本	VB.NET 版本
2010/04	Visual Studio 2010	.NET Framework 4.0	4.0	10.0
2012/08	Visual Studio 2012	.NET Framework 4.5	4.0	11.0
2013/10	Visual Studio 2013	.NET Framework 4.5.1	4.0	11.0
2015/07	Visual Studio 2015	.NET Framework 4.6	4.0	14.0
2017/04	Visual Studio 2017	.NET Framework 4.7	4.0	15.0
2019/04	Visual Studio 2019	.NET Framework 4.7	4.0	16.0

本书主要基于 Visual Studio 2019/.NET Framework 4.7，讲述 Visual Basic 16.0 的语言基础知识，以及使用 Visual Basic 16.0 语言的开发应用实例。

注：本书涉及的内容绝大部分也适用于 Visual Studio 2010 及以后的版本。

1.1.2 VB.NET 语言各版本的演变历史

VB.NET 语言的各版本的主要演变历史及新增功能如下。

1. Visual Basic 7.0（Visual Studio.NET）：新语言诞生

Visual Basic 7.0（VB.NET）是为.NET Framework 设计的 Visual Basic（从 Visual Basic 6.0 发展而来），是一种面向对象的编程语言。由于其使用了新的核心和特性，所以不兼容老版本的 VB 程序，很多 VB 的程序需要改写迁移后才能正常运行。

2. Visual Basic 8.0（Visual Studio 2005）：泛型

增加了 My 伪命名空间和帮助程序类型（对应用、计算机、文件系统、网络的访问），可以帮助用户快速开发应用程序。增加了泛型、操作符重载等新语言特性。

3. Visual Basic 9.0（Visual Studio 2008）：LINQ

提供支持 IIF 函数、匿名类、LINQ、Lambd 表达式、XML 数据结构等新语言特性。

4. Visual Basic 10.0（Visual Studio 2010）：动态编程

提供了动态语言运行时（dynamic language runtime，DLR)、自动实现属性、集合初始化、泛型协变/逆变、全局命名空间访问、不需要在代码断行书写时输入下划线“_”等新语言特性。

5. Visual Basic 11.0（Visual Studio 2012）：异步编程

新增两个关键字 async 和 await，从而实现了更为便捷有效的异步编程方法。增加了迭代器、调用方信息特性。

6. Visual Basic 14.0（Visual Studio 2015）：.NET Core

增加了一些语法糖，可以减少代码编写量。主要包括自动属性初始化、字符串插值等。

.NET Core 是开源的.NET 运行时，基于模块化的 NuGet 包，支持跨平台（各种 Windows 设备、Linux、OS X）。

7. Visual Basic 15.0（Visual Studio 2017）：提高编程效率

增加的不少新特性和语法糖，可提升编程效率并降低出错率。主要包括元组、二进制文本和数字分隔符等。

8. Visual Basic 16.0（Visual Studio 2019）：全新 AI 支持

新增了一键清除代码（即单击即可处理所有的警告信息）、Visual Studio 的全新 AI 支持（Visual Studio IntelliCode）、同时引入了实时共享功能。相应地，微软优化了 Visual Studio 的 Debug 功能，使之变得更加高效便捷。

1.1.3 VB.NET 特点和开发应用范围

1．VB.NET 语言特点

VB.NET 是一种现代的、面向对象的、类型安全的编程语言。VB.NET 具有下列特点。

（1）面向对象

Visual Basic 6.0 是基于对象（object-based）而不是面向对象（object-oriented）的语言，而 Visual Basic.NET 是完全面向对象的语言。VB.NET 支持数据封装、继承、多态和接口。类只能直接从一个父类继承（不支持多重继承），但它可以实现任意数量的接口。所有 VB.NET 类型（包括诸如 Integer 和 Double 之类的基元类型）都继承于一个唯一的根类型：Object。

（2）类型安全（type-safe）

VB.NET 是强类型语言，即每个变量和对象都必须具有声明类型。数组类型下标从零开始而且进行越界检查。

（3）现代 VB.NET 语言包括许多现代先进语言的特性

- 支持属性（property），充当私有成员变量的访问器；
- 支持封装的方法签名（称为“委托”），它实现了类型安全的事件通知；
- 支持特性（attribute），提供关于运行时类型的声明性元数据；
- 支持内联 XML 文档注释，可以编写 API 文档；
- 支持泛型方法和类型，从而提供了更出色的类型安全和性能；
- 语言集成查询（LINQ）表达式使强类型查询成为了一流的语言构造；
- 扩展方法，使用静态方法扩展现有类，这些静态方法可以通过实例方法语法进行调用；
- 匿名类型，无需预先显式定义，其类型名由编译器生成，广泛用于 LINQ 查询表达式；
- 分部方法定义，分部类型可以包含分部方法；
- 迭代器，用于迭代集合中的元素；
- 异步编程，用于提高应用程序的相应能力；
- 动态编程，提供动态编程能力；
- 垃圾回收（garbage collection），将自动回收不再使用的对象所占用的内存；
- 异常处理（exception handling），提供了结构化和可扩展的错误检测和恢复方法。

2．VB.NET 语言开发应用范围

VB.NET 语言主要用来构建在.NET Framework 上运行的各种安全、可靠的应用程序。使用 VB.NET，可以创建下列类型的应用程序和服务：

- 控制台应用程序，基于命令行窗口的控制台（console）应用程序；
- 桌面应用，包括 Windows 窗体应用程序、Windows presentation foundation（WPF）应用程序；
- UWP 应用，通用 Windows 平台应用，是指可以运行于所有以 Windows 10 为内核的系统和设备上，包括桌面设备、移动设备、XBox、HoloLens 甚至物联网设备的应用程序；
- Web 应用，包括 ASP.NET 应用程序、ASP.NET Core、ASP.NET MVC、Web 服务等；
- Office 平台应用程序；
- Windows 服务。

1.2　VB.NET 语言的编译和运行环境

1.2.1　VB.NET 语言与.NET Framework

VB.NET 程序在.NET Framework 上运行。.NET Framework 是 Windows 的一个组件，包括一个称为公共语言运行库（common language runtime，CLR）的虚拟运行环境和一组统一的类库（framework class library，FCL）。

用 VB.NET 编写的源代码被编译为中间语言（intermediate language，IL）。IL 代码与资源（例如位图和字符串）一起作为一种称为程序集的可执行文件存储在磁盘上，通常具有的扩展名

