新起点电脑教程

第 1 章

C#语言基础

本章要点

- C#语言介绍
- 史上最强开发工具——Microsoft Visual Studio
- 初识 Microsoft Visual Studio 开发环境

本章主要内容

C#语言是当前最热门的编程技术之一，它综合了 C、C++和 Java 等语言的优点，引领了编程行业的开创性的变革。C#功能强大，不但能够开发 Windows 窗体程序，而且也能开发 Web 程序和移动应用程序。本章将引导读者进入 C#世界，使读者逐渐掌握 C#语言的基本知识，并掌握搭建 C#开发环境的方法和 Microsoft Visual Studio 2017 可视化集成开发环境的基本知识。

1.1 C#语言介绍

C#读作 C Sharp，是从 C 和 C++进化而来的新一代的编程语言，是微软公司的一款杰出开发语言。本节将简要介绍 C#语言的基本知识，让读者充分认识 C#这门功能强大的编程语言。

↑扫码看视频

1.1.1 C#语言的特点

1. 简单易学

和 C++相比，C#的语法更加简单。在 C#中舍弃了指针等烦琐的内容，这样有助于初学者快速地掌握核心语法等内容。

2. 面向对象

C#支持所有关键的面向对象的概念，例如封装、继承和多态性。面向对象是高级编程语言的最大特点，和面向过程的语言相比更简单，也更容易来描述现实世界。

3. 类型安全

在使用过程中必须遵守 C#的一些相关变量规则，因为 C#使用了最严格的类型安全来保护自己及垃圾收集器，所以不能使用没有初始化的变量，取消了不安全的类型转换，实施边界检查。

4. 兼容

C#允许使用 NGWS 运行时环境的通用语言规定来访问不同的 API。在 CLS(语言规范)中规定了一个标准，能够使符合这种标准的语言的内部之间进行操作。为了加强 CLS 的编译，C#编译器检测所有的公共出口编译，并在通不过时列出错误。

5. 最好的开发工具

微软公司为 C#语言提供了史上最强的可视化编程工具 Visual Studio .NET，可以帮助开发者更加高效地开发出软件程序。

1.1.2 .NET Framework 框架

微软在 2000 年推出了.NET Framework 框架，微软公司将其作为重新树立自己在软件业

界的信心和地位的崭新战略。通过.NET 平台，可以使用多种语言开发同一个项目，实现了这些语言的跨平台应用。微软为.NET 专门配备了一门新的语言，这便是 C#语言。

.NET Framework 是微软为开发应用程序而创建的一个平台，利用它可以开发 Windows 桌面应用程序、Web 应用程序、Web 服务以及其他类型的应用程序。.NET Framework 的设计方式确保它可以用于各种语言，包括 C#、C++、Visual Basic 和 JavaScript 等。正因如此，微软公司还推出了不同语言的.NET 版本，所有的这些语言都可以访问.NET Framework，它们彼此之间还可以通信，例如，C#开发人员可以使用 Visual Basic 程序员写的代码，反之亦然。

.NET Framework 包含一个非常大的代码库，可以在客户语言(如 C#)中通过面向对象编程技术来使用这些代码。这个代码库分为不同的模块，这样就可以根据希望得到的结果来选择使用其中的各个部分。使用.NET Framework 编写应用程序，就是使用.NET 代码库来编写代码(使用支持 Framework 的任何一种语言)。

为了能够执行 C#程序代码，必须将其转换为目标操作系统能够理解的语言，即本机代码。这种转换称之为编译代码，由编译器执行。但是，在.NET Framework 框架下，该过程包含两个阶段：CIL 和 JIT。在编译使用.NET Framework 库的代码时，不会立即创建用于操作系统的本机代码，而是把代码编译为通用中间语言(Common Intermediate Language, CIL)代码，这些代码并非能用于任何一种操作系统，也并非专门用于 C#。对于其他的.NET 语言，例如 Visual Basic.NET 也可以在第一阶段编译为这种语言。而在开发 C#应用程序时，这个编译步骤由 Visual Studio .NET 来完成。很显然，要想执行应用程序，必须要完成更多的工作，这就是 JIT(Just In Time)编译器的任务，它把 CIL 编译为专用于 OS(操作系统)和目标机器结构的本机代码。这样 OS 才能执行应用程序。这里编译器的名称 Just In Time(准时生产)反映了 CIL 代码仅在需要时才编译的事实。C#只是用于.NET 开发的一种语言，但它是最好的语言之一。C#的优点是：它是唯一彻头彻尾为.NET Framework 设计的语言，是可以到其他操作系统上的.NET 版本中使用的主要语言。

知识精讲

学习程序开发之路是充满挑战之路，枯燥的代码和烦琐的调试有时会使你感觉到无味；但这同时也是充满乐趣之路，每一个功能的调试成功会使你充满自豪和成就感。学习 C#也是如此，学习路上肯定会既充满挑战也充满乐趣。作为一名初学者，我们该怎样学好 C#呢？下面给出几点建议。

(1) 培养兴趣。

兴趣是我们学习任何知识的动力，在现实中，往往我们会对喜欢的事情充满热情，也乐于耗费精力。对于编程来说，只要你喜欢感受那调试成功的喜悦，就说明你已经对编程产生了兴趣。另外，调试成功的喜悦会让你更加喜欢编程，更能带给你成就感。

(2) 多看代码，多实践。

当有一定的语法基础以后，一定要多看别人的代码，看的目的是掌握程序的结构和流程，看完之后需要自己亲自实践。程序开发讲究精细，哪怕是一个标点的错误都不会调试成功。有人说学习编程的秘诀是“编程、编程、再编程，练习、练习、再练习”，这就充分说明了实践的重要性。

1.2 史上最强开发工具——Microsoft Visual Studio

Microsoft Visual Studio 2017 是微软推出的全新专用开发工具，它是一个集成的开发环境工具。Microsoft Visual Studio 2017 是一款功能齐全且可扩展的免费 IDE，能够创建适用于 Windows、Android 和 iOS 的应用程序。在本节将详细讲解 Microsoft Visual Studio 2017 工具的基本知识。

