第2章　C#程序基本结构

兴趣是最好的老师。

——爱因斯坦

本章主要知识点

- ❑ *C#*程序的组成元素
- ❑ 类的基本概念
- ❑ 标识符及关键字
- ❑ 命名空间的概念及引用命名空间
- ❑ *C#*代码或代码段的注释

2.1　编写第一个C#程序

教学录像：光盘\mr\video\第2章\编写第一个C#程序.exe

C#是一种面向对象的编程语言，主要用于开发可以在***.NET***平台上运行的应用程序。它是从***C***和***C++***派生来的一种简单、现代、面向对象和类型安全的编程语言，并且能够与***.NET***框架完美结合。对于初学者来说，讲解抽象的理论远不如讲解一个具体的实例。下面就从一个简单的实例讲起，然后再深入学习。具体操作步骤如下。

（1）选择“开始”/“程序”/***Microsoft Visual Studio 2010/Microsoft Visual Studio 2010***命令，打开***Visual Studio 2010***。

（2）选择***Visual Studio 2010***工具栏中的“文件”/“新建”/“项目”命令，打开“新建项目”对话框，如图2.1所示。

图2.1　“新建项目”对话框

（3）选择“控制台应用程序”选项，命名为***Hello_World***，选择保存在***E:\Test01***\上，然后单击“确定”按钮创建一个控制台应用程序。

（4）在***Main***方法中输入代码。

【例2.1】创建一个控制台应用程序，在该程序的***Main***主函数中输出“真聪明，你成功编写了一个***C#***程序！”，代码如下。（实例位置：光盘***mr\example***\第2章\2.1）

代码位置：光盘\mr\example\第2章\2.1\test01\test01\Program.cs

```
01   using System;
02   using System.Collections.Generic;
03   using System.Linq;
04   using System.Text;
05   namespace ConsoleApplication1
06   {
07       class Program
08       {
09           static void Main(string[] args)        //在Main方法中编写代码
10           {
11               Console.WriteLine("真聪明，你成功编写了一个C#程序！");
                                        //输出“真聪明，你成功编写了一个C#程序！”
12               Console.ReadLine();
13           }
14       }
15   }
```

程序的运行结果如图2.2所示。

图2.2　输出字符串“真聪明，你成功编写了一个C#程序！”

说明：***Console***类提供的***WriteLine***方法将指定的数据在控制台输出，然后换行。***Console***类提供的***ReadLine***方法用于从标准输入流读取一行字符。在用户按下回车键之前不会返回，直到用户按下回车键为止。

试一试：参照以上程序，要求在控制台输出“我真聪明，这么快就学会编写***C#***程序了！”。

上机练习

上机练习1　输出多行字符串

本练习要求创建一个控制台应用程序，运行后在控制条中输出“我喜欢***C#***编程！”和“***I Love CSharp!***”两行字符串，如图2.3所示。

图2.3　输出两行字符串

上机练习2 读取字符串并输出

本练习要求创建一个控制台应用程序，调用***System***下的***Console.ReadLine***方法从键盘读取一行字符串，用于存储您输入的的名字，然后再调用***System***下的***Console.WriteLine***方法将输入的名字输出，如图2.4所示。

图2.4 输入并输出您的姓名

2.2 初识C#程序结构

教学录像：光盘\mr\video\第2章\初始C#程序结构.exe

C#程序结构大体可以分为注释、命名空间、类、***Main***方法、标识符、关键字和语句。下面将对***C#***程序的结构进行详细的讲解。

2.2.1 命名空间

C#程序是利用命名空间组织起来的。命名空间既用做程序的“内部”组织系统，也用做向“外部”公开的组织系统（一种向其他程序公开自己拥有的程序元素的方法）。如果要调用某个命名空间中的类或者方法，首先需要使用***using***指令引入命名空间，***using***指令将命名空间名所标识的命名空间内的类型成员导入当前编译单元中，从而可以直接使用每个被导入的类型的标识符，而不必加上它们的完全限定名。