为.exe（应用程序）或.dll（库）。

执行 VB.NET 程序时，程序集将加载到 CLR 中，然后根据程序集清单中的信息执行不同的操作。如果符合安全要求，CLR 执行实时编译将 IL 代码转换为本机机器指令，并执行。CLR 还提供与自动垃圾回收、异常处理和资源管理有关的其他服务。

VB.NET 源代码文件、.NET Framework 类库、程序集和 CLR 的编译时与运行时的关系如图 1-1 所示。

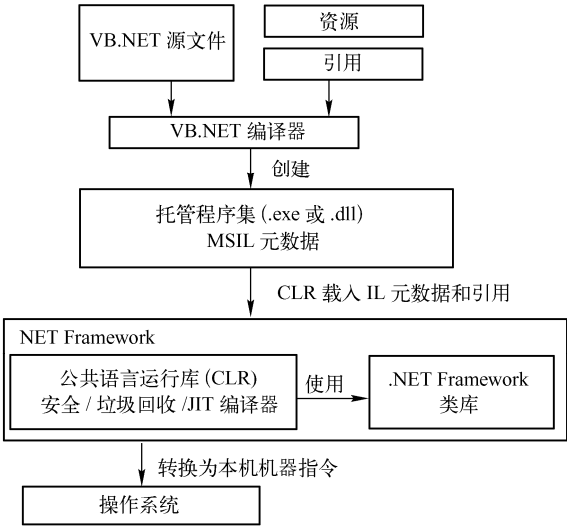

图 1-1 VB.NET 源代码的编译运行环境

注：由 CLR 执行的代码称为“托管代码”，而直接编译为面向特定系统的本机机器语言的代码则称之为“非托管代码”。

有关.NET Framework 的详细信息，请参见附录 A。

1.2.2 VB.NET 的运行环境

VB.NET 的运行环境也即.NET Framework 的运行环境。

（1）.NET Framework

Windows 7 中包含了.NET Framework 3.5；Windows 10 中包含了.NET Framework 4.6；Windows 10 v1703 中包含了.NET Framework 4.7。安装 Visual Studio 时，也会安装相应版本对应的.NET Framework。可以从 Microsoft 官网下载安装最新版本的.NET Framework。

各版本的.NET Framework 包括相应的语言包，语言包支持本地化信息文本（如错误信息）的显示，每种语言对应一个语言包，可以同时安装多个语言包。

（2）.NET Core

.NET Core 是新一代的开源.NET Framework 开发和运行环境，是具有跨平台（Windows、Mac OSX、Linux）能力的应用程序开发框架。

（3）Mono

Mono 是由 Xamarin 公司开发的跨平台开源.NET 开发和运行环境，同样是具有跨平台（Windows、Mac OS X、Linux、Android）能力的应用程序开发框架。请参见官网：http://www.mono-project.com/。

1.2.3 VB.NET 的开发环境

要开发 VB.NET 应用程序，可以使用文本编辑器（如 Notepad）编写代码，并使用.NET

Framework 中的编译器进行编译、运行；也可以使用微软集成开发工具（如 Microsoft Visual Studio）。

1．.NET Framework SDK

.NET Framework 软件开发工具包（SDK）包括开发人员编写、生成、测试和部署.NET Framework 应用程序时所需要的一切，如文档、示例、命令行工具和编译器等。

2．Microsoft Visual Studio

Microsoft Visual Studio 基于.NET Framework 开发应用程序的专业平台，提供了高级开发工具、调试功能、数据库功能和创新功能，帮助在各种平台上快速创建当前最先进的应用程序。

本教程使用下列软件组成一个完整的、基于.NET 的应用系统开发运行环境。

① Windows 10。

② Microsoft Visual Studio Community 2019（社区版）。

注意：可以到微软网站下载 Microsoft Visual Studio Community 2019。其下载网址为 https://visualstudio.microsoft.com/。

3．Xamarin Studio

Xamarin Studio 是一个免费的.NET IDE，可运行于 Windows、Mac OS X 和 Linux。提供类似于 Visual Studio Community 功能，适合于在其他操作系统平台开发构建.NET 应用程序。Xamarin Studio 和 Visual Studio 支持相互读入对方创建的项目。请参见官网 http://xamarin.com/。

4．Visual Studio Code

Visual Studio Code（VS code、VSC）是一款运行于 Mac OS X、Windows 和 Linux 之上的，主要针对网页开发和云端应用的免费开源的现代化轻量级代码编辑器，支持几乎所有主流的开发语言的语法高亮、智能代码补全、自定义热键、括号匹配、代码片段、代码对比 Diff、GIT 等特性，并支持插件扩展。

1.3　创建简单的 VB.NET 程序

1.3.1　“Hello World”程序

本节讲述 VB.NET 程序设计的基本流程，为了读者能够深入了解 VB.NET 源代码的结构及其编译和运行过程，将采用手工编码的方式。实际开发过程中，一般采用集成开发工具 Microsoft Visual Studio。本书第 2 章将阐述使用 Microsoft Visual Studio 的基本流程。

使用任意编辑软件（如 Notepad.exe）创建程序文件 Hello.vb（VB.NET 源文件的扩展名通常是.vb）。注意：例 1-1 中，行号是为了阐述方便，实际源代码中不包括行号。

【例 1-1】　“Hello World”程序。

```
01  'Chapter01/Hello.vb   (a "Hello World!" program)
02  'compile：vbc Hello.vb   ->    Hello.exe
03  Imports System
04  Module Module1
05      Sub Main()
06          Console.WriteLine("Hello World!")
07          MsgBox("Hello, World!")
08      End Sub
09  End Module
```

1.3.2 代码分析

行 1 和行 2 为注释。

行 3 是一个 Imports 指令，它引用了 System 命名空间。命名空间（namespace）提供了一种分层的方式来组织 VB.NET 程序和库。命名空间中包含类型声明及子命名空间声明（例如，System 命名空间包含 Console 类和其他的类）。这样行 6 就可以通过非限定方式直接使用 Console.WriteLine，以代替完全限定方式 System.Console.WriteLine。注：默认情况下，Visual Basic 编译器自动引用 System 命名空间和 Microsoft.VisualBasic 命名空间，故行 3 也可以省略。