↑扫码看视频

1.2.1 Visual Studio 2017 的新功能

- 代码导航：Visual Studio 2017 极大地改善了代码导航功能，并对结果进行了着色，提供了自定义分组、排序、过滤和搜索功能。通过使用强大的 Go to All 功能，可以对解决方案中的任何文件、类型、成员或符号声明实现快速、完整的搜索。
- 写入和读取代码：除了导航，开发者花了很多时间在写入和读取代码上，Visual Studio 2017 侧重于促进编写正确的代码，以及维持开发人员的代码的可读性。在 Visual Studio 2015 的基础上，Visual Studio 2017 的智能感知更加强大，更加注重重构和代码修复，可自定义代码风格的配置和执行。
- 测试代码：Visual Studio 2017 包括 C#和 Visual Basic 的动态单元测试。动态单元测试(Live Unit Testing)可以在运行时分析数据，在编辑后仅测试运行受影响部分，并通过编辑器中测试的状态提供即时反馈。
- 调试代码：当所有方法都失效后，开发者依靠调试可以帮助他们确定问题的来源。Visual Studio 2017 可以大大节省时间和动作，如通过单步执行程序，可定位到异常信息。

1.2.2 Visual Studio 2017 的版本

(1) 企业版：能够提供点对点的解决方案，充分满足正规企业的要求。这是功能最为强大的版本，价格最高。

(2) 专业版：提供专业的开发者工具、服务和订阅。功能强大，价格适中，适合于专业用户和小开发团体。

(3) 社区版：提供全功能的 IDE，完全免费，适应于一般开发者和学生。

1.2.3　安装 Microsoft Visual Studio 2017

(1) 登录网址：https://www.visualstudio.com/zh-hans/，如图 1-1 所示。

(2) 单击“下载 Visual Studio”下的 Enterprise 2017 链接开始下载，如图 1-2 所示。下载后得到一个 exe 格式的安装文件 vs_enterprise__2050403917.1499848758.exe，如图 1-3 所示。

图 1-1　微软 Visual Studio 官网

图 1-2　Enterprise 2017 链接

图 1-3　可安装文件

(3) 鼠标右击下载文件 vs_enterprise__2050403917.1499848758.exe，选择“使用管理员模式进行安装”命令，在弹出的界面中单击“继续”按钮，这表示同意了许可条款，如图 1-4 所示。

图 1-4　单击“继续”按钮

(4) 在弹出的“正在安装”界面选择要安装的模块，本书内容需要安装如下所示的模块。

- 通用 Windows 平台开发。
- .NET 桌面开发。

- ASP.NET 和 Web 开发。
- 数据存储和处理。
- 使用.NET 的移动开发。

上述各模块的具体功能在本界面中也进行了详细说明，如图 1-5 所示。在左下角可以设置安装路径，单击“安装”按钮后开始进行安装。

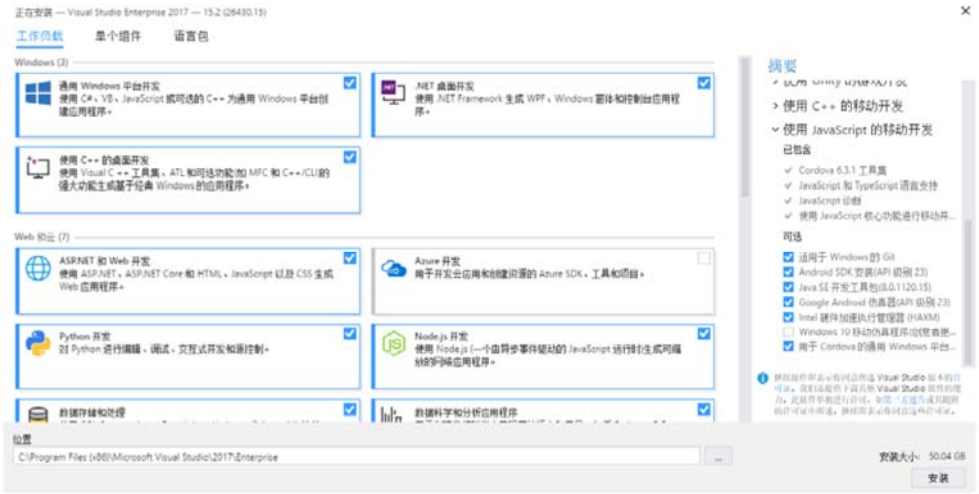


图 1-5 “正在安装”界面

(5) 单击“安装”按钮后弹出安装进度界面，这个过程比较耗费时间，读者需要耐心等待，如图 1-6 所示。

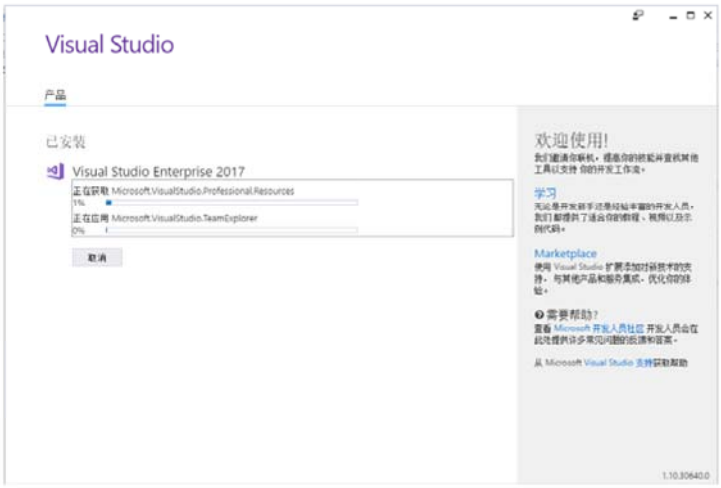


图 1-6 安装进度界面

(6) 安装成功后的界面效果如图 1-7 所示。

(7) 选择“开始”|“所有应用”|Visual Studio 2017 命令，就可运行我们刚安装的 Visual Studio 2017，如图 1-8 所示。

图 1-7　安装成功

图 1-8　Visual Studio 2017 图标

知识精讲

C#和 Java、C++的关系

C#从 Java 中继承了它的大多数特点，包括使用语法和范围等。例如最基本的 类”，在 C#中类的声明方式和在 Java 中很相似。Java 的关键字 import 在 C#中被替换成了 using，但是它们起到了同样的作用，并且一个类开始执行的起点是静态方法 Main()。

