第 1 章　C＃语言简介及运行环境

本章将对C＃语言的起源、性能、特点以及C＃开发环境进行简单介绍，并通过一个简单的实例介绍控制台应用程序的编写方法。

1.1　C＃语言简介

C＃(读为C sharp)是一种专门为.NET应用而开发的程序设计语言，它由微软公司在2000年7月发布，具有简单、安全、面向对象等特点。C＃吸收了C++、Visual Basic、Delphi、Java等语言的优点，体现了当今最新的程序设计技术。

1.1.1　C＃语言的起源

C和C++自从面世以来都已成为开发领域中使用最广泛的语言之一，它们为程序员提供了十分灵活的操作方式，但同时也牺牲了一定的开发效率。与Visual Basic等语言相比，同等级别的C/C++应用程序往往需要更长的开发时间。

虽然目前有些语言以牺牲灵活性为代价提高了开发效率，但这些灵活性正是C/C++程序员所需要的。这些解决方案对编程人员的限制过多(如屏蔽了一些底层代码控制)，提供的功能难以令人满意。同时，这些语言无法很好地和当前的网络编程相结合。

对于C/C++用户来说，最理想的解决方案无疑是在快速开发的同时又可以调用底层平台的所有功能，并能与最新的网络标准保持同步以及能和已有的应用程序良好地整合。基于上述原因，许多程序员试图寻找一种能在功能与效率之间达到一个更为理想的平衡点的新的语言。

C＃是微软公司对上述问题推出的解决方案。微软公司整合了自己最佳的资源，开发出了C＃语言，这是一种最新的、面向对象的程序设计语言。C＃使程序员可以快速地编写各种基于.NET平台的应用程序。

最重要的是，C＃像C++一样能开发出高效的程序。C＃与C/C++具有极大的相似性，熟悉这些语言的开发者可以很快掌握C＃。

1.1.2　C＃语言的应用

C＃面向对象的卓越设计使它成为构建各类组件的理想选择——无论是高级的商业应用，还是系统级的应用程序。通过一些C＃构造，可以将各种组件方便地转变为基于Web的应用，这些应用能够通过网络被各种系统或是使用其他语言开发的应用程序调用。开发人员可以利用C＃在.NET平台上快速开发出种类丰富的应用程序，包括：

(1)Windows应用程序。开发人员可以利用Windows窗体模块创建Windows应用程

序,如传统的 Win32 应用程序。Windows 窗体模块是一个控件库,其中的控件(如按钮、工具栏、菜单等)可以用于建立标准的 Windows 用户界面(UI)。

(2)Web 应用程序。Web 应用程序是指可以通过任何 Web 浏览器查看及使用的 Web 页面。.NET 框架包括一个动态生成 Web 内容的强大系统——ASP .NET,开发人员可以通过 ASP .NET 创建 Web 应用程序。Web 窗体为 Web 应用程序定义了用户接口,包含静态文本和服务控件,而应用程序逻辑则驻留在后台代码文件中。

1.1.3 C#语言的特点

C#是专门为.NET 应用而开发的语言,这从根本上保证了 C#与.NET 框架的完美结合。C#具有以下优点:

(1)语法简单:C#中不再使用指针,只需使用“.”符号即可实现对类成员的引用和访问。

(2)现代性:C#为程序设计提供了许多内建的支持,程序设计人员可以更容易地使用它们创建具有良好性能的功能强大的应用程序。C#语言在数据类型、垃圾回收、内存压缩、异常处理等诸多方面都显示出了它的现代性。

(3)面向对象:C#支持数据封装、继承、多态等所有面向对象的概念。

(4)类型安全:在 C#中不允许进行不安全的类型转换,如不能将 int 类型转换成 boolean 类型;数组下标从零开始,并进行越界检查;对溢出进行检查及异常处理。

