Chapter 3

第3章 C#语言基础

本章导学

- C#语言概述
- C#语言程序的结构
  - ✓ 命名空间
  - ✓ 类和程序入口
  - ✓ 输入输出
  - ✓ 注释
  - ✓ 编译和运行程序
- 变量和常量
- C#数据类型
  - ✓ 值类型
    - 简单类型
    - 结构类型
    - 枚举类型
  - ✓ 引用类型
    - 类类型
    - 数组类型
    - 接口类型
    - 代表类型
  - ✓ 类型转换
- C#语句
  - ✓ 运算符
  - ✓ 条件语句
  - ✓ 循环语句
- C#面向对象编程
- 独立实践

3.1 C# 语言概述

C#是微软公司专门为.NET 量身定做的编程语言,它与.NET 有着密不可分的关系。C# 的类型就是.NET 框架所提供的类型,C#本身并无类库,而是直接使用.NET

框架所提供的类库。另外，类型安全检查、结构化异常处理也都是交给 CLR 处理的。因此，C# 是最适合开发. NET 应用的编程语言。

C# 不仅具有 C++ 的强大功能，而且具有 Visual Basic 简单易用的特性。C# 的语法与 C++ 和 Java 基本相似。如果以前对 C++ 或 Java 比较熟悉的话，学习 C# 则是一件非常容易的事情。在默认情况下，C# 代码在. NET 框架提供的受控环境下运行，不允许直接操作内存。它带来最大的变化是 C# 没有了 C 和 C++ 中复杂的指针。与此相关的，那些在 C++ 中被大量使用的指针运算符已经不再出现，C# 只支持一个“.”运算符。

C# 具有面向对象编程语言所应有的一切特性，如封装、继承和多态，在 C# 的类型系统中，每种类型都可以看做一个对象，和 Java 一样，C# 只允许单继承。即一个类不会有多个基类，从而避免了类型定义的混乱。

C# 没有了全局函数、全局变量和全局常量，所有都必须封装在一个类中，体现了“一切都是对象”的完全面向对象的特色。因此，用 C# 编写的代码具有更好的可读性，而且减少了发生命名冲突的可能。

3.2　C# 语言程序的结构

我们还是从最简单、最经典的 HelloWorld 程序开始学习，利用这个程序来说明 C# 程序的结构和 C# 编译器的使用。

```
/*
案例名称：第一个 C#程序
程序清单：3-01.cs
*/
using System;                                    //引入命名空间
class Hello                                      //定义类
{
    /*以下为程序入口*/
    public static void Main()
    {
        Console.WriteLine("您好！");
        Console.ReadLine();
    }
}
```

可以在任意一种编辑软件中完成上述代码的编写，然后把文件存盘，文件名为 3-01. cs。典型的 C# 源文件以. cs 作为文件的扩展名。

3.2.1　命名空间

程序清单中的 using System 表示导入命名空间，高级语言总是依赖于许多系统预定义的元素。C 或 C++ 的程序员一定对使用 #include 之类的语句来导入其他 C 或 C++ 源

文件非常熟悉。C# 中的含义与此类似，用于导入预定义的元素，这样在自己的程序中就可以自由地使用这些元素。

如果没有导入命名空间，程序还能保持正确吗？答案是肯定的。那样的话，必须把代码改写成下面的样子：

```
/*
案例名称：不导入命名空间的 C#程序
程序清单：3-02.cs
*/
class Hello                                                    //定义类
{
    /*以下为程序入口*/
    public static void Main()
    {
        System.Console.WriteLine("您好！");
        System.Console.ReadLine();
    }
}
```

也就是说，在 Console 前加上一个前缀“System.”，这个小圆点“.”表示 Console 是作为 System 的成员而存在的。C# 中抛弃了 C 和 C++ 中繁杂且极易出错的运算符像“::”和“->”等，C# 中的复合名字一律通过“.”运算符来连接。

3.2.2 类和程序入口

在 C# 语言中，每个元素都必须属于一个类。如果是一个 C 或 C++ 的程序员，那么请暂时忘掉那些全局变量和全局函数。

在上面的程序清单中，class Hello 声明了一个类，类的名字叫做 Hello。我们所做的事情就是依靠它来完成的。和 C、C++ 中一样，源代码块被包含在一对大括号“{”和“}”中。每一个右括号“}”总是和它前面离它最近的一个左括号“{”相配套，如果左括号和右括号没有全部配套，那程序就是一个有语法错误的程序。

public static void Main()表示类 Hello 中的一个方法，方法总是为我们完成某件工作的。在 C# 程序中，Main()方法是程序的入口，程序的执行总是从 Main()方法开始的。一个程序中不允许出现两个或两个以上的 Main()方法。对于习惯了写 C 控制台程序的读者请牢记 C# 中 Main()方法必须被包含在一个类中。Main()方法的返回类型可以是 void(表示没有返回值)或 int(返回代表应用程序错误级别的整数)。