C#中的各命名空间就好像是一个存储了不同类型的仓库，而***using***指令就好比是一个钥匙，命名空间的名称就好比仓库的名称，可以通过钥匙打开指定名称的仓库，从而在仓库中获取所需的物品。

using指令的基本形式为：

```
using  命名空间名;
```

如果要调用命名空间下某个类提供的方法，可以使用下面的语法：

```
命名空间.命名空间……命名空间.类名称.静态方法名(参数, ……);
```

或者

```
命名空间.命名空间……命名空间.实例名称.方法名(参数, ……);
```

例如：

```
System.Console.WriteLine("欢迎您来到C#语言世界！");
```

C#中常用的命名空间如表2.1所示。

表2.1 C#中常用的命名空间

命名空间	描 述
System	定义通常使用的数据类型和数据类型的基本.NET类
System.Collections	定义列表、队列等字符串表

（续）

命名空间	描　述
System.Text	ASCII、Unicode、UTF-7和UTF-8字符编码处理
System.Data	定义ADO.NET数据库结构
System.Drawing	提供对基本图形功能的访问
System.Web	浏览器和Web服务器功能

【例2.2】创建一个控制台应用程序，在***System***命名空间下调用***Console***类的***WriteLine***方法，输出字符串“欢迎您来到***C#***语言世界！”，实现代码如下。

```
01  using System.Collections.Generic;
02  using System.Linq;
03  using System.Text;
04  namespace Hello_World                              //定义Hello_World命名空间
05  {
06      class Program                                  //定义Program类
07      {
08          static void Main(string[] args)            //程序入口方法Main
09          {
10              System.Console.WriteLine("欢迎您来到C#语言世界！");
11              System.Console.ReadLine();             //等待读入信息
12          }
13      }
14  }
```

说明：前缀“***System.***”表示***Console***类在***System***命名空间下。因为在***Main***函数中的代码块中加了前缀***System***，所以在程序的开始就不需要加上命名空间***using System***。

【例2.3】创建一个控制台应用程序，建立一个命名空间***N1***，在命名空间***N1***中有一个类***A***，在项目中使用***using***指令引入命名空间***N1***，然后在命名空间***Test01***中即可实例化命名空间***N***1中的类，然后调用此类中的***show***方法，代码如下。（实例位置：光盘***mr\example***\第2章\2.3）

代码位置：光盘\mr\example\第2章\2.3\test02\test02\Program.cs

```
01  using System;
02  using System.Collections.Generic;
03  using System.Linq;
04  using System.Text;
05  using N1;                                          //引入N1命名空间
06  namespace Test01                                   //声明Test01命名空间
07  {
08      class Program
09      {
10          static void Main(string[] args)            //程序入口方法Main
11          {
12              A a = new A();                         //实例化N1中的类A
13              a.show();                              //调用类A中的show方法
14          }
15      }
16  }
17  namespace N1                                       //建立命名空间N1
18  {
```

```
19        class A                                          //自定义类A
20        {
21            public void show()                           //方法show用于输出信息
22            {
23                Console.WriteLine("只有引用了N1命名空间，类A才可以被使用");//输出字符串
24                Console.ReadLine();
25            }
26        }
27    }
```

运行结果为“只有引用了***N1***命名空间，类***A***才可以被使用”。

注意：如果在程序中没有引用命名空间***N1***，就会显示错误提示。如图2.5所示。

图2.5 没有引用命名空间

试一试：在以上程序的基础上，再创建一个类***B***（也定义***show***方法），定义该类位于命名空间***N2***中，然后在***A***类的***show***方法中创建***B***类的实例，并调用该实例的***show***方法。

2.2.2 类

类是一种数据结构，它可以封装数据成员、函数成员和其他的类。类是创建对象的模板。***C#***中所有的语句都必须位于类内。因此，类是***C#***语言的核心和基本构成模块。***C#***支持自定义类，使用***C#***编程就是编写自己的类来描述实际需要解决的问题。