另外，C#也从 C 和 C++中继承一些特点，这主要体现在如下 3 个方面。

- 编译：程序直接编译成标准的二进制可执行形式，但 C#的源程序并不是被编译成二进制可执行形式，而是一种中间语言，类似于 Java 字节码。例如一个名为 Hello.cs 的程序文件，它将被编译成命名 Hello.exe 的可执行程序。
- 结构体：一个 C#的结构体与 C++的结构体是相似的，因为它能够包含数据声明和方法。
- 预编译：存在预编译指令，支持条件编译、警告、错误报告和编译行控制。

1.3　初识 Microsoft Visual Studio 开发环境

在成功安装 Microsoft Visual Studio 2017 后，需要对其进行专门设置，才能使其符合自己的开发需求。本节将简要介绍 Microsoft Visual Studio 2017 的设置方法，并对其开发环境进行说明，为读者步入本书后面知识的学习打下基础。

↑扫码看视频

1.3.1　设置工作

(1) 首次打开安装后的 Microsoft Visual Studio 2017，将弹出“以熟悉的环境启动”界面。因为本书讲解的是 C#开发，所以选择 Visual C#选项，如图 1-9 所示。然后单击“启动

Visual Studio”按钮开始配置。

(2) 配置完成后将来到 Microsoft Visual Studio 2017 的集成开发界面，如图 1-10 所示。

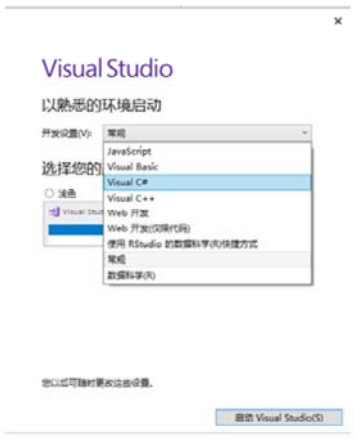


图 1-9 “以熟悉的环境启动”界面

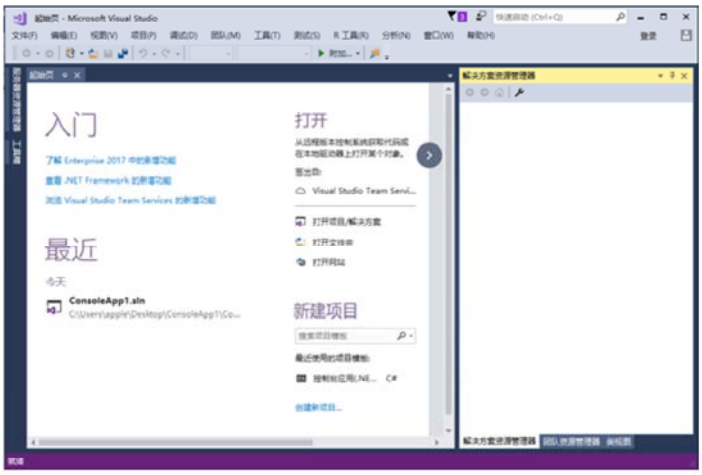


图 1-10 Visual Studio 2017 默认集成开发界面

1.3.2 新建项目

通过 Microsoft Visual Studio 2017，可以迅速创建一个项目，包括 Windows 应用程序、控制台程序和 Web 应用程序等常用项目。方法是选择“文件”|“新建”|“项目”命令，弹出“新建项目”对话框，在此可以设置项目的类型，如图 1-11 所示。

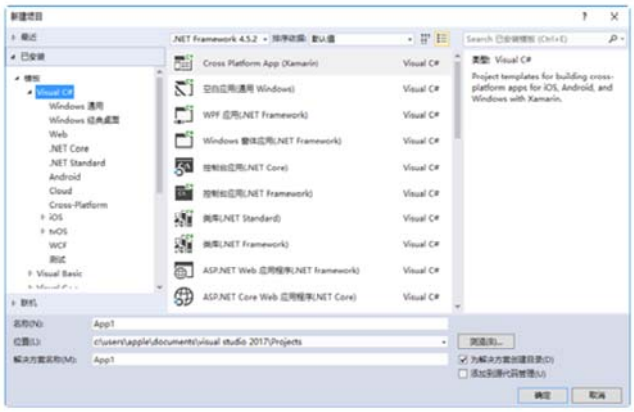


图 1-11 “新建项目”对话框

选择“文件”|“新建”|“网站”命令，弹出“新建网站”对话框，在此可以迅速创建一个不同模板类型的网站项目，如图 1-12 所示。

选择“文件”|“新建”|“文件”命令，弹出“新建文件”对话框，在此可以创建一个不同模板类型的文件，如图 1-13 所示。

如果在 Microsoft Visual Studio 2017 中创建一个项目，可以自动生成必需格式的代码。例如新建一个控制台项目后，将在项目文件内自动生成必需格式的代码，并且在右侧的“解