3.2.3 输入输出

程序所完成的输入输出功能都是通过 Console 类来完成的。Console 是在命名空间 System 中已经定义好的一个类。上面的代码中类 Console 为我们展现了两个最基本的方法 WriteLine 和 ReadLine 的用法，Console.ReadLine 表示接受输入设备(键盘)的输

入，Console. WriteLine 则用于在输出设备(屏幕)上输出。

3.2.4　注释

应用程序并不只是写给自己看的，在程序维护过程中，源代码需要广泛的交流，养成良好的代码注释的习惯是一名优秀的程序员必备的条件之一。代码注释不会浪费您的编程时间，相反它会提高编程效率，使程序更加清晰完整。

C#注释的方式和 C++ 没有太大的区别，每一行中双斜杠“//”后面的内容以及在“/*”和“*/”之间的内容都将被编译器忽略。这样就可以采用“//”进行单行注释，采用“/*”和“*/”进行多行注释。

3.2.5　编译和运行程序

.NET Framework SDK 内置了 C#编译器 csc. exe(C Sharp Compiler)，该文件在“\Windows安装目录\Microsoft. NET\Framework\v 版本号”目录下，例如，Windows XP 系统下的目录就是 C:\WINDOWS\Microsoft. NET\Framework\v1. 1. 4322。如果不能执行该命令，需要将该目录添加到操作系统的 PATH 变量中。然后可以在命令窗口编译 C#文件，假如要编译名为 3-01. cs 的 C#文件，命令行的语法为：

```
csc 3-01.cs
```

如果成功编译，就可以得到可执行文件 3-01. exe 文件。运行该文件输出相应的字符串，如图 3-1 所示。

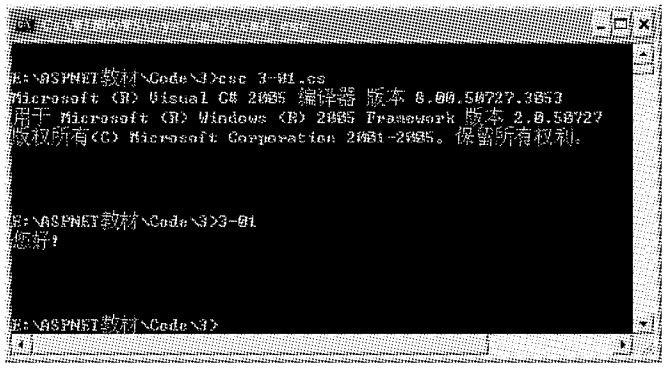

图 3-1　命令行编译和运行 C#程序

这样编译和运行每一个程序很不方便。前面介绍过主要是在 UltraEdit 中编辑 C#程序，那么如果能在 UltraEdit 的环境中编译和运行 C#程序就方便多了。通过配置，UltraEdit 可以提供这样的功能。

选择 UltraEdit 的“高级”→“工具栏配置”命令，打开“工具配置”对话框。“命令行”输入“csc %n%e”；“工作目录”输入“%p”；“菜单项名称”随意。然后将“保存活动文件”、“输出到列表窗口”、“捕捉输出”这 3 个选项都选上，如图 3-2 所示。最后单击“插入”按