行 4 到行 9 的 Module…End Module 定义了模块 Module1。

行 5 到行 8 的 Sub…End Sub 定义了 Main 过程，将作为程序的入口点。

行 6 通过非限定方式调用 Console 类的静态方法 Console.WriteLine("Hello World!")，在控制台上输出字符串"Hello World!"。System.Console 是 VB.NET 常用的类，本书范例中将大量使用其静态方法用于从控制台输入/输出数据，有关 System.Console 的使用以及格式化约定的信息，请参见附录 D。

行 7 调用了 Microsoft.VisualBasic 的 MsgBox 过程，使用对话框显示信息。编译器自动引用命名空间 Microsoft.VisualBasic，故可以通过非限定方式直接使用 MsgBox 过程。Microsoft.VisualBasic 包含大量实用的过程，请参见附录 C。

1.3.3 编译和运行结果

先执行 Windows【开始】|【所有应用】|【Visual Studio 2019】|【Developer Command Prompt for VS 2019】菜单命令，进入 Visual Studio 2019 命令提示状态，并将当前目录切换到 Hello.vb 所在的目录，然后使用命令行“vbc Hello.vb”调用 Microsoft VB.NET 编译器（有关 VB.NET 程序的编译器的详细信息，请参见附录 B）编译 Hello.vb 程序，如图 1-2 所示。编译后将产生一个名为 Hello.exe 的可执行程序集。

注：Hello.vb 使用了 System.Console 类和 MsgBox 过程（程序集 Microsoft.VisualBasic.dll 和 System.dll）。默认情况下，Microsoft VB.NET 编译器自动连接 Microsoft.VisualBasic.dll 和 System.dll。VB.NET 语言本身不具有单独的运行时库。事实上，.NET Framework 就是 VB.NET 的运行时库。

当此应用程序运行时，先在命令行界面输出 Hello World!，紧接着将弹出一个消息框，显示“Hello World!”，如图 1-3 所示。

图 1-2 编译 Hello.vb 程序

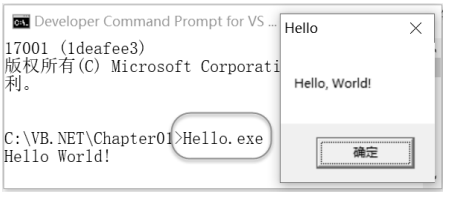

图 1-3 运行 Hello.exe 程序

注：本书提供的源代码，既可以使用 Microsoft VB.NET 编译器编译，也可以使用 Visual Studio 集成开发环境创建相应的项目，然后编译、调试和运行（参见第 2 章）。

1.4 VB.NET 程序的基本结构

1.4.1 程序结构

VB.NET 程序的基本要素如下：

（1）VB.NET 程序由一个或者多个源文件组成（文件后缀为.vb）。

（2）VB.NET 程序源文件中可以声明类型，包含模块、类、结构、接口、枚举和委托等类型。

（3）VB.NET 类型包含数据成员和函数成员，具体包括常量、字段、方法、属性和事件等。

（4）语句是 VB.NET 程序基本构成元素，用于定义类型和类型成员等。

（5）语句通常包含表达式，表达式由操作数和运算符构成，操作数可以是变量或常量。

（6）VB.NET 变量表示存储位置，每个变量必须具有一个类型。

（7）在编译 VB.NET 程序时，源文件被物理地打包为程序集。程序集为应用程序（application）时，其文件扩展名为.exe；应用程序集为库（library）时，其文件扩展名为.dll。可执行应用程序必须包含一个 Main 方法，用于控制程序的开始和结束。

（8）程序中声明的类型按命名空间组织成层次结构。

（9）程序中可以包含注释语句，以增加代码的可维护性。编译时忽略注释信息。

包含这些元素的 VB.NET 程序结构如下所示。

```
'VB.NET 程序结构
Option Explicit On/Off                    'Option 语句 强制显式或者允许隐式声明变量
Imports System                            '引用命名空间
Namespace YourNamespace                   '命名空间
    Module YourModule                     '模块
        [模块成员定义]
    End Module
    Class YourClass                       '类
        [类成员定义]
    End Class
    Structure YourStructure               '结构
        [结构成员定义]
    End Structure
    Interface IYourInterface              '接口
        [接口成员定义]
    End Interface
    Delegate Sub YourDelegate()           '委托
    Enum YourEnum                         '枚举
        [枚举成员定义]
    End Enum
    Namespace YourNestedNamespace         '嵌套的命名空间
        [类型定义]
    End Namespace
    Module YourMainModule                 '程序主模块
        Sub Main(ByVal cmdArgs() As String)   'Main 主程序
            'Your program starts here...       '程序体
    End Sub
    End Module
End Namespace
```

【例 1-2】 在 VBNETBook.Chapter01 的命名空间中声明一个名为 Stack 的类。Stack 实现 FILO（先进后出）的堆栈功能。

```
'Chapter01\Stack.vb
```

```
'compile：vbc /t:library Stack.vb  --> Stack.dll
Imports System
Namespace VBNETBook.Chapter01
    Public Class Stack
        Dim top As Entry
        Public Sub Push(ByVal data As Object)                    '进栈
            top = New Entry(top, data)
        End Sub
        Public Function Pop() As Object                          '出栈
            If (top Is Nothing) Then
                Throw New InvalidOperationException()            '异常处理
            End If
            Dim result As Object = top.data                      '获取堆栈顶端数据
            top = top.nextData
            Return result
        End Function
        Class Entry
            Public nextData As Stack.Entry
            Public data As Object
            Sub New(ByVal nextData As Stack.Entry, ByVal data As Object)
                '成员初始化
                Me.nextData = nextData
                Me.data = data
            End Sub
        End Class
    End Class
End Namespace
```

在例 1-2 中，在 VBNETBook.Chapter01 的命名空间中声明了一个名为 Stack 的类，这个类的完全限定名为 VBNETBook.Chapter01.Stack。Stack 类包含 4 个成员：一个成员变量（top），两个方法（Push 和 Pop）和一个嵌套类（Entry）。Entry 类包含 3 个成员：两个成员变量（nextData 和 data）和一个构造函数（New）。例 1-2 没有 Main 入口点的代码，故执行以下命令行命令：

```
vbc /t:library Stack.vb
```