决方案资源管理器”面板中显示自动生成的项目文件，如图 1-14 所示。

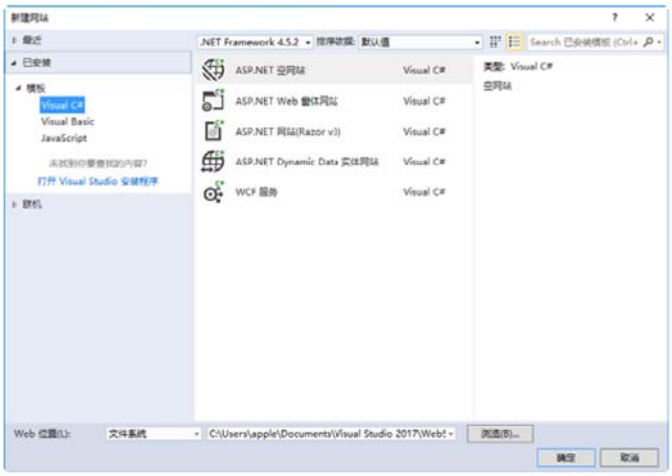


图 1-12　“新建网站”对话框

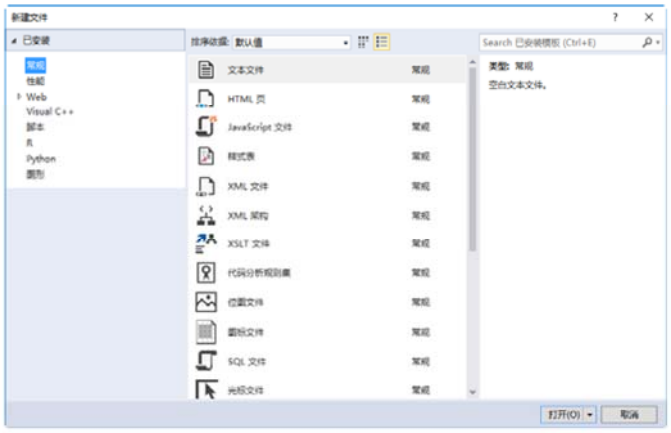


图 1-13　“新建文件”对话框

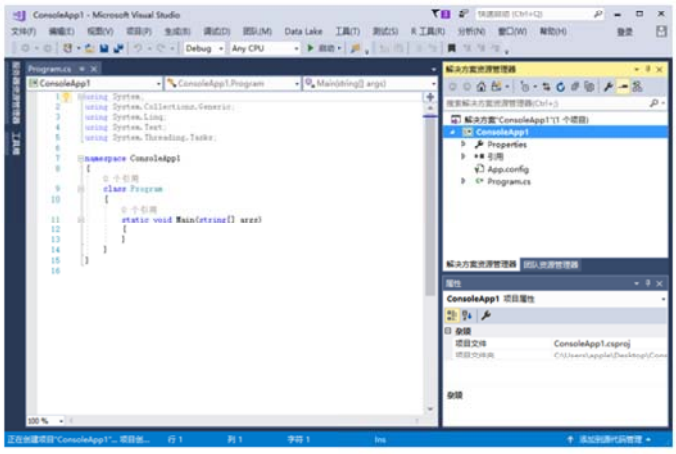


图 1-14　自动生成的代码和文件

1.3.3 解决方案和类视图

解决方案和类视图是 Microsoft Visual Studio 2017 的重要组成工具，通过它们可以更加灵活地对项目进行控制和管理。下面将对 Microsoft Visual Studio 2017 解决方案和类视图的基本知识进行简要介绍。

1. 解决方案

当创建一个项目后，会在“解决方案资源管理器”面板中显示自动生成的项目文件。解决方案中包含一个或多个项目，每个项目都对应于软件中的一个模块。在资源管理器中，Visual Studio 2017 将同类的文件放在一个目录下，当单击这个目录后，会将对应目录下的文件全部显示出来。例如，单击“引用”目录后，则将引用的程序集显示出来，如图 1-15 所示。

右击“解决方案资源管理器”面板中的每个节点，都将弹出一个上下文菜单，通过其中的菜单命令可以对节点对象进行操作。例如，右击项目名并依次选择“添加”|“新建项目”命令，可以在项目内添加一个新的项目文件，如图 1-16 所示。

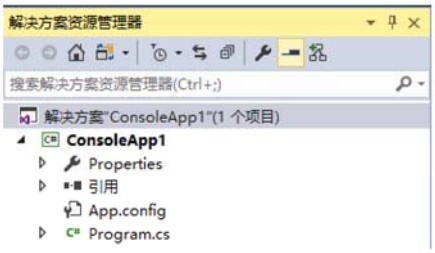


图 1-15 “引用”目录的程序集

图 1-16 新建一个项目

2. 类视图

资源管理器是以文件为主角的项目管理，而 C#是一种面向对象的编程语言，其基本的对象编程单位是类。为此，Microsoft Visual Studio 2017 提供了类视图来进行项目对象的管理。

选择“视图”|“类视图”命令，将在资源管理器中显示当前项目内的所有类对象，如图 1-17 所示。

在图 1-17 中，上方显示项目的命名空间、基类和各种子类，具体说明如下。

- 符号{}：表示命名空间。
- 符号：表示基类。
- 符号：表示普通类或子类。

在如图 1-17 所示的类视图中选中一个类的类型，然后单击右键，将弹出一系列和类相关的操作命令，如图 1-18 所示。例如，选择“查看类图”命令，可以查看这个类的关系图