钮。这样就在 UltraEdit 的“高级”菜单下增加一个菜单项，如图 3-3 所示。

图 3-2　在 UltraEdit 中编译 C# 程序的工具栏配置

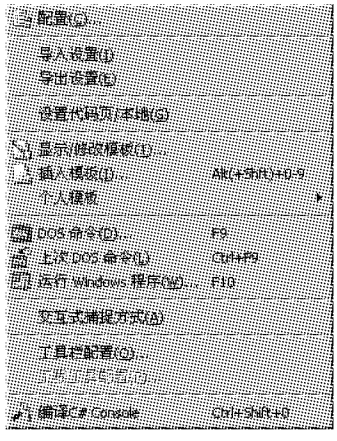

图 3-3　配置后的菜单项

以后，在编辑 C# 程序的过程中只要选择“编译 C# Console”命令，就可以编译正在编辑的 C# 程序，如图 3-4 所示。

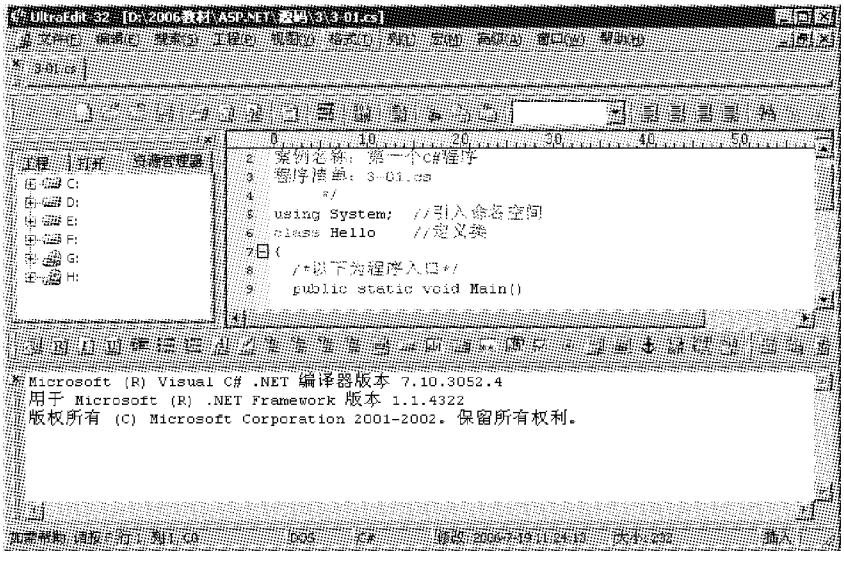

图 3-4　在 UltraEdit 中编译 C# 程序

以同样的方法增加另一个菜单项“运行 C# Console”，进行如图 3-5 所示的配置，就可以在 UltraEdit 的环境中运行 C# 程序了。

以后，在编辑 C# 程序的过程中选择“编译 C# Console”命令编译正在编辑的 C# 程序后，再选择“运行 C# Console”命令就可以运行了，如图 3-6 所示。

图 3-5　在 UltraEdit 中运行 C#程序的配置

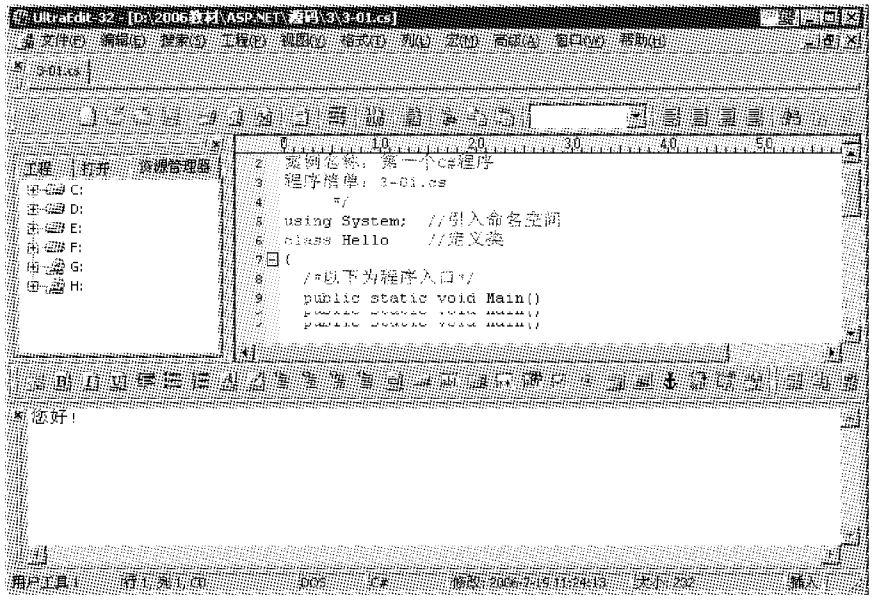

图 3-6　在 UltraEdit 中运行 C#程序

3.3　变量和常量

C#是大小写敏感的，即大写字母和小写字母被认为是不同的字母。例如，变量名 something 和 Something，SOMETHING 是不同的变量。命名变量名要遵守如下的规则。