将例 1-2 编译为一个库，并产生一个名为 Stack.dll 的程序集（库）。

程序集包含中间语言（IL）指令形式的可执行代码和元数据（Metadata）形式的符号信息。在执行程序集之前，.NET 公共语言运行库的实时（just in time，JIT）编译器将程序集中的 IL 代码自动转换为特定于处理器的代码。

【例 1-3】 在 VBNETBook.Chapter01 的命名空间中声明一个名为 StackTest 的类，测试例 1-2 中的 Stack 类。

```
'Chapter01\StackTest.vb
'compile：vbc /r:Stack.dll StackTest.vb  ->  StackTest.exe
Imports System
Imports VBNETBook.Chapter01
Namespace VBNETBook.Chapter01
    Class StackTest
```

```
        Shared Sub Main()
            Dim s As Stack = New Stack()
            s.Push(1)                          '1 进栈
            s.Push(10)                         '10 进栈
            s.Push(100)                        '100 进栈
            Console.WriteLine(s.Pop())         '出栈，输出 100
            Console.WriteLine(s.Pop())         '出栈，输出 10
            Console.WriteLine(s.Pop())         '出栈，输出 1
            Console.ReadLine()
        End Sub
    End Class
End Namespace
```

程序运行结果如下：

```
100
10
1
```

例 1-3 使用了例 1-2 生成的程序集（Stack.dll）中的 Stack 类。由于 VB.NET 程序集是自描述的功能单元，它既包含代码，又包含元数据，因此，VB.NET 中不需要#include 指令和头文件。若要在 VB.NET 程序中使用某特定程序集中包含的公共类型和成员，只需在编译程序时引用该程序集即可。例 1-3 相应的编译命令如下：

```
vbc /r:Stack.dll StackTest.vb
```

将创建名为 StackTest.exe 的可执行程序集，运行结果如上所示。

1.4.2　命名空间

.NET Framework 类库包含大量的类型，用户也可以自定义类型。为了有效地组织 VB.NET 程序中的类型并保证其唯一性，VB.NET 引入了命名空间的概念，从而最大限度地避免类型重名错误。

与文件或组件不同，命名空间是一种逻辑组合。在 VB.NET 文件中定义类时，可以把它包括在命名空间定义中。VB.NET 程序中类型由指示逻辑层次结构的完全限定名（fully qualified name）描述。例如，VBNETBook.Chapter01.HelloWorld 表示 VBNETBook 命名空间的子命名空间 Chapter01 中的 HelloWorld 类。

1．定义命名空间

VB.NET 程序中使用 Namespace…End Namespace 关键字声明命名空间。声明格式如下：

```
Namespace 命名空间名称
End Namespace
```

其中，命名空间名称的一般格式如下：

```
<Company>.(<Product>|<Technology>)[.<Feature>][.<Subnamespace>]
```

例如，微软公司所有关于移动设备的 DirectX 的类型可以组织到命名空间 Microsoft.WindowsMobile.DirectX 中。Acme 公司的 ERP 项目中关于数据访问的类型可以组织到命名空间

Acme.ERP.Data 中。

一个源程序文件中可以包含多个命名空间，同一命名空间可以在多个源程序文件中定义；命名空间可以嵌套；同一命名空间中不允许定义重名的类型。

注意：如果源代码中没有指定 Namespace，则使用默认命名空间。除非简单的小程序，一般不推荐使用默认命名空间。

2．访问命名空间

要访问命名空间中的类型，可以通过如下的完全限定方式访问：

```
<Namespace>[.<Subnamespace>].类型
```

例如，命名空间 System 中的 Console 类的静态方法 WriteLine()，可以使用全限定名称：

```
System.Console.WriteLine("Hello, World!")
```

【例 1-4】 命名空间和类型声明及其关联的完全限定名示例。

```
Class A                 'A  默认命名空间
End Class
Namespace X             'X
    Class B             'X.B
        Class C         'X.B.C
        End Class
    End Class
    Namespace Y         'X.Y
        Class D         'X.Y.D
        End Class
    End Namespace
End Namespace
Namespace X.Y           'X.Y
    Class E             'X.Y.E
    End Class
End Namespace
```

如果应用程序频繁使用某命名空间，为了避免程序员在每次使用其中包含的方法时都要指定完全限定的名称，可以在 VB.NET 应用程序开始时使用 Imports 指令引用该命名空间，以通过非限定方式直接引用该命名空间中的类型。例如通过在程序开头包括行：

```
Imports System
```

可以引用命名空间 System；则在程序中可以直接使用代码：

```
Console.WriteLine("Hello, World!")
```

还可以直接导入指定命名空间中的类型的静态成员，随后在程序中直接使用，进一步减少代码量。例如：

```
Imports static System.Console
Console.WriteLine("Hello, World!")
```

3．命名空间别名

Imports 指令还可用于创建命名空间的别名，别名用于提供引用特定命名空间的简写方法。

使用 Imports 指令指定命名空间或类型的别名的格式如下：

```
Imports 别名 = 命名空间或类型名
```

如果别名指向命名空间，则使用“别名.类型”的形式进行调用；如果别名指向类型名，则使用“别名.方法”进行调用。

【例 1-5】 命名空间别名的使用示例。

```
'Chapter01\AliasNSTest.vb
'compile：vbc AliasNSTest.vb  ->  AliasNSTest.exe
Imports AliasNS = System
Imports AliasClass = System.Console
Namespace CSharpBook.Chapter01
    Class AliasNSTest
        Shared Sub Main()
            AliasNS.Console.WriteLine("Hi 1")
            AliasClass.WriteLine("Hi 2")
            Console.ReadKey()
        End Sub
    End Class
End Namespace
```

程序运行结果如下：

```
Hi 1
Hi 2
```

4．全局命名空间

当成员可能被同名的其他实体隐藏时，可以使用全局命名空间来访问正确的命名空间中的类型。VB.NET 程序中，如果使用全局命名空间限定符 Global.，则对其右侧标识符的搜索将从全局命名空间开始。

【例 1-6】 全局命名空间的使用示例。

```
' Chapter01\GlobalNSTest.vb
' compile：vbc GlobalNSTest.vb  ->  GlobalNSTest.exe
Namespace CSharpBook.Chapter01
    Class GlobalNSTest
        ' 定义一个名为'System'的新类，为系统制造麻烦
        Public Class System
            ' 定义一个名为'Console'的常量，为系统制造麻烦
            Const Console As Integer = 7
            Const number As Integer = 66
            Shared Sub Main()
                ' 出错啦：访问常量 Console
                'Console.WriteLine(number)
                Global.System.Console.WriteLine(number) 'OK
                Global.System.Console.ReadKey()
            End Sub
        End Class
```

```
    End Class
End Namespace
```