结构，并且可以在 Microsoft Visual Studio 2017 的底部窗口查看这个类的详细信息，如图 1-19 所示。

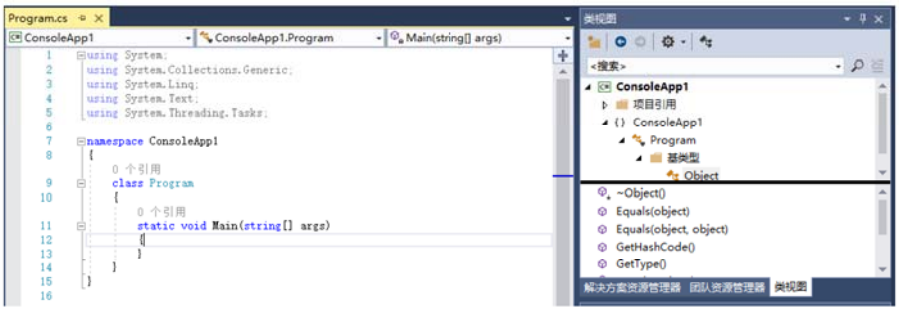


图 1-17　项目类视图

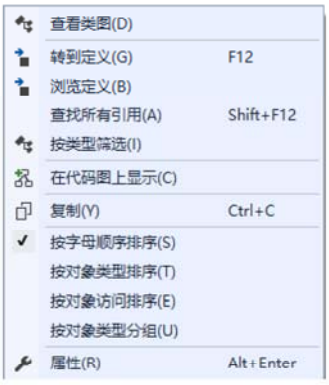


图 1-18　类操作命令

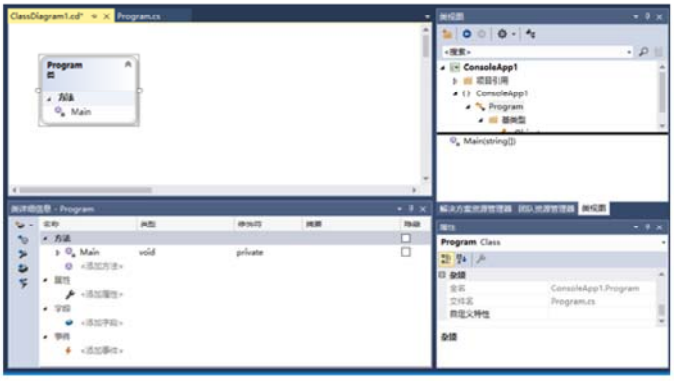


图 1-19　类关系结构和详细信息

1.3.4　文本编辑器

双击“解决方案资源管理器”面板中的文件名，就可以查看这个文件的源代码。如果在 Microsoft Visual Studio 2017 中打开多个项目文件，会在栏目内显示多个文件的文件名，如图 1-20 所示。

图 1-20　文件名栏

Microsoft Visual Studio 2017 文本编辑器主要有如下几个特点。

1. 用不同的颜色显示不同的语法代码

在 Microsoft Visual Studio 2017 文本编辑器中，使用蓝色显示 C#的关键字，用青色来显示类名。

2. 代码段落格式自动调整

Microsoft Visual Studio 2017 中的文件源代码段落会自动缩进，这样可以加深代码对用户的视觉冲击。如图 1-21 所示的就是段落缩进的代码格式。

图 1-21　源代码段落缩进

3. 语法提示

当用户使用文本编辑器进行代码编写时，编辑器能够根据用户的输入代码来提供对应的语法格式和关键字。例如在图 1-21 所示的代码界面输入字符“na”后，编辑器将自动弹出对应的提示字符，如图 1-22 所示。

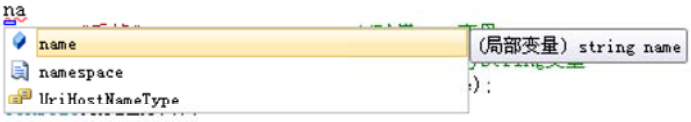


图 1-22　语法提示界面效果

4. 行数显示

在 Microsoft Visual Studio 2017 中会显示文件源代码的行数标记，这和 Dreamweaver 等工具一样，能够便于用户对程序的维护，迅速找到对应的代码所在。在初始安装 Microsoft Visual Studio 2017 时，默认的是不显示代码行号。解决方法如下。

(1) 选择“工具”|“选项”命令，弹出“选项”对话框。

(2) 在左侧列表框中依次选择“文本编辑器”|C#|“常规”选项，然后勾选右侧的“行号”复选框，如图 1-23 所示。

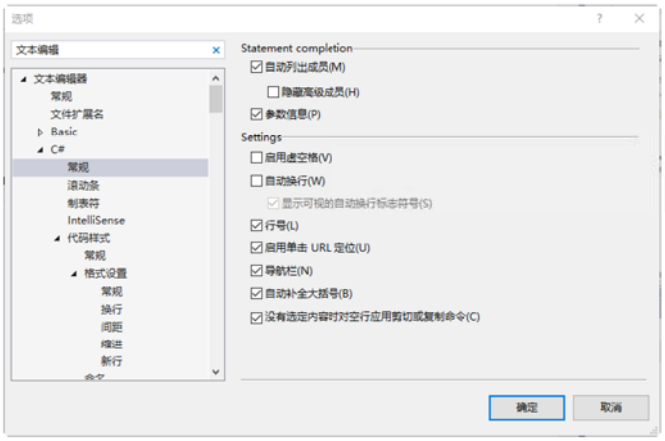


图 1-23　勾选“行号”复选框

(3) 单击“确定”按钮后返回代码界面，此时文件中每行源代码前都将显示一个行号，如图 1-24 所示。

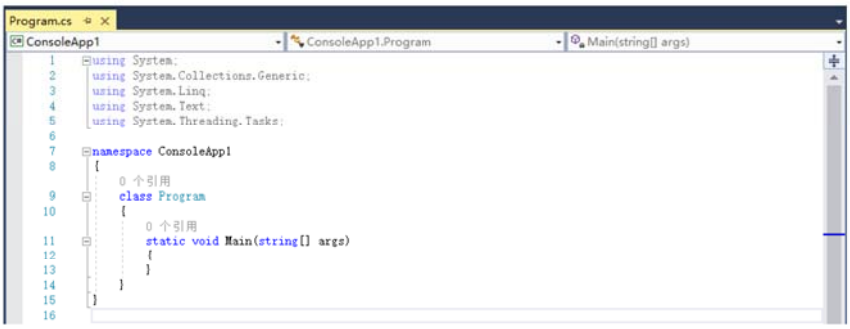


图 1-24　显示行号

1.3.5　生成与查错

选择“生成”|“生成解决方案”命令，可以生成当前解决方案的所有项目。当使用“生成”功能时，不会编译已经生成过并且生成后没有被修改的文件。如果使用“重新生成”功能，则将重新生成所有的文件。

解决方案和项目有如下两种生成模式。

- 调试模式：即 Debug 模式，生成的代码中含有调试信息，可以进行源代码级的调试。
- 发布模式：即 Release 模式，生成的代码中不含有调试信息，不能进行源代码级的调试，但是运行的速度很快。

选择“生成”|“配置管理器”命令，在弹出的“配置管理器”对话框中可以设置项目的生成模式，如图 1-25 所示。如果项目中出现错误，则不能成功生成，并在“错误列表”窗口中输出错误提示，如图 1-26 所示。