(5)兼容性:可以在 C#中直接使用其他中间代码语言编写的组件。C#还允许用户有限制地使用指针。

(6)可伸缩性:要扩展使用 C#编写的程序时,只需用新文件覆盖旧文件,而不需要注册动态链接库。

(7)版本控制:C#支持版本控制。

1.2 C#运行环境

C#程序基于.NET Framework 运行。.NET Framework 是微软公司开发的用于生成、部署和运行应用程序和 XML Web 服务的多语言环境,主要由公共语言运行库和统一编程类组成。

1.2.1 .NET 概述

.NET 是微软公司提供的一种全新的开发平台,这个平台将推动以新体系为基础的协同 Web 应用程序开发。.NET 是微软贴在现有产品和未来产品上的一个新式行销标签,其最令人感兴趣的特色在于它的开发平台、语言和协议。可以说,无论是在技术上还是在战略上,微软都对.NET 寄予了厚望。.NET 程序可以运行在 Windows 98 以上版本的所有 Windows 操作系统上。微软开发的 VS .NET 平台为.NET 程序提供了一个功能强大的集成开发环境,通过使用 VS .NET 可以实现各种.NET 应用系统的快速开发。

1.2.2　.NET 架构

.NET 架构覆盖了在操作系统上开发软件的各个方面，为集成 Windows 或任意平台上的显示技术、组件技术和数据技术提供了最大的可能。使用.NET 架构创建出来的整个体系可以使 Web 应用程序的开发就像桌面应用程序的开发一样简单。

.NET 架构实际上封装了操作系统，可以把用.NET 架构开发的软件与大多数操作系统特性隔离开来，例如，文件处理和内存分配。这样，用.NET 架构开发的软件就可以移植到不同的硬件和操作系统上。

.NET 架构支持 Windows 2003、Windows XP 和 Windows 2000 的所有版本。使用.NET创建的程序也可以运行在 Windows NT、Windows 98、Windows Me 上，但其集成开发环境 VS .NET 不能在这些系统上运行。需要注意的是，在某些情况下，要运行.NET 程序必须先安装相关的服务包。图 1-1 显示了.NET 架构的主要组件，其最高层（即顶层）提供了显示用户界面的多种方式，底层用于进行内存管理和组件加载，顶层和底层之间的层仅提供开发人员需要的系统级功能。