程序运行结果如下：

```
66
```

5．命名空间举例

例 1-7 演示了在 2 个不同的命名空间中分别定义名称相同的类（SampleClass），并演示其调用方法。

【例 1-7】 命名空间示例。

```
' Chapter01\NamespaceTest.vb
' compile：vbc NamespaceTest.vb  ->  NamespaceTest.exe
Imports System
Namespace VBNETBook.Chapter01
    Class SampleClass
        Public Sub SampleMethod()
            Console.WriteLine("SampleMethod inside VBNETBook.Chapter01")
        End Sub
    End Class
    Namespace NestedNamespace    ' 创建嵌套的命名空间
        Class SampleClass
            Public Sub SampleMethod()
                Console.WriteLine("SampleMethod inside VBNETBook.Chapter01. Nested Namespace")
            End Sub
        End Class
    End Namespace
    Module Module1
        Sub Main()
            ' 显示"SampleMethod inside VBNETBook.Chapter01."
            Dim outer As SampleClass = New SampleClass()
            outer.SampleMethod()
            '显示"SampleMethod inside VBNETBook.Chapter01."
            Dim outer2 As VBNETBook.Chapter01.SampleClass = New VBNETBook.Chapter01.SampleClass()
            outer2.SampleMethod()
            ' 显示"SampleMethod inside VBNETBook.Chapter01.NestedNamespace."
            Dim inner As NestedNamespace.SampleClass = New NestedNamespace.SampleClass()
            inner.SampleMethod()
            '显示"SampleMethod inside VBNETBook.Chapter01.NestedNamespace."
            Dim inner2 As VBNETBook.Chapter01.NestedNamespace.SampleClass = _
                    New VBNETBook.Chapter01.NestedNamespace.SampleClass()
            inner2.SampleMethod()
            Console.ReadKey()       '按任意键结束
        End Sub
    End Module
End Namespace
```

程序运行结果如下：

```
SampleMethod inside VBNETBook.Chapter01
SampleMethod inside VBNETBook.Chapter01
SampleMethod inside VBNETBook.Chapter01.NestedNamespace
SampleMethod inside VBNETBook.Chapter01.NestedNamespace
```

1.4.3　类型

VB.NET 程序主要由.NET Framework 类库中定义的类型和用户自定义类型组成。VB.NET 程序主要包含模块、类、结构、接口、枚举、委托等类型。

类是最基础的 VB.NET 类型。类是一个数据结构，将状态（数据成员）和操作（方法和其他函数成员）组合在一个单元中，用于实现诸如 Windows 窗体、用户界面控件和数据结构等功能元素。可以使用 New 运算符创建类的实例对象，通过调用对象的方法进行各种操作，实现应用程序的不同功能。

“模块”（又称为“标准模块”），一般用于定义全局的变量、属性、事件和过程。模块具有与程序相同的生存期。

有关 VB.NET 类型的详细信息，后续章节将陆续展开。

【例 1-8】 类和模块示例。在命名空间 VBNETBook.Chapter01 中定义类 Point；然后定义模块 TestModule，创建类 Point 的对象实例。

```
' Chapter01\ClassAndObject.vb
' compile：vbc ClassAndObject.vb  ->  ClassAndObject.exe
Namespace VBNETBook.Chapter01
    Public Class Point                                  '定义平面点坐标
        Public x As Integer
        Public y As Integer
        Public Sub New(ByVal x As Integer, ByVal y As Integer)
            Me.x = x
            Me.y = y
        End Sub
    End Class
    Module TestModule
        Sub Main()
            Dim p1 As Point = New Point(0, 0)           '点 1
            Dim p2 As Point = New Point(10, 20)         '点 2
            Console.WriteLine("两个点的坐标分别为：")
            Console.WriteLine("p1: x={0}, y={1}", p1.x, p1.y)
            Console.WriteLine("p2: x={0}, y={1}", p2.x, p2.y)
            Console.ReadKey()                           '按任意键结束
        End Sub
    End Module
End Namespace
```

程序运行结果如下：

```
两个点的坐标分别为：
p1: x=0, y=0
p2: x=10, y=20
```

1.4.4 Main 过程

1．Main 过程概述

VB.NET 的可执行程序（扩展名通常为.exe）必须包含一个 Main 过程，用于控制程序的开始和结束。Main 过程是驻留在模块、类或结构内的静态过程，在 Main 过程中可以创建对象和执行其他过程。

控制台应用程序可以独立运行，因此必须提供一个 Main 过程。Windows 窗体应用程序可以独立运行，但 Visual Basic 编译器会在此类应用程序中自动生成一个 Main 过程，因而不需要编写此过程。

在编译VB.NET控制台或 Windows 应用程序时，默认情况下，编译器会在源代码中查找 Main 过程，并使该过程成为程序的入口。如果有多个 Main 过程，编译器就会返回一个错误。但是，可以使用/main 选项，其后跟 Main 过程所属类的全名（包括命名空间），明确告诉编译器把哪个 Main 过程作为程序的入口点。

注：如果使用 Visual Studio 集成开发环境，则可以通过项目属性窗口指定程序的主入口点。请参见第 2 章例 2-4。

【例 1-9】 Main 过程编译选项示例。

```
'Chapter01/MainTest.vb
'compile error：vbc MainTest.vb   -> MainTest.exe
'compile OK：vbc /main:VBNETBook.Chapter01.HelloWorld2 MainTest.vb -> MainTest.exe
Namespace VBNETBook.Chapter01
    Class HelloWorld1
        Shared Sub Main()
            Console.WriteLine("Hello World 1!")
            Console.ReadKey()                '按任意键结束
        End Sub
    End Class
    Class HelloWorld2
        Shared Sub Main()
            Console.WriteLine("Hello World 2!")
            Console.ReadKey()                '按任意键结束
        End Sub
    End Class
End Namespace
```