类就好比医院的各个科室，如内科、骨科、泌尿科、眼科等，在各科室中都有自己的工作方法，相当于在类中定义的变量、方法等。如果要救治车祸重伤的病人，只靠一个科室是不行的，可能要内科、骨科、脑科等多个科室一起治疗才行，这时可以让这几个科室临时组成一个小组，对病人进行治疗，这个小组就相当于类的继承，也就是该小组可以动用这几个科室中的所有资源和设备。

使用任何新的类之前都必须声明它，一个类一旦被声明，就可以当做一种新的类型来使用，在***C#***中通过使用***class***关键字来声明类，声明形式如下。

```
01    [类修饰符]  class  [类名]  [基类或接口]
02    {
03       [类体]
04    }
```

在***C#***中，类名是一种标识符，必须符合标识符的命名规则。类名要能够体现类的含义和用途。类名一般采用第一个字母大写的名词，也可以采用多个词构成的组合词。

【例2.4】下面的代码是声明类***A***，并在该类中声明方法***show***。实现代码如下。

```
01    class A                                              //自定义类A
02    {
03        public void show()                               //声明show方法
```

```
04        {
05            Console.WriteLine("我是自定义的类A！");          //输出信息
06            Console.ReadLine();                              //等待读入信息
07        }
08    }
```

2.2.3 Main方法

Main方法是程序的入口点，***C#***程序中必须包含一个***Main***方法，在该方法中可以创建对象和调用其他方法，一个***C#***程序中只能有一个***Main***方法，并且在***C#***中所有的***Main***方法都必须是静态的。***C#***是一种面向对象的编程语言，即使是程序的启动入口点，它也是一个类的成员。由于程序启动时还没有创建类的对象，因此，必须将入口点***Main***方法定义为静态方法，使它可以不依赖于类的实例对象而执行。

Main方法就相当于汽车的电瓶，在生产汽车时，将各个零件进行组装，相当于程序的编写；当汽车组装完成后，就要检测汽车是否可用，如果想启动汽车，就必须通过电瓶来启动汽车的各个部件，如发动机、车灯等，电瓶就相当于启动汽车的入口点。

默认的***Main***方法代码如下。

```
01    static void Main(string[] args)
02    {
03    }
```

可以用3个修饰符修饰***Main***方法，分别是***public***、***static***和***void***。

- ***public***：说明***Main***方法是共有的，在类的外面也可以调用整个方法。
- ***static***：说明方法是一个静态方法，即这个方法属于类的本身而不是这个类的特定对象。调用静态方法不能使用类的实例化对象，必须直接使用类名来调用。
- ***void***：此修饰符说明方法无返回值。

Main方法是一个特别重要的方法，使用时需要注意以下几点。

- ***Main***方法是程序的入口点，程序控制在该方法中开始和结束。
- 该方法是类或结构的内部声明。它必须为静态方法，而且不能为公共方法。
- 它可以具有***void***或***int***返回类型。
- 声明***Main***方法时既可以使用参数，也可以不使用参数。

2.2.4 标识符及关键字

标识符是指在程序中用来表示事物的单词，例如，***System***命名空间中的类***Console***，以及***Console***类的***WriteLine***方法都是标识符，标识符的命名有3个基本规则，分别介绍如下。

- 标识符只能由数字、字母和下划线组成。
- 标识符必须以字母或者下划线开头。
- 标识符不能是关键字。

例如，在“欢迎您学习***C#***语言！”程序中的***Console***就是标识符，如图2.6所示。

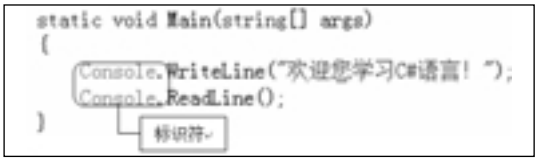

图2.6 标识符

标识符在命名时最好具有一定的含义，例如，***System***命名空间中的类***Console***（译作控制台），以及***Console***类的方法***WriteLine***（译作输出一行）都是标识符。