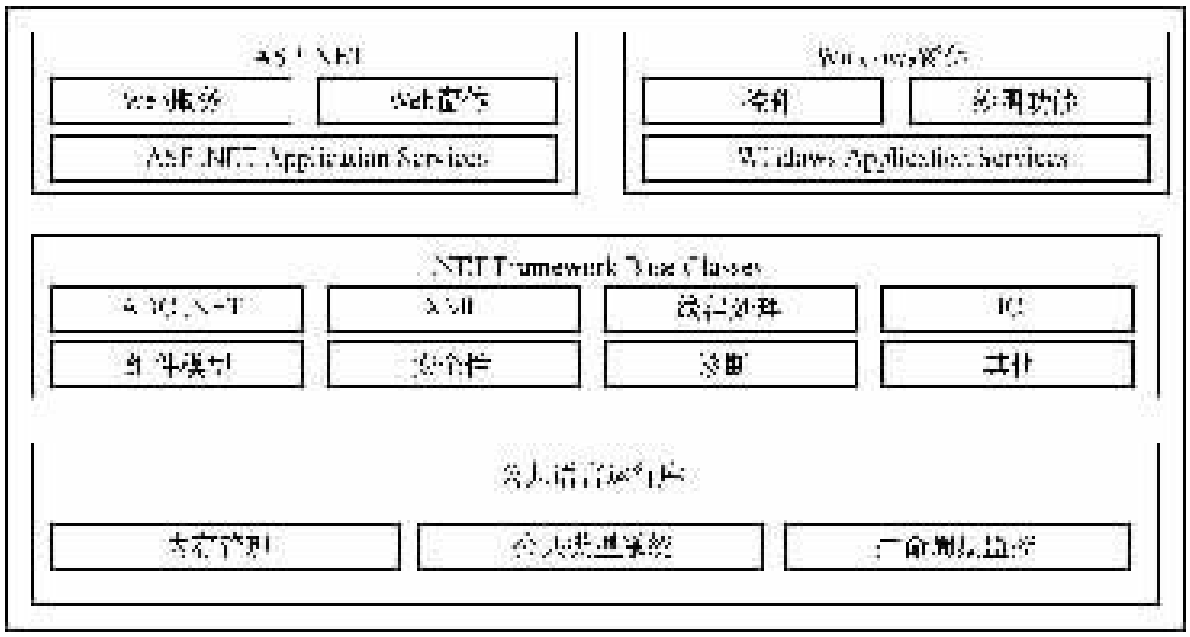

图 1-1　.NET 架构

.NET 架构的顶层用于显示用户界面。Windows 窗体是实现标准 Win32 应用程序窗口的一种更高级的新方式，它具有直观的界面，基本上可以实现开发人员所希望的任何功能。Web 窗体提供了基于网络的用户界面，是一种可以在服务器上动态生成 Web 页面的可缩放公共语言运行库编程模型。作为 ASP 的逻辑演变，ASP .NET 可以创建和使用封装了常用功能的可重用 UI 控件，由此减少了 Web 页面的代码编写量。Web 服务是一种基于 XML 和 HTTPS 的应用程序，其通信协议主要基于 SOAP（Simple Object Access Protocol，简单对象访问协议），可以用任何语言在任何平台上编写 Web 服务。Web 窗体和 Web 服务组成了.NET 与 Internet 的接口，并通过.NET 架构中的 ASP .NET 来实现。借助窗体设计器等工具，VS .NET 为快速开发友好、交互及个性化的 Windows 窗体应用程序和 Web 程序提供了强大支持。

.NET 架构的中间层包括下一代标准系统服务，如管理数据和 XML 的类。这些服务在

架构的控制下可以在各处通用。

.NET 架构的底层是公共语言运行库，通常简写为 CLR(Common Language Runtime)，它是.NET 框架的核心，是驱动关键功能的引擎。CLR 的主要功能是内存管理、安全性检查、生命周期监控和代码管理等。设计公共语言运行库的主要目的是提供极佳的工具支持，实现更快速的开发以及更简单、安全的部署，并使系统具有可伸缩性。

1.2.3 Visual Studio 2005 项目

可以使用多种方法创建.NET 应用程序。例如，可以使用文本编辑器编写 C# 代码，然后下载.NET 开发工具包(SDK)，再使用 C# 命令行编译器 csc.exe 编译.NET 程序，但这种方法在实际应用中很复杂，尤其对于初学者而言。使用可视化编程工具可以使应用程序的开发和编译更为简单，从而提高开发效率。典型的可视化.NET 程序开发工具是由微软公司开发的 Visual Studio 2005。

Visual Studio 2005 具有 GUI 设计器、数据库操作工具、对象和项目浏览工具以及帮助系统，它集成了 Visual Basic 2005、Visual C# 2005、Visual C++ 2005、Visual Web Developer 2005 等多种语言的开发平台。其中，Visual C# 2005 支持的主要项目类型如表 1-1所示。

表 1-1 Visual C# 2005 支持的项目类型

项目类型	描　　述
Windows 应用程序	用于创建 Word 之类的桌面应用程序
类库	用于创建可重复使用的代码库，代码可以跨平台使用
控制台应用程序	可以创建基于控制台的应用程序
空项目	可以生成没有 C# 初始化代码文件的项目

注：本书主要介绍如何进行控制台应用程序编程，用户不需要考虑图形界面的结构，有利于集中学习 C# 语言的基础知识。

1.2.4 Visual Studio 2005 集成开发环境

1. Visual Studio 2005 起始页

启动 Visual Studio 2005，显示一个如图 1-2 所示的起始页。在起始页中可以打开已有的项目或建立新的项目。

2. 新建 Visual C# 项目

在 Visual Studio 2005 集成开发环境中执行“文件”→“新建”→“项目”命令，弹出“新建项目”对话框，可以在此对话框中选择不同的编程语言来创建各种项目，这些语言将共享 Visual Studio 2005 的集成开发环境，如图 1-3 所示。