程序运行结果如下：

```
Hello World 2!
```

2．Main 过程声明

VB.NET 可以使用下列方式之一声明 Main 过程。

方法一：不带参数，不返回值。声明为一个不使用参数的 Sub 过程。

```
Module Module1
    Sub Main()
        ' 主程序代码
    End Sub
End Module
```

方法二：不带参数，返回整型值。声明为一个不使用参数的 Function 过程。

```
Module Module1
    Function Main() As Integer
        Dim returnValue As Integer = 0
        '主程序代码
        ' 返回值 0 通常表示程序执行成功
        Return returnValue
    End Function
End Module
```

方法三：带参数，不返回值。声明为一个使用 String 数组参数的 Sub 过程。

```
Module Module1
    Sub Main(ByVal cmdArgs() As String)
        '主程序代码
    End Sub
End Module
```

方法四：带参数，返回整型值。声明为一个使用 String 数组参数的 Function 过程。

```
Module Module1
    Function Main(ByVal cmdArgs() As String) As Integer
        Dim returnValue As Integer = 0
        '主程序代码
        ' 返回值 0 通常表示程序执行成功
        Return returnValue
    End Function
End Module
```

如果在类中声明 Main 过程，则必须使用 Shared 关键字。在模块中，Main 默认为 Shared，无需也不允许显示声明。例如：

```
Class MainClass
    Shared Sub Main()
        ' 主程序代码
    End Sub
End Class
```

3．命令行参数

Main 过程的参数是表示命令行参数的 String 数组。通常通过测试 cmdArgs.Length 属性来检查参数是否存在，cmdArgs(0)表示第一个参数，cmdArgs(1)表示第二个参数，依次类推。

可以使用 For 语句或 For Each 语句循环访问命令行参数字符串数组。如果需要，可以使用 Convert 类或 Parse 过程（请参见 3.5 节）将命令行字符串参数转换为数值类型。

【例 1-10】 命令行参数示例：输出命令行参数个数以及各参数内容。

```
'Chapter01/CommandLine.vb
'compile：vbc CommandLine.vb   -> CommandLine.exe
Module mainModule
```

```
    Sub Main(ByVal cmdArgs() As String)
        Console.WriteLine("参数个数 = {0}", cmdArgs.Length)
        '使用 for 语句输出各参数值
        If cmdArgs.Length > 0 Then
            For argNum As Integer = 0 To cmdArgs.Length-1
                Console.WriteLine("Arg({0}) = {1}", argNum, cmdArgs(argNum))
            Next argNum
        End If
        Console.ReadKey()       '按任意键结束
    End Sub
End Module
```

程序运行结果（不带参数、带参数）如图 1-4 所示。

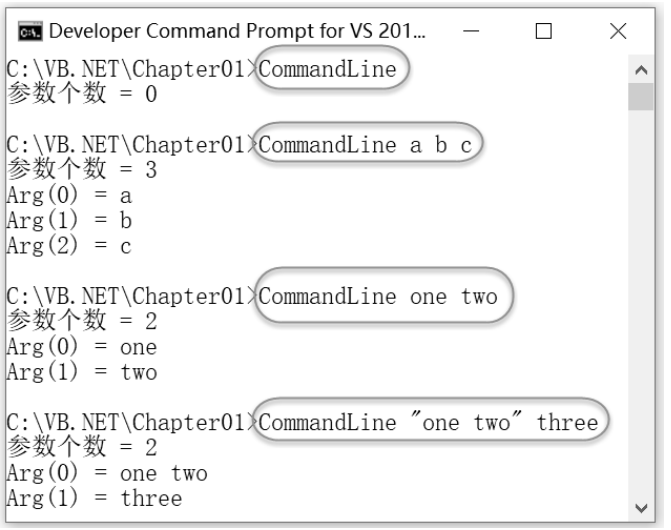

图 1-4　例 1-10 的运行结果

4．Main 返回值

Main 过程可以不返回值，也可以返回整型值。如果不需要使用 Main 的返回值，则使用 Sub 过程可以使代码变简洁。而返回整数可使程序将状态信息传递给调用该可执行文件的其他程序或脚本文件。

【例 1-11】 Main 返回值示例：如果不带命令行参数，则给出相应的提示；否则，输出命令行参数信息。

```
'Chapter01/MainRVTest.vb
'compile：vbc MainRVTest.vb   -> MainRVTest.exe
Module mainModule
    Function Main(ByVal cmdArgs() As String) As Integer
        If cmdArgs.Length = 0 Then
            Console.WriteLine("请输入一个 string 作为参数!")
            Return 1
        Else
            Console.WriteLine("Hello," + cmdArgs(0))
```

```
            Return 0
        End If
    End Function
End Module
```

MainRVTest.exe 运行结果（不带参数、带参数）如图 1-5 所示。

可以使用批处理文件调用前面的代码示例所生成的可执行文件 MainRVTest.exe：如果没有输入参数，则执行失败，并给出提示信息；如果输入参数，则执行成功，并输出命令行参数信息。使用记事本（notepad.exe）创建批处理文件 MainRVTest.bat，其内容如下所示。

```
rem MainRVTest.bat
@echo off
MainRVTest
@if "%ERRORLEVEL%" == "0" goto good
:fail1
    echo Execution Failed
    echo return value = %ERRORLEVEL%
    goto end1
:good1
    echo Execution Succeded
    echo return value = %ERRORLEVEL%
    goto end
:end1
MainRVTest    Mary
@if "%ERRORLEVEL%" == "0" goto good0
:fail0
    echo Execution Failed
    echo return value = %ERRORLEVEL%
    goto end0
:good0
    echo Execution Succeded
    echo return value = %ERRORLEVEL%
    goto end0
:end0
```

批处理文件 MainRVTest.bat 运行结果如图 1-6 所示。

图 1-5　例 1-11 的运行结果

图 1-6　MainRVTest.bat 的运行结果

1.4.5 编码规则

VB.NET 主要的编码规则如下：

① 代码不区分字母的大小写；

② 一行可以书写多条语句，语句间使用冒号（:）分隔；

③ 一条语句可以分多行书写。在每行的最后，一个空格后跟一个下划线字符（_）和一个回车符表示该语句续行；（注：Visual Studio 2010 支持直接回车续行）

④ 适当增加注释以增加程序的可读性。Visual Basic 在编译过程中忽略注释，并且注释不影响编译后的代码。

1.4.6 注释

1．VB.NET 注释

VB.NET 使用单引号（'）字符或者关键字 Rem 将该行的其余内容转换为注释内容。例如：

```
' This is a single line comment     '2018/12/08
Rem This is a single-line comment
System.Console.WriteLine("This will compile")        ' 打印信息
```