所谓的关键字是指在***C#***语言中具有特殊意义的单词，它们被***C#***设定为保留字，不能随意使用。例如，在“欢迎您学习***C#***语言！”程序中的***static***、***void***和***string***都是关键字，如图2.7所示。

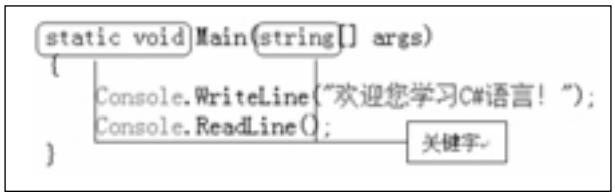

图2.7 关键字

2.2.5 C#语句

语句是构造所有***C#***程序的基本单位。语句可以声明局部变量或常数、调用方法、创建对象或将值赋给变量、属性或字段，语句通常以分号终止。

【例2.5】下面的代码就是一条语句。

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

此语句便是调用***Console***类中的***WriteLine***方法，输出指定的字符串“***Hello World***！”

上机练习

上机练习3 自定义类并输出信息

本例要求仿照“例2.4”自定义一个类***MyFirstClass***，然后在该类的自定义方法***Show***中输出字符串“这是我自定义的第一个类！”

2.3 添加代码注释

教学录像：光盘\mr\video\第2章\添加代码注释.exe

在程序开发过程中，为了方便日后的维护和增强代码的可读性，有必要养成在代码中加入注释内容的良好习惯。在代码关键的位置加入注释，可以帮助理解代码的实现目的与实现方式，使程序工作流程更加清晰明了。

编译器编译程序时不执行注释的代码或文字。注释可以分为对单行代码进行注释、对多行代码注释和给代码添加说明三种，单行注释都以“//”开头，多行注释“/*”开始，以“*/”结束，使用“///”标记给代码段添加说明。

要对单行代码进行注释时，首先选中要注释的内容，然后单击***Visual Studio 2010***工具栏中的“ ”按钮，这样可以注释选中的内容。同样也可以单击***Visual Studio 2010***工具栏中的“ ”按钮取消选中行的注释。

【例2.6】在“真聪明，你成功编写了一个***C#***程序！”程序中使用了单行注释，实现代码如下。

```
01  using System;
02  using System.Collections.Generic;
03  using System.Linq;
```

```
04   using System.Text;
05   static void Main(string[] args)                          //程序的Main方法
06   {
07       Console.WriteLine("真聪明，你成功编写了一个C#程序！");//输出“真聪明，你成功编写了一个C#程序！”
08       Console.ReadLine();
09   }
```

如果注释的行数较少，一般使用行注释。对于连续多行的大段注释，则使用块注释，块注释通常以“/*”开始，以“*/”结束，注释的内容放在它们之间。

【例2.7】在“真聪明，你成功编写了一个C#程序！”程序中使用了多行注释，代码如下。

```
01   static void Main(string[] args)
02   {
03       Console.WriteLine("真聪明，你成功编写了一个C#程序！");
04       Console.ReadLine();
05   }
06   /*程序的show方法中可以输出“I Love C#！”字符串        //块注释开始
07   static void show()
08   {
09       Console.WriteLine("I Love C#！");
10        Console.ReadLine();
11   }
12   */
```

程序的运行结果如图2.8所示。

图2.8　使用“/*”“*/”注释

说明：由于对show方法使用了多行注释，所以这段代码在编译时不能被执行。

注意：注释可以出现在代码的任意位置，但是不能分隔关键字和标识符。

使用“//”标记可以注释单行代码，使用“///”标记也可以注释单行代码，而且“///”标记单行注释的使用方法几乎与“//”标记相同。如果在某一行中出现“///”标记，那么“///”标记所在行后面的内容均为注释内容。但是“///”标记很少用于单行代码的注释，“///”标记一般用于为代码段添加说明。

可以在<summary>和</summary>之间添加方法的描述，在<param>和</param>之间添加参数的描述，在<returns>和</returns>之间添加返回值的描述。