(1) 不能是 C# 关键字。

(2) 第一个字符必须是字母或下划线,不要太长,一般以不超过 31 个字符为宜;不能以数字开头。

(3) 中间不能有空格。

(4) 变量名中不能包含特殊符号。实际上,变量名中除了能使用 26 个英文大小写字母和数字外,只能使用下划线“_”。

(5) 变量名不要与 C# 中的系统函数名、类名或对象名相同。

变量通常应具有描述性的名称。例如,看到 numberOfStudents 这个变量名,就知道它表示学生人数,这样的命名方式使程序可读性和维护性大大提高。除了循环变量或临时变量之外,尽量不要使用单字母的变量名如 x、y、z、i、j、k 等。变量命名的方式决定了程序书写的风格,在整个程序中保持一致的、良好的可读性风格是很重要的。

变量有两种典型的命名方法:Camel 表示法和 Pascal 表示法。Camel 表示法以小写字母开头,以后的单词都以大写字母开头。如 myBook、theBoy、numOfStudent 等。Pascal 表示法则是每个单词都以大写字母开头,如 MyBook、NumOfStu 等。一般来说,局部变量、函数参数采用 Camel 表示法,全局变量、类型名、函数名采用 Pascal 表示法。

在程序中,如果想让变量的内容初始化后一直保持不变,可以定义一个常量。例如,在圆面积计算中经常要用常数 π,可以通过命名一个容易理解和记忆的名字来改进程序的可读性,同时在定义中加关键字 const,将它规定为常量性质,可以预防程序出错。常量的命名方式一般是单词的每个字母都大写。

下面的实例 3-03.cs 说明了变量和常量的使用。

```
/*
案例名称:变量和常量的使用
程序清单:3-03.cs
*/
using System;
class Sample                                    //类名采用 Pascal 大写
{
  public static void Main()
  {
      const double PI=3.14159;                  //定义常量
      double radiusOfRound=8.5;                 //定义变量圆半径并赋值,Camel 大写
      double areaOfRound;                       //定义变量圆面积,Camel 大写
      areaOfRound=radiusOfRound*PI;             //计算圆面积
      Console.WriteLine("圆的面积为:{0} ",areaOfRound);
  }
}
```

运行结果如图 3-7 所示。

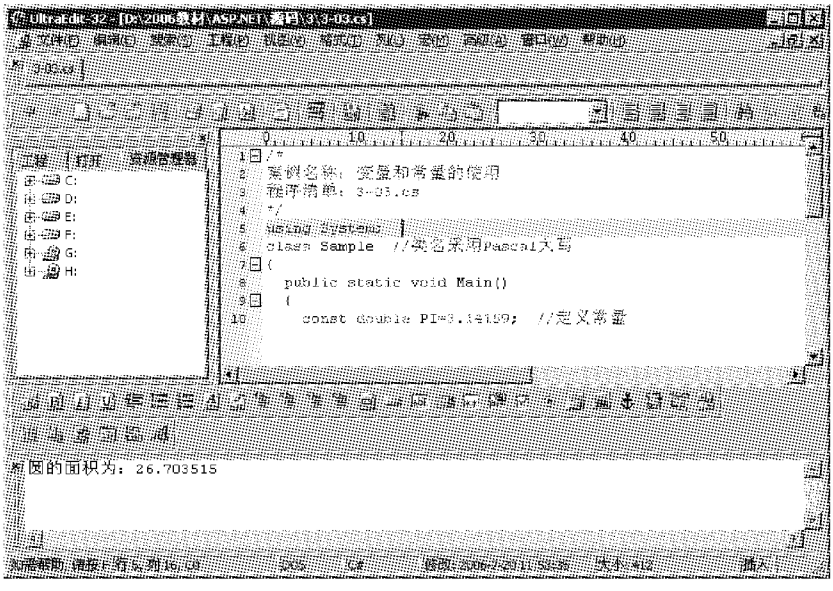

图 3-7　变量和常量的使用

3.4　C# 数据类型

C#的数据类型分为值类型(Value Type)和引用类型(Reference Type)两大类。值类型包括绝大多数简单类型(Simple Type)、结构类型(Struct Type)和枚举类型(Enum Type)3 种。引用类型包括类类型(Class Type)、数组类型(Array Type)和代表类型(Delegate Type)等。

这两种类型的区别在于：值类型存储的是自身的值，而引用类型存储的是对值的引用。例如，一个 int 类型的变量保存对应的实际值，一个 string 类型的变量本身并不包含一个字符串，而表示对存储器中某一个字符串的引用(可以说是该字符串在内存中的地址)。