注意：字符串中出现的注释字符会按照一般的字符来处理，不再作为注释语句。例如：

```
System.Console.WriteLine( " 'This is just a normal string")
```

运行结果为“' This is just a normal string”。

2．XML 文档注释

VB.NET 除了支持上述风格的注释外，还支持特定的以三个单引号（'''）开头的单行注释。在这些注释中，可以把包含类型和类型成员的文档说明的 XML 标识符放在代码中。使用/doc 进行编译时，编译器将在源代码中搜索所有的 XML 标记，并创建一个 XML 格式的文档文件。

【例 1-12】 XML 文档注释信息的使用示例。

```
'Chapter01\XMLDoc.vb
'compile：vbc /doc:XMLDoc.xml XMLDoc.vb  ->  XMLDoc.xml / XMLDoc.exe
Imports System
''' <summary>
''' XML 注释文档示例。</summary>
''' <remarks>
'''  本示例演示使用 XML 注释生成 XML 注释文档的方法和过程 </remarks>
Public Class XMLDoc
    ''' <summary>
    '''  在控制台窗口中显示欢迎信息。 </summary>
    ''' <param name="sName">sName：用户名字符串。</param>
    ''' <seealso cref="String">请参见 String。</seealso>
    Public Shared Sub SayHello(ByVal sName As String)
        Console.WriteLine(sName + ", Welcome to VB.NET world!")
    End Sub
    ''' <summary>
    '''  应用程序的入口点。
    ''' </summary>
```

```
    ''' <param name="args">用户名</param>
    Public Shared Function Main(ByVal args() As String) As Integer
        If (args.Length = 0) Then
            Console.WriteLine("请输入您的姓名，形式如下：XMLDoc.exe yourname")
            Return 1
        Else
            XMLDoc.SayHello(args(0))
            Return 0
        End If
    End Function
End Class
```

程序运行结果（不带参数、带参数）如图 1-7 所示。

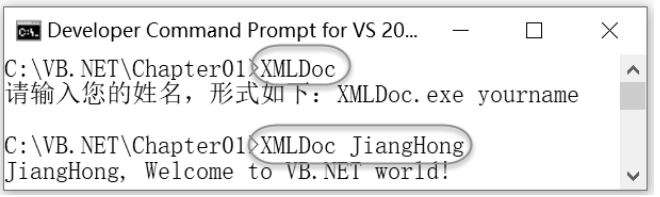

图 1-7　例 1-12 的运行结果

有关 VB.NET XML 文档注释及其支持的 XML 的标记的详细信息，请参见附录 E。

1.5　控制台输入和输出

编写基本的 VB.NET 程序时，常常使用 System.Console 类的几个静态方法来读写数据。输出数据时，则需要根据数据类型通过格式化字符串进行格式化。

1.5.1　System.Console 类概述

System.Console 类表示控制台应用程序的标准输入流、输出流和错误流。控制台应用程序启动时，操作系统会自动将三个 I/O 流（In、Out 和 Error）与控制台关联。应用程序可以从标准输入流（In）读取用户输入；将正常数据写入到标准输出流（Out）；将错误数据写入到标准错误输出流（Error）。

Console 类提供用于从控制台读取单个字符或整行的方法，常用的方法如表 1-2 所示。

表 1-2　System.Console 类提供的常用方法

方法	说明
Beep	通过控制台扬声器播放提示音
Clear	清除控制台缓冲区和相应的控制台窗口的显示信息
Read	从标准输入流读取下一个字符
ReadKey	获取用户按下的下一个字符或功能键
ReadLine	从标准输入流读取下一行字符
Write	将指定值的文本表示形式写入标准输出流
WriteLine	将指定的数据（后跟当前行终止符）写入标准输出流

1.5.2　控制台输入输出

使用 System.Console 类提供的静态方法，可以实现控制台的输入和输出。在控制台程序

中，大量使用该方法实现交互。本书讲解 VB.NET 语言基础知识时，为了侧重语言的基本要素，故主要采用控制台程序。

【例 1-13】 控制台输入/输出示例（ConsoleIO.vb）。

```
'Chapter01\ConsoleIO.vb
'compile：vbc ConsoleIO.vb  ->  ConsoleIO.exe
Module Module1
    Sub Main()
        Dim s As String
        Console.Clear()                          '清屏
        Console.Write("请输入您的姓名：")        '提示输入
        s = Console.ReadLine()                   '读取一行，以回车结束
        Console.Beep()                           '提示音
        Console.WriteLine("欢迎您！" & s)        '输出读取的内容
        Console.ReadKey()                        '按任意键结束
    End Sub
End Module
```

程序运行结果如下：

```
请输入您的姓名：Jiang Hong
欢迎您！Jiang Hong
```

1.5.3 格式化输出

使用 Console.WriteLine()方法输出结果时，可以使用复合格式，控制输出内容的格式。其基本语法为：

```
Console.WriteLine(复合格式字符串, 输出对象列表) '输出对象列表的格式化字符串
String.Format(复合格式字符串, 输出对象列表) '把对象列表格式化成字符串
对象.ToString(复合格式字符串) '把对象格式化成字符串
Format(对象, 复合格式字符串) '把对象格式化成字符串
```

其中，复合格式字符串由固定文本和格式项混合组成，其中格式项又称为索引占位符，对应于列表中的对象。例如：

```
Console.WriteLine("(C) Currency: {0:C}, (E) Scientific:{1:E}", -123, -123.45f);
```

复合格式产生的结果字符串由原始固定文本和列表中对象的字符串的格式化表示形式混合组成。上例的输出结果为：

```
(C) Currency: ￥-123.00, (E) Scientific: -1.234500E+002
```

在上例中，{0:C}/{1:E}为格式项（索引占位符）。其中 0、1 为基于 0 的索引，表示列表中参数的序号，索引号后的冒号后为格式化字符串。在例子中，C 表示格式化为货币（currency）；E 表示格式化为科学计数法（scientific notation）。