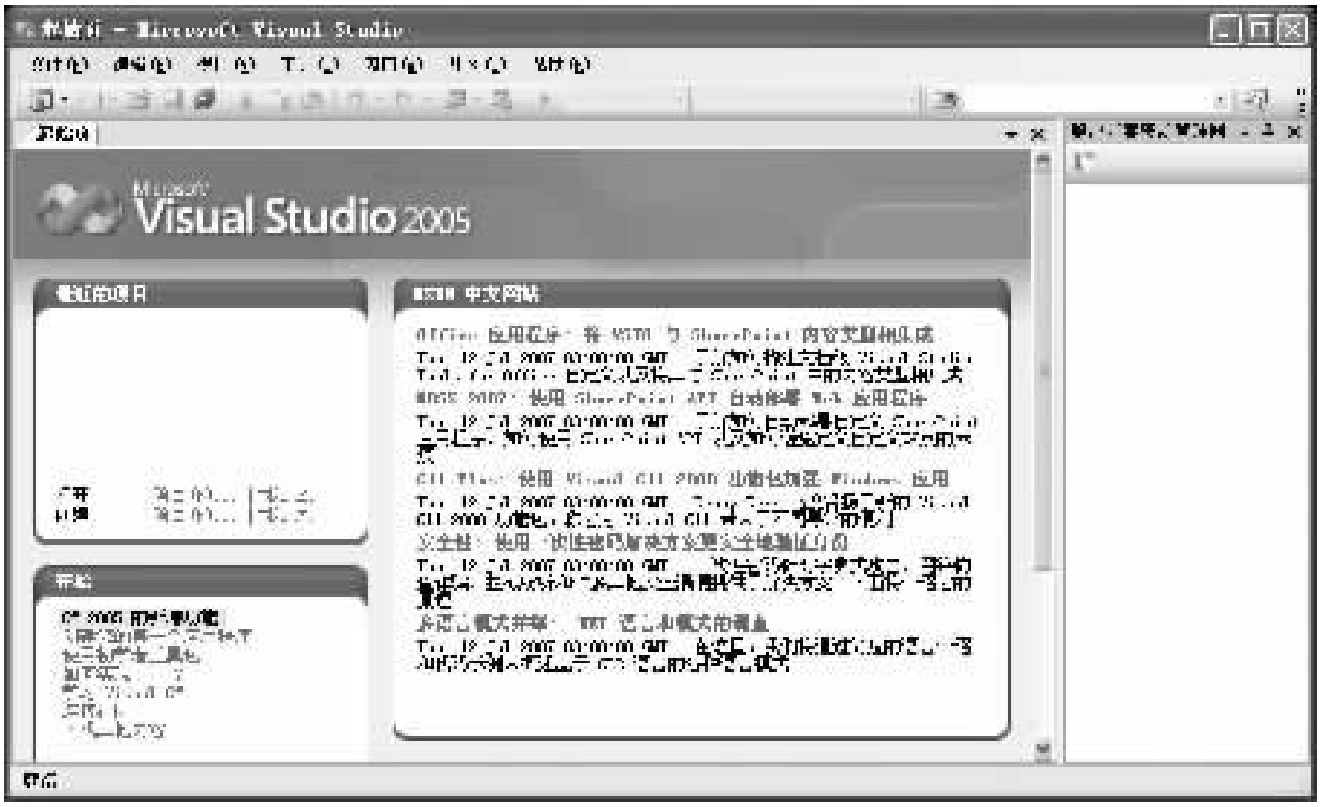

图 1-2　Visual Studio 2005 起始页

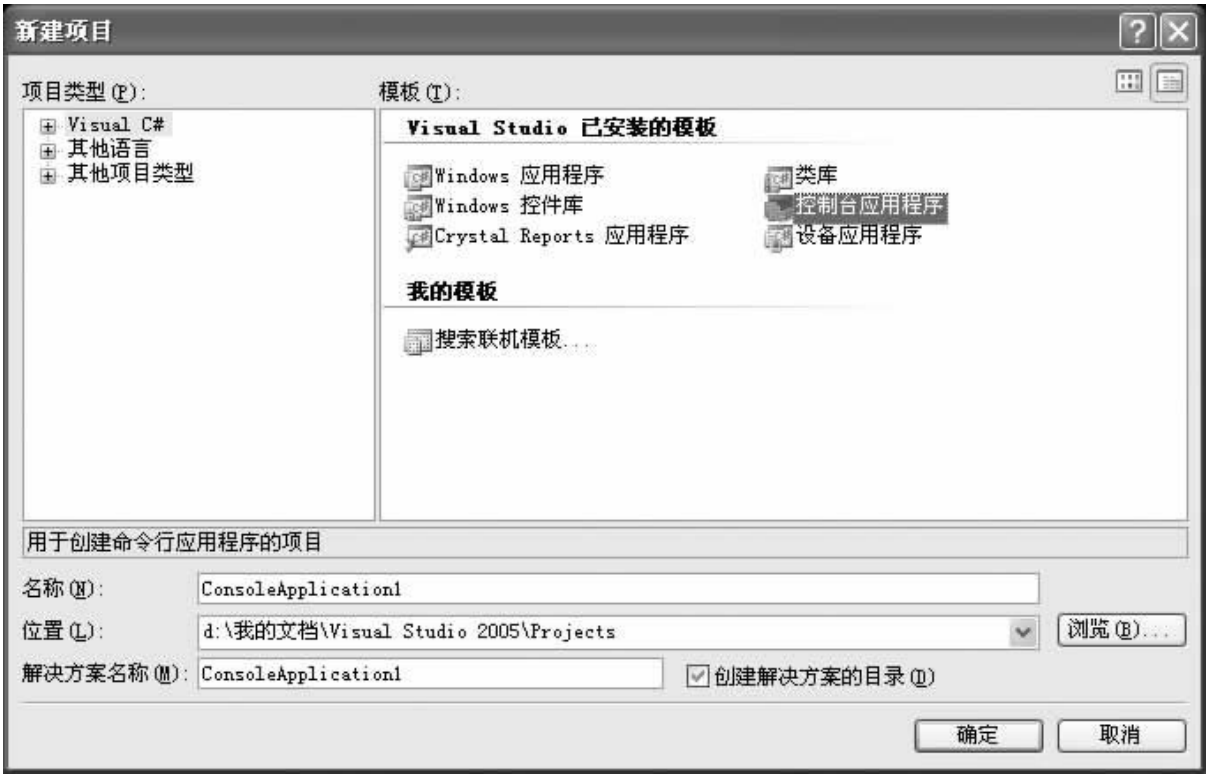

图 1-3　“新建项目”对话框

要创建新的 Visual C＃项目，需要在“新建项目”对话框的“项目类型”列表框中选择“Visual C＃”选项，在“模板”列表框中选择“控制台应用程序”选项，然后在“位置”文本框中输入项目的保存位置(路径)，在“名称”文本框中输入项目名称，单击“确定”按钮，进入如图 1-4 所示的 Visual Studio 集成开发环境。