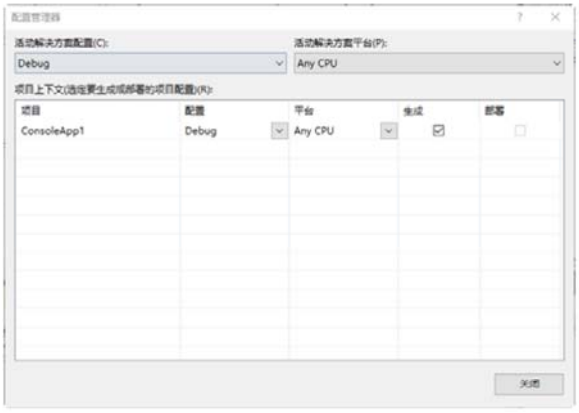


图 1-25　“配置管理器”对话框

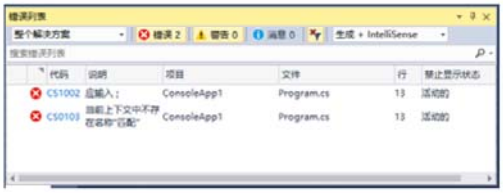


图 1-26　生成错误提示

Microsoft Visual Studio 2017 能够实现查错处理功能，在“输出”窗口将显示错误信息的具体详情，如图 1-27 所示。

如果将错误修改正确，则能正确生成，并在“输出”窗口内显示对应的生成处理结果，如图 1-28 所示。

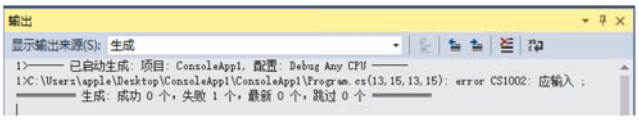


图 1-27　查错结果详情

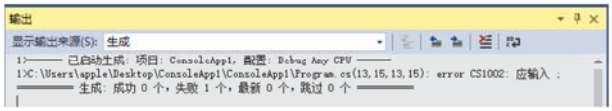


图 1-28　生成处理结果

1.4　实践案例与上机指导

通过本章的学习，读者基本可以掌握 C#语言的知识。其实 C#语言的基础知识还有很多，这需要读者通过课外渠道来加深学习。下面通过练习操作，达到巩固学习、拓展提高的目的。

↑扫码看视频

1.4.1　创建工程并编写代码

实例 1-1：创建并运行第一个 C#程序
源文件路径：daima\1\1-1

(1) 打开 Visual Studio 2017，选择“文件”|“新建”|“项目”命令，弹出“新建项目”对话框，然后选择创建一个“控制台应用(.NET Framework)”，如图 1-29 所示。

(2) 创建成功后，核心程序文件 Program.cs 将自动生成如下所示的代码：

```
using System.Collections.Generic;        //使用 using 引用命名空间
using System.Linq;                       //使用 using 引用命名空间
using System.Text;                       //使用 using 引用命名空间
using System.Threading.Tasks;            //使用 using 引用命名空间
namespace ConsoleApp1{                   //自定义命名空间 ConsoleApp1
    class Program                        //定义类 Program
    {
        static void Main(string[] args) //使用系统内置的主方法 Main()
        {
        }
    }
}
```

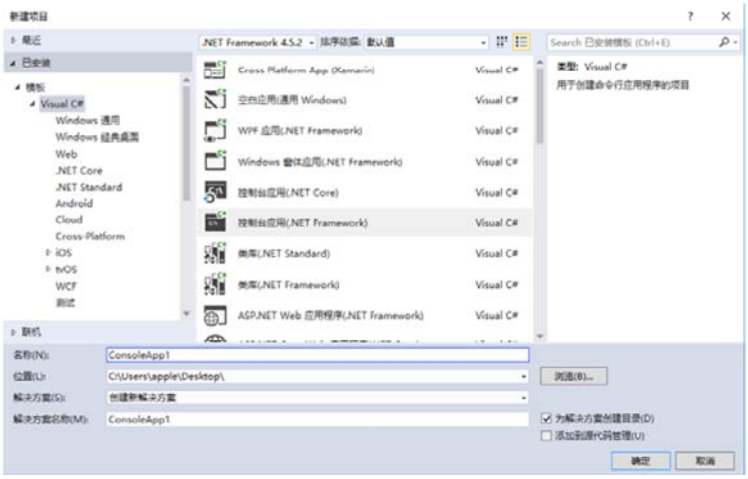


图 1-29　创建一个控制台应用

上述代码是 Visual Studio 2017 自动生成的，不会实现任何功能。接下来在方法 Main() 后面添加如下所示的 4 行代码，然后保存项目。

```
Console.Out.WriteLine("************************");      //在屏幕中输出星号
Console.WriteLine("哇塞，这是我的第一个程序。好棒哦！");      //在屏幕中输出文本
Console.WriteLine("************************");          //在屏幕中输出星号
Console.ReadLine();                                  //在屏幕中显示双引号中的所有内容
```

1.4.2　运行并调试

单击 Visual Studio 2017 工具栏中的“启动”按钮 ▶ 启动，即可运行本实例程序，执行效果如图 1-30 所示。

图 1-30　执行效果

1.4.3　分析代码

(1)　如下前面带有 using 的 4 行代码：using 是 C#语言中的指令，表示引用命名空间中的内容，这 4 个命名空间是微软官方预先编写好的，程序员直接使用即可。即使是最简单的控制台程序，也必须加入这 4 行代码。每行代码的具体含义读者此时无须理解，只要知道这是固定格式，具体含义将在本书后面的内容中进行讲解。

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

(2) 如下一行代码表示程序员自己定义的命名空间，任何一个C#程序都必须具有一个命名空间。自定义命名空间使用关键字namespace进行定义。

```
namespace ConsoleApp1
```

(3) 如下所示的代码都属于类Program中的成员，类使用关键字class进行定义。

```
class Program{
    static void Main(string[] args){
        Console.Out.WriteLine("************************");
        Console.WriteLine("哇塞，这是我的第一个程序。好棒哦！");
        Console.WriteLine("************************");
        Console.ReadLine();
    }
}
```

在上述代码中，Main(string[] args)行是一个方法，Main()方法是C#程序运行的起点，最重要的代码就写在Main方法后面的大括号中。在Main()方法里面有4行代码，它们是以分号结束的，具体说明如下所示。