3.4.1　值类型

值类型，也称为简单类型，是直接由一系列元素构成的数据类型。C#语言中提供了一组已经定义好的简单类型，可以分为整数类型、实数类型、布尔类型、字符类型、结构类型和枚举类型。

1. 整数类型

整数类型的变量的值为整数。C#中有 9 种整数类型，这些整数类型在数学上的表示以及在计算机中的取值范围如表 3-1 所示。

当某种类型存储的值超出所能表示的范围时，就会出现溢出情况，但是程序并不会报错，结果会呈现不正确的值。

表 3-1 整数类型

整数类型	特　征	取值类型
sbyte	有符号 8 位整数	−128～127
byte	无符号 8 位整数	0～255
short	有符号 16 位整数	−32768～32767
ushort	无符号 16 位整数	0～65535
int	有符号 32 位整数	−2147483648～2147483647
uint	无符号 32 位整数	0～4294967295
long	有符号 64 位整数	−9223372036854775808～9223372036854775807
ulong	无符号 64 位整数	0～18446744073709551615

2. 实数类型

实数在 C# 中采用两种数据类型来表示：单精度(float)和双精度(double)。它们的区别在于取值范围和精度不同。单精度数的取值范围为 $\pm 1.5\times10^{-45}$～3.4×10^{38}，精度为 7 位。双精度数的取值范围为 $\pm 5.0\times10^{-324}$～1.7×10^{308} 之间，精度为 15～16 位。C# 还专门定义了一种十进制类型(decimal)，主要用于方便做金融和货币方面的计算。在现代的企业应用程序中，不可避免要进行大量的这方面的计算和处理。十进制类型是一种高精度、128 位的数据类型，它所表示的范围为 1.0×10^{-28}～7.9×10^{28}，有效数字为 28～29 位。当定义一个变量并赋值给它的时候，使用 m 后缀来表明它是一个 decimal 型。例如：

```
decimal cur=100.0m
```

如果省略了 m，则变量被赋值之前将被编译器认作 double 型。

3. 布尔类型

布尔类型是用来表示“真”和“假”的。布尔类型表示的逻辑变量只有两种取值，在 C# 中，采用 true 和 false 两个值来表示。

在 C 语言中，用 0 来表示“假”，其他任何非 0 的值表示“真”。在 C# 中，布尔型变量只能是 true 或者 false。这也使得 C# 语言的类型系统更加健壮，避免很多不必要的歧义。

4. 字符类型

字符包括数字字符、英文字母和表达符号等，C# 提供的字符类型按照国际标准，采用 Unicode 字符集。一个 Unicode 的标准字符长度为 16 位，用它可以来表示世界上大多数语言。给一个变量赋值的语法为：

```
char mychar='M';
```

也可以直接通过十六进制或者 Unicode 赋值。例如：

```
char mychar='\x0034';                    //mychar='4'
char mychar='\u0039';                    //mychar='9'
```

5. 结构类型

在程序设计中，经常把一组相关的信息放在一起，把一系列相关的变量组成一个单一的实体，这个单一的实体类型叫做结构。结构类型采用关键字 struct 来进行声明。C# 的结构类型和 C++ 不同，C++ 的结构其实和类的实质是一样的，是引用类型。而 C# 的结构类型是值类型，但它和 C 语言的结构类型又不一样。C# 的结构类型可以包含常量、变量、方法、构造函数、属性、索引器等。下面用一个实例来说明结构类型的用法。

```
/*
案例名称：结构类型的使用
程序清单：3-04.cs
*/
using System;

struct Student                           //定义结构类型 Student
{
    public string name;                  //结构成员
    public int age;
    public string sex;
}

class TestOfStruct                       //主类
{
    public static void Main()
    {
        Student st;                      //声明结构变量
        st.name="张三";                  //为结构变量各成员赋值
        st.age=20;
        st.sex="男";
        //下面打印结构变量各成员的值
        Console.WriteLine("姓名：{0},年龄：{1},性别：{2}",st.name,st.age,st.sex);
    }
}
```

程序在 UltraEdit 中的运行结果如图 3-8 所示。

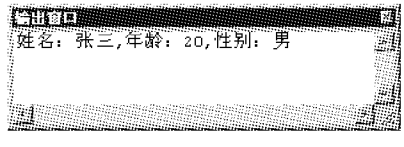

图 3-8　程序 3-04.cs 的运行结果

6. 枚举类型

枚举(Enum)为一组在逻辑上密不可分的整数值提供便于记忆的符号，枚举类型可以使程序的可读性增强，便于维护。程序 3-05.cs 描述了枚举类型的用法。