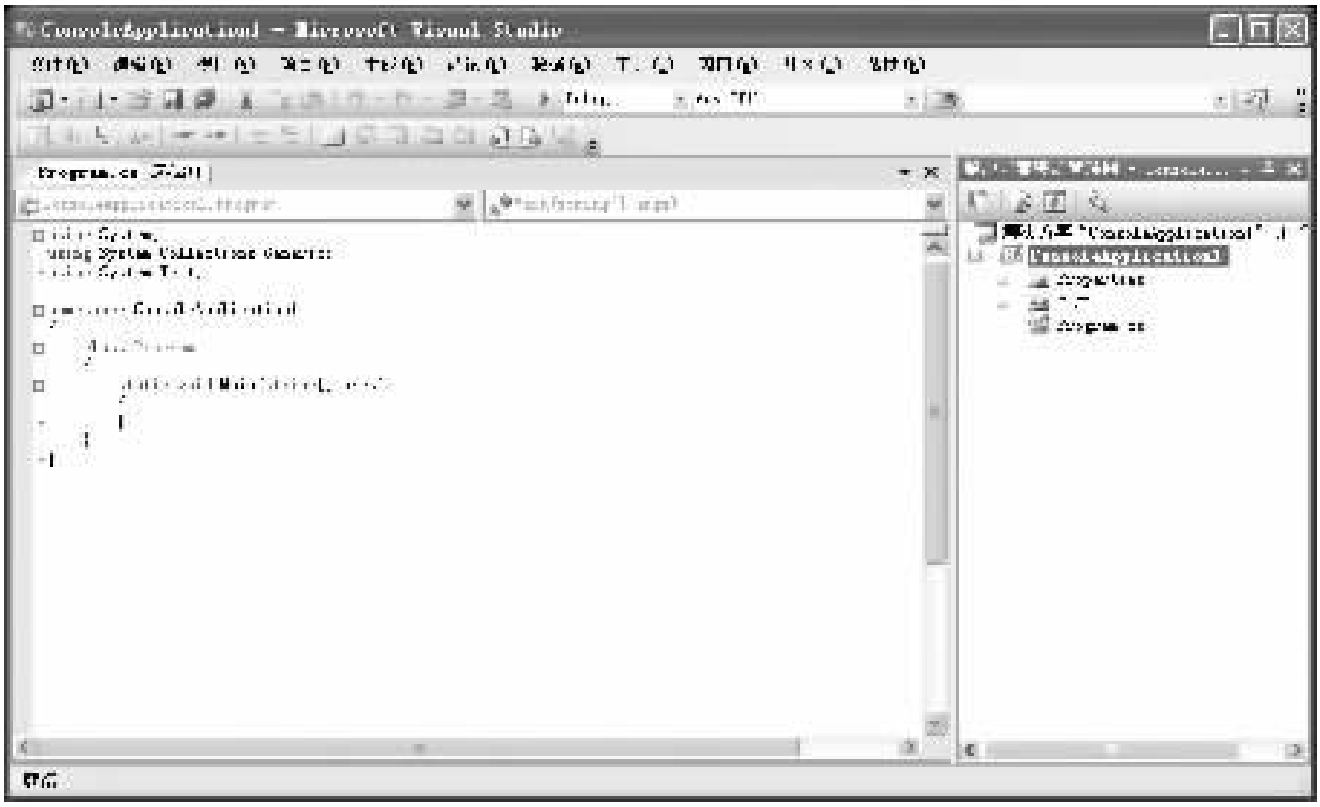

图 1-4 Visual Studio 集成开发环境

3. 解决方案资源管理器

项目可以视为编译后的一个可执行单元，可以是应用程序、动态链接库等。企业级的解决方案往往需要多个可执行程序的合作，为便于管理多个项目，Visual Studio 集成开发环境中引入了解决方案资源管理器，如图 1-5 所示。

图 1-5 解决方案资源管理器

1.2.5 第一个 C# 程序

C# 主要用于开发 3 类程序，即控制台程序、Windows 程序和 ASP .NET 程序。C# 程序的扩展名为.cs，使用编译器 csc.exe 可将其编译成可执行文件。