- 第1行代码：作用是调用微软预先编写好的内置方法WriteLine()，在控制台中打印双引号之间的内容，也就是打印输出**********。
- 第2行代码：作用是调用微软预先编写好的内置方法WriteLine()，在控制台中打印双引号之间的内容，也就是打印输出“哇塞，这是我的第一个程序。好棒哦！”。
- 第3行代码：作用是调用微软预先编写好的内置方法WriteLine()，在控制台中打印双引号之间的内容，也就是打印输出“**********”。
- 第4行代码：作用是通过方法ReadLine()等待控制台DOS窗口的输入。当在控制台窗口按回车键时，控制台窗口会自动关闭。如果不加这一行代码，执行效果会如何呢？控制台窗口会显示一下马上关闭，出现一闪而过的效果。我们为了查看完整的执行效果，所以不能让控制台窗口一闪而过，为此就加入了上述第4行代码，让程序等待在控制台窗口的输入动作。

智慧锦囊

为什么C#语言会有命名空间、类等这种一层套一层的内容呢？这些都是C#应用程序应该有的框架，其中class是C#程序的最小单元，C#程序是由一个一个的类组成的。而命名空间的作用是用来组织和管理类的，一个命名空间可以包含多个类。在我们编写程序的时候，有很多类是微软编写好的，它们集成在Visual Studio 2017中，供我们直接使用，这样可以减少代码量。比如本实例程序的开头用using指令导入的就是微软官方提供的命名空间，目的是方便使用其中的类。

1.5 思考与练习

本章详细讲解了C#语言的基础性的知识，循序渐进地讲解了C#语言的历史、史上最强

开发工具——Microsoft Visual Studio 和初识 Microsoft Visual Studio 开发环境等知识。在讲解过程中，通过具体实例介绍了 C#语言的基础性知识。通过本章的学习，读者应该熟悉什么是 C#语言，并掌握其使用方法和技巧。

一、选择题

(1) Visual Studio 2017 是(　　)公司推出的开发工具。

A. 微软　　B. 谷歌　　C. 苹果　　D. 华为

(2) 下面不是面向对象编程语言的是(　　)。

A. C#　　B. C　　C. C++　　D. Java

二、判断对错

(1) .NET Framework 是微软为开发应用程序而创建的一个平台，利用它可以开发 Windows 桌面应用程序、Web 应用程序、Web 服务以及其他类型的应用程序。　　(　　)

(2) C#可以开发出 WPF 程序，利用 Visual Studio .NET 工具，C#能够实现和 Vista 界面一样的绚丽效果，吸引广大用户的眼球。　　(　　)

三、上机练习

(1) 编写第一个 C#程序，并尝试运行这个程序。

(2) 在网上下载一个 C#程序，并尝试运行这个程序。

新起点电脑教程

第 2 章

语法基础

本章要点

- 语句和注释
- 变量
- 常量
- 类型转换

本章主要内容

因为C#语言本身强大的语法特性决定了它能实现的功能，所以读者在学习C#的过程中，应该首先掌握C#语法的基本知识，这样才能真正了解并掌握C#这门语言。本章将引导读者从其最基本的语法知识开始进行循序渐进的学习，为读者步入本书后面知识的学习打下基础。

2.1 语句和注释

C#语言的最重要特质是其本身的语法结构，因为 C#是从 C 和 C++进化而来的，所以从外观和语法定义上看，C#和两者有着很多相似之处。在使用 C#语言编写软件程序的过程中，必须遵循它本身的独有特性，即基本的语法结构。

↑扫码看视频

2.1.1 语句

语句是指一段程序，是 C#程序的最基本构成单位。C#语句具有如下所示的 5 个基本特性。

1. 字符过滤性

和其他常用语言的编译器不同，若代码中含有空格、回车符或 Tab 字符，C#都不会考虑而是忽略不计。这样程序员在编写代码时，会有很大的自由度，不会因疏忽加入空白字符而造成程序的错误。

2. 语句结构

C#程序代码是由一系列的语句构成的，并且每个语句都必须以分号“;”结束。因为 C#中的空格和换行等字符被忽略，所以可以在同行代码中放置多个处理语句。

3. 代码块

因为 C#是一门面向对象的编程语言，所以其代码结构十分严谨和清晰。实现同一功能的 C#代码语句构成了独立的代码块，通过这些代码块可以使整个代码的结构变得更加清晰。所以说，C#代码块是整个 C#代码的核心。

4. 严格区分大小写

因为 C#语言中的大小写字符代表不同的含义，所以在编写代码时，必须注意每个字符的大小写格式，避免因大小写差异而出现名称错误等问题。

5. C#的代码块以大括号“{”开始，以大括号“}”结束

由此可见，语句就是一行或一段代码，这段代码能够实现某个功能。结构中包含多行语句，语句之间用分号分隔。在代码中使用缩进格式和非过滤处理，这使整个 C#代码变得更加清晰明了，提高了代码的可读性。

2.1.2　注释

注释是 C#程序中的必要构成元素之一，通过注释可以帮助程序员和使用人员快速了解当前语句的功能。特别是在大型应用程序中，因为整个项目内的代码块繁多，所以加入合理的注释必不可少。在 C#程序中，有如下两种添加注释的方法。

1. 两端放置

两端放置是指在程序的开头和结尾放置，具体格式是在开头插入“/*”，在结尾插入“*/”，在两者之间输入注释的内容。例如下面的代码：

```
/* 代码开始了 */
static void Main(string[] args)
```

2. 单“//”标记

单“//”标记和上面的两端放置不同，其最大特点是以“//”为注释的开始，在注释内容编写完后不必以“//” 结束，只需注释内容和 “//”在同行即可。例如下面的代码：

```
//代码开始了
static void Main(string[] args)
```

但是下面的代码是错误的：

```
//代码开始了
这也是注释，这行是错的注释
static void Main(string[] args)
```

智慧锦囊

(1) 独立语句独立代码行：虽然 C#允许在同行内放置多个语句，但为了提高代码的可读性，建议将每个语句放置在独立的代码行中，即每代码行都以分号“ ;”结束。

(2) 代码缩进处理：对程序内的每个代码块都要遵循缩进原则，使各代码块在整个程序中以更加清晰的效果展现出来。在使用 Visual Studio 2017 进行 C#开发时，Visual Studio 2017 能够自动实现代码缩进。