【例2.8】在“欢迎您来到C#语言世界！”程序中使用“///”注释，实现代码如下。

```
01   namespace ConsoleApplication1
02   {
03       ///<summary>
04       ///定义Program类
05       ///</summary>
06       class Program
07       {
08           ///<summary>
```

```
09          ///程序入口方法
10          ///</summary>
11          ///<param name="args">用于向方法传递的参数</param>
12          ///<returns>方法无返回值</returns>
13          static void Main(string[] args)
14          {
15              Console.WriteLine("欢迎您来到C#语言世界！");
16              Console.ReadLine();
17          }
18      }
19  }
```

技巧：“///”标记不仅可以为代码段添加说明，它还有一项更重要的工作，就是用于生成自动文档。自动文档一般用于描述项目，使项目更加清晰直观。在***Visual Studio 2010***中可以通过设置项目属性来生成自动文档。具体的步骤如下：在“解决方案资源管理器”对话框中右击项目，在弹出的快捷菜单中选择“属性”命令，再在弹出的对话框中选择“生成”选项卡，然后在“输出”栏中选中“***XML***文档文件”复选框，最后重新调试程序，便可以生成***XML***文档。生成的***XML***文档的默认存储路径为当前程序的***bin\Debug***文件夹下。

在***Visual Studio 2010***开发环境中，可以使用***#region***和***#endregion***关键字指定可展开或可折叠的代码块，这样可以使程序更好的布局。

【例2.9】在“欢迎您来到***C#***语言世界！”程序中使用“***#region***”和“***#endregion***”注释，实现代码如下。

```
01  #region
02  static void Main(string[] args)
03  {
04      System.Console.WriteLine("欢迎您来到C#语言世界！");    //输出信息
05      System.Console.ReadLine();                            //等待读入信息
06  }
07  #endregion
```

上机练习

上机练习4　使用“///”为代码段添加说明

使用“///”标记为代码段添加说明，如图2.9所示。

图2.9　使用“///”为代码段添加说明

2.4　术语

（1）命名空间既用做程序的“ 内部 ”组织系统，也用做向“ 外部 ”公开的组织系统（一种向其他程序公开自己拥有的程序元素的方法）。

（2）类是一种数据结构，它可以封装数据成员、函数成员和其他的类。

（3）标识符是指在程序中用来表示事物的单词。

（4）关键字是指在***C#***语言中具有特殊意义的单词，它们被***C#***设定为保留字，不能随意使用。

2.5　小结

本章通过一个简单的***C#***程序对***C#***程序的结构进行了详细介绍，让读者对***C#***程序的组成结构有一个概括性的认识。在***C#***程序的组成结构中，读者需要重点掌握命名空间、类以及***C#***语句。命名空间在***C#***程序中占有重要的地位，通过引入命名空间，可以将空间下的成员引入到当前的编译单元中；而类是***C#***语言的核心和基本构成模块，程序员可以通过编写类来描述实际开发需要解决的问题。另外，本章还介绍了几种常用的代码注释方法，读者可以根据实际情况为自己的代码添加注释，以方便后期的阅读和维护。

2.6　练习

1. 创建一个控制台应用程序，输出“ *^__^* ”笑脸图案。

2. 创建一个控制台应用程序，然后在该应用程序中添加一个类文件（****.class***文件），并且要求在该类文件中编写的类与控制台应用程序默认的命名空间不一致，最后在***Main***方法中引用并实例化这个自定义类。

3. 对上题应用程序的***Main***方法使用“ /// ”标记进行整体注释，并对自定义的类使用“ // ”标记进行逐行注释。

4. 编写***C#***程序，除了使用***Visual Studio 2010***等开发工具外，还可以使用***Windows***系统自带的记事本来编写，编写后，使用***C#***语言编译工具***csc.exe***对其进行编译即可。本练习要求在记事本中编写“ 例2.1 ”中的代码，然后使用***csc.exe***编译工具对其进行编译，编译命令的格式可参考“ ***csc /out:C:***\编译文件***.exe C:***\源代码文件***.txt*** ”。