1. 命名空间

就像在文件系统中用一个目录容纳多个文件一样，可以将命名空间看作某些类的容器。通过命名空间可以把相关的类组织起来，以避免命名冲突。命名空间既可以用于程序的内部组织系统（一种不向其他程序公开自己拥有的程序元素的方法），也可以用于外部组织系统（一种向其他程序公开自己拥有的程序元素的方法）。

命名空间可以包含其他的命名空间，即命名空间可以嵌套。使用命名空间的优点是可以像目录那样方便地对类进行管理。与目录不同的是，命名空间只是一种逻辑上的划分，而不是物理上的存储分类。

声明命名空间的语法格式如下：

```
namespace name
{
  类型声明
}
```

其中，namespace 为关键字，name 为命名空间名称。命名空间名称可以是任何合法的标识符。嵌套使用的命名空间的名称可以简写为以下形式：

```
namespace N1.N2
{
    class A{}
    class B{}
}
```

等同于：

```
namespace N1
{
  namespace N2
  {
  class A{}
  class B{}
  }
}
```

可以在命名空间中声明一个或多个数据类型，如类、接口、结构、枚举、委托等。即使未显式声明命名空间，VS .NET 集成开发环境也会自动创建默认的命名空间，也称为全局命名空间。全局命名空间中的任何标识符都可以在其他命名空间中使用。隐式声明的命名空间具有公共访问权限。

1）系统定义的命名空间

命名空间分为用户定义的命名空间和系统定义的命名空间两类。表 1-2 列出了 C# 常用的系统定义的命名空间。

表 1-2 C#常用的系统定义的命名空间

命名空间	描　　述
System	定义通常使用的数据类型和数据转换的基本.NET 类
System. Collection	定义列表、队列
System. Data	定义 ADO .NET 数据库结构
System. Drawing	提供对基本图形的访问功能
System. IO	允许读写数据流和文件
System. Net	提供对网络的访问功能
System. Net. Mail	提供用于发送邮件消息的类
System. Net. Sockets	提供对 Windows 套接字的访问
System. Runtime. Remoting	提供用于创建和配置分布式应用程序的类和接口
System. Security	提供.NET Framework 安全系统的基础结构
System. Text	提供 ASCII、Unicode、UTF-7 和 UTF-8 字符编码处理
System. Threading	提供支持多线程编程的类和接口
System. Timers	在指定的时间间隔引发事件
System. Web	提供支持浏览器/服务器通信的类和接口
System. Windows. Forms	提供用于创建基于 Windows 应用程序的类
System. XML	提供对 XML 文档处理的支持

2)命名空间的使用

使用 using 指令能够直接引用或使用别名类引用类或命名空间,其语法格式如下:

using [别名=]类或命名空间名;

下面是一个 using 指令的应用示例:

```
using System;  //直接引用
using AliasToMyClass=NameSpace1.MyClass;  //别名引用
namespace NameSpace1
{
  public class MyClass
  {
    public override string ToString()
    {
      return "You are in NameSpace1.MyClass";
    }
  }
}
namespace NameSpace2
```

```
{
  class MyClass
  { }
}
namespace NameSpace3
{
  using NameSpace1;
  using NameSpace2;
  class Test
  {
    public static void Main()
    {
      AliasToMyClass somevar=new AliasToMyClass();
      Console.WriteLine(somevar);
    }
  }
}
```

2. 控制台程序

控制台是一个操作系统窗口，用户可以使用控制台实现与操作系统或基于文本的控制台应用程序的交互。例如，Windows 操作系统中的控制台是一个命令提示窗口，可以接受 MS-DOS 命令。C# 中的 Console 类对从控制台读取字符以及向控制台写入字符的应用程序提供基本支持。控制台应用程序启动时，操作系统会自动将标准输入流、输出流和错误流以 in、out 和 error 属性值的形式提供给控制台。