有关复合格式字符串的使用及格式化约定的信息，请参见附录 D。

【例 1-14】 复合格式示例（ComFormat.vb）。

```
'Chapter01\ComFormat.vb
'compile：vbc ComFormat.vb  ->  ComFormat.exe
```

```
Module Module1
    Sub Main()
        Dim date1 As DateTime
        Console.WriteLine("{0:C3}", 12345.6789) '显示：￥12,345.679
        Console.WriteLine("{0:D8}", 12345) '显示：00012345
        Console.WriteLine("{0:E10}", 12345.6789) '显示：1.2345678900E+004
        Console.WriteLine("{0:F3}", -17843) '显示：-17843.000
        Console.WriteLine("{0:00000.000}", 123.45) '显示：00123.450
        Dim str1 as String
        str1 = String.Format("{0:#####.###}", 123.45) '使用 String.Format 先格式化成字符串
        Console.WriteLine(str1) '显示：123.45
        date1 = New DateTime(2020, 4, 10, 6, 30, 0)
        Msgbox(date1.ToString("yyyy/MM/dd hh:mm:ss")) '显示：2020/04/10 06:30:00
        Console.ReadKey()       '按任意键结束
    End Sub
End Module
```

程序运行结果如图 1-8 所示。

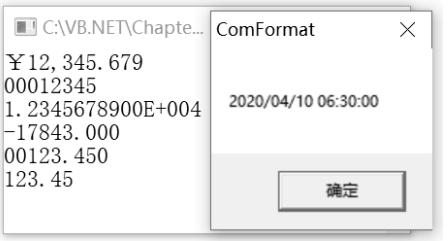

图 1-8　复合格式示例的运行结果

1.6　Visual Basic 运行时库交互函数/过程

Visual Basic 包含许多实用用户交互函数/过程，可以实现用户交互输入和输出。Visual Basic 运行时包含的模块及各模块包含的内容，请参见附录 C。

1.6.1　使用 MsgBox 显示消息框

MsgBox 函数用于在对话框中显示消息，等待用户单击按钮，并返回指示用户单击的按钮的整数。其语法形式如下：

```
MsgBox(Prompt, Buttons = vbOkOnly, Title=null)
```

其中，必选参数 Prompt 是要显示的消息；可选参数 Buttons 是要显示的按钮：vbOkOnly（只显示一个“确定”按钮，默认值）、vbOkCancel（显示“确定”和“取消”按钮）、vbAbortRetryIgnore（显示“终止”“重试”“忽略”按钮）、vbYesNo（显示“是”和“否”按钮）、vbRetryCancel（显示“重试”和“取消”按钮）、vbCritical（显示错误图标）、vbQuestion（显示疑问图标）、vbExclamation（显示警告图标）、vbInformation（显示信息图标）、vbDefaultButton1（第 1 个按钮是默认值）、vbDefaultButton2（第 2 个按钮是默认值）、vbDefaultButton3（第 3 个按钮是默认值）；可选参数 Title 是对话框的标题，默认值为应用程序名称。

说明：参数 Buttons 既可以使用 Visual Basic 预定义常量（例如 vbOkOnly），也可以使用

MsgBoxStyle 枚举（例如 MsgBoxStyle.OkOnly），还可以使用数值（例如 0）。参数 Buttons 允许按位组合成员值，例如，vbYesNo Or vbDefaultButton2 Or vbCritical，显示错误信息和“是”“否”按钮，且默认值为“否”按钮。

返回值为对应于用户单击的按钮：1（vbOk 或者 MsgBoxResult.OK，确定）；2（vbCancel 或者 MsgBoxResult.Cancel，取消）；3（vbAbort 或者 MsgBoxResult.Abort，终止）；4（vbRetry 或者 MsgBoxResult.Retry，重试）；5（vbIgnore 或者 MsgBoxResult.Ignore，忽略）；6（vbYes 或者 MsgBoxResult.Yes，是）；7（vbNo 或者 MsgBoxResult.No，否）。

【例 1-15】 MsgBox 函数使用示例（MsgBox.vb）。

```
'Ch01\ MsgBox.vb
'compile：vbc MsgBox.vb  ->  MsgBox.exe
Module Module1
    Sub Main()
        Dim msg = "是否继续执行?"
        Dim title = "MsgBoxTest"
        Dim style = vbYesNo Or vbDefaultButton2 Or vbCritical
        Dim response = MsgBox(msg, style, title)
        If response = vbYes Then
            MsgBox("是, 继续执行!", , title)
        Else
            MsgBox("否, 停止执行!", , title)
        End If
    End Sub
End Module
```

程序运行结果如图 1-9 所示。

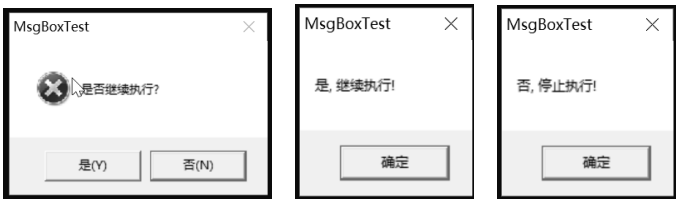

图 1-9　MsgBox 函数使用示例的运行结果

1.6.2　使用 InputBox 提示用户输入

InputBox 函数用于显示输入提示框，等待用户输入文本，单击“确定”按钮后返回用户输入的内容的字符串。其语法形式如下：

```
InputBox(Prompt, Title="", DefaultResponse = "", XPos = -1, YPos = -1)
```

其中，必选参数 Prompt 是提示信息；可选参数 Title 是提示框的标题；可选参数 DefaultResponse 是默认返回值，默认为空字符串；可选参数 XPos 和 YPos 是提示框显示的位置，默认为-1，显示在屏幕中央。

单击“确定”按钮后返回值是用户输入的内容的字符串，单击“取消”按钮后返回值是长度为 0 的空字符串。

【例 1-16】 InputBox 函数使用示例（InputBoxTest.vb）。

```
'Ch01\ InputBoxTest.vb
```

```
'compile：vbc InputBoxTest.vb  ->  InputBoxTest.exe
Module Module1
    Sub Main()
        Dim message, title, defaultValue, response As String
        message = "请输入您的姓名："
        title = "InputBoxTest"
        defaultValue = "World"     '设置默认值
        response = InputBox(message, title, defaultValue)
        MsgBox("Hello, " + response)
    End Sub
End Module
```

程序运行结果如图 1-10 所示。

图 1-10　InputBox 函数使用示例的运行结果