【例 1-1】 在 Visual Studio 2005 中编写 C# 代码实现一个简单的控制台程序，要求输入名字后显示欢迎进入 C# 世界的信息。

步骤如下：

(1)启动 Microsoft Visual Studio 2005，执行“文件”→“新建”→“项目”命令，打开“新建项目”对话框，如图 1-3 所示。

(2)在“新建项目”对话框中的“项目类型”列表框中选择“Visual C#”选项，在右侧的“模板”列表框中选择“控制台应用程序”选项，然后指定项目名称为“welcomeXM”，存放位置为“D:\C#”，如图 1-6 所示。

(3)单击“确定”按钮，在 Visual Studio 的代码编辑窗口中可看到自动生成的程序代码框架，如图 1-7 所示。

(4)在代码编辑窗口中将类的名称“Program”修改为“WlcXM”，然后在 Main 函数中加入如下代码：

```
Console.WriteLine("请输入你的名字:");
string xm=Console.ReadLine();
Console.WriteLine("欢迎{0}同学进入 C# 世界!",xm);
Console.Read();
```

图 1-6　新建项目

图 1-7　代码编辑窗口

(5)执行“生成”→“生成 welcomeXM”命令编译程序,如图 1-8 所示。

(6)执行“调试”→“启动调试”命令运行程序,弹出如图 1-9 所示的命令行窗口。

(7)在“请输入你的名字:”提示符处输入“Green”,按 Enter 键,显示结果如图 1-10 所示。

图 1-8　编译程序

图 1-9　输入名字

图 1-10　显示欢迎信息

程序分析如下：

(1)程序中的第一至第三行语句的作用是导入命名空间，此语句类似于 C/C++ 中的 #include命令。using. System 语句引用一个由 .NET 框架类库提供的名为 System 的命名空间。System. Collections. Generic 是 C# 2. 0 中新增的一个命名空间，用于保障类型转换的安全。System. Text 命名空间包含与字符串处理和编码相关的类。

(2)namespace 语句声明了一个命名空间 welcomeXM。每个 C# 程序可以包含一个或多个命名空间。

(3)Console 是 System 命名空间中的一个类，利用此类的 ReadLine 和 WriteLine 方法可以进行输入/输出操作：WriteLine 方法用于在屏幕上显示字符串，ReadLine 方法用于读取一行字符串(到换行符但不包括换行符)。

(4)class 为定义类的关键字，类的名称为 WlcXM。

(5)程序中的语句"static void Main(string[] args)"为类 WlcXM 声明了一个主方法。在 C# 程序中，程序的执行总是从 Main 开始，一个程序只能包含一个主方法。

(6)"Console. WriteLine("欢迎{0}同学进入 C# 世界!",xm);"中的"{0}"用于代替

WriteLine方法的参数表中紧随字符串后的第一个变量，可以使用这种方法格式化多个变量，例如：

```
Console.WriteLine("Hello {0} {1},from {2}",strFirstname,strLastname,strCity);
```

提示：本例中的C#程序也可以在Windows记事本等文本编辑工具中编写，然后保存为.cs文件（如welcomeXM.cs），并执行“程序”→“Microsoft Visual Studio 2005”→“Visual Studio Tools”→“Microsoft Visual Studio 2005命令提示”命令，打开命令提示窗口，再切换到welcomeXM.cs所在的目录，输入命令“csc welcomeXM.cs”，按Enter键，即可生成可执行文件welcomeXM.exe。在命令提示窗口中输入命令“welcomeXM.exe”，按Enter键，即可运行程序。

本章小结

C#是专门为.NET平台打造的通用开发工具，具有高效、安全、灵活等特点，现已成为最流行的程序开发语言之一。从最普通的控制台应用到大规模的商业开发，C#与.NET平台的结合都提供了完整的解决方案。

本章先介绍了C#的起源、性能和特点，以及.NET架构，然后介绍了C#程序的开发与运行环境，最后通过一个实例介绍了C#程序的基本结构。

习 题 1

一、简答题

1. 简述C#的性能及特点。
2. 简述.NET架构。

二、操作题

1. 熟悉C#程序的开发与运行环境。
2. 上机调试【例1-1】中的应用程序。