2.2　变　　量

↑扫码看视频

在 C#语言中，变量表示内存地址的名称。C#变量包括 3 个主要元素，分别是名称、类型和值。变量名称是变量在程序代码中的标识；变量类型决定其所代表的内存大小和类型；变量值代表内存块中的数据。

2.2.1 C#语言的数据类型

C#语言支持 Microsoft Visual Studio 2017 框架中定义的类，C#的变量类型是用类来定义的，即所有的类型都是类。C#语言中的类型如表 2-1 所示。

表 2-1 C#语言中的类型信息

类 型		描 述
值类型	简单类型	符号整型：sbyte，short，int，long
		无符号整型：byte，ushort，uint，ulong
		Unicode 字符：char
		浮点型：float，double
		精度小数：decimal
		布尔型：bool
	枚举类型	枚举定义：enum name{}
	结构类型	结构定义：Struct name{}
引用类型	类类型	基类：object
		字符串：string
		类定义：class name
	接口类型	接口定义：interface
	数组类型	数组定义：int[]
	委托类型	委托定义：delegate name

由表 2-1 可以看出，C#变量的常用类型分为两大类：引用类型和值类型。

2.2.2 引用类型

在 C#语言中，所有被称为类的变量类型都是引用类型，主要包括类、接口、数组和委托。具体说明如下所示。

- 类类型：能够定义包含数据成员、函数成员和嵌套类型的数据结构，其中的数据成员包括常量和字段，函数成员包括方法、属性和事件等。
- 接口类型：能够定义一个协定，实现某接口的类或结构必须遵循该接口定义的协定。
- 数组类型：是一种数据结构，包含可通过计算索引访问的任意一个变量。
- 委托类型：是一种数据结构，能够引用一个或多个方法。

实例 2-1：使用引用数据类型显示信息
源文件路径：daima\2\2-1

实例文件 Program.cs 的主要实现代码如下所示。

```
class C{                              //创建一个类 C
```

```
    public int Value;          //声明一个公共 int 类型的变量 Value
}
static void Main(string[] args) {    //调用系统内置的主方法 Main()
    int v1 = 89;               //声明一个 int 类型的变量 v1，并初始化为 89
    int v2 = v1;               //声明一个 int 类型的变量 v2,并将 v1 赋值给 v2
    v2 = 157;                  //重新将变量 v2 赋值为 157
    C r1 = new C();            //使用 new 关键字创建引用对象
    C r2 = r1;                 //使 r1 等于 r2
    r2.Value = 112;            //设置变量 r2 的 Value 值为 112
    Console.WriteLine("金州勇士队本赛季的单场最低得分和最高得分别
                      是:{0},{1}", v1, v2);  //输出变量 v1 和 v2 的值
    Console.WriteLine("金州勇士队本赛季的平均得分是:{0},{1}", r1.Value,
                      r2.Value);     //输出引用类型对象的 Value 值
    Console.ReadLine();
}
```

执行效果如图 2-1 所示。

```
金州勇士队本赛季的单场最低得分和最高得分别是:89,157
金州勇士队本赛季的平均得分是:112,112
```

图 2-1　执行效果

2.2.3　值类型

如果在 C#程序中只有引用类型，则会很容易影响整个程序的性能，而通过值类型能够很好地解决这个问题。值类型能够存储程序中用到的数值，例如通过一个名为 mm 的变量存储数值 100，这样在程序中只需调用变量名 mm，即可调用数值 100。

在 C#程序中，值类型是从类 System.ValueType 中继承的，其中包括结构、枚举和大多数的基本类型。具体说明如下所示。

- 结构类型：能够声明常量、字段、方法和属性等。
- 枚举类型：是具有命名常量的独特类型，每个枚举类型都有一个基础的类型，是通过枚举来声明的。

实例 2-2：使用值类型显示信息

源文件路径：daima\2\2-2

实例文件 Program.cs 的具体实现代码如下所示。

```
static void Main(string[] args) {  //调用系统内置的主方法 Main()
        bool tr;                   //定义 bool 类型变量 tr
        int s;                     //定义 int 类型变量 s
        float fl = 3.14F;          //定义 float 类型变量 fl,并设置初始值是 3.14F
        double d = 3.14D;          //定义 double 类型变量 d,并设置初始值是 3.14D
        fl = (float)d;             //将变量 d 转换为 float 类型，并赋值给变量 fl
        int kuli = 30;             //声明一个 int 类型的变量 kuli
        byte hadeng = 29;          //声明一个 byte 类型的变量 hadeng
        Console.WriteLine("本赛季常规赛得分第一库里平均得分：{0}", kuli);
                                   //输出 int 类型变量 kuli
        Console.WriteLine("本赛季常规赛得分第二哈登平均得分：{0}", hadeng);
```

```
                                        //输出 byte 类型变量 hadeng
            Console.ReadLine();
}
```

执行效果如图 2-2 所示。

```
本赛季常规赛得分第一库里平均得分：30
本赛季常规赛得分第二哈登平均得分：29
```

图 2-2　执行效果

2.2.4　基本类型

基本类型是编译器直接支持的类型。在 C#程序中，使用关键字来命名基本类型，它是构造其他类型的基础。其中值类型的基本类型通常被称为简单类型，例如下面的代码声明了一个 int 类型变量，变量名是 mm。

```
int mm=123;
```

1. 整型

在 C#语言中定义了 8 种整型，具体说明如表 2-2 所示。

表 2-2　C#整型信息

类　型	值的范围
sbyte	-128～127 的整数
byte	0～255 的整数
short	-32768～32767 的整数
ushort	0～65535 的整数
int	-2147483648～2147483647 的整数
uint	0～4294967295 的整数
long	-9223372036854775808～-2147483647 的整数
ulong	64 位无符号整数，占 8 个字节，取值范围为 0～18446744073709551615 的整数

智慧锦囊

某变量名前的字符 u 表示不能在此变量中存储负值。

2. 浮点型

在 C#语言中，浮点型包括 float 和 double 两种，具体说明如表 2-3 所示。

表 2-3　C#浮点型信息

类　型	允许值的范围
float	32 位单精度实数，占 4 个字节，取值范围为 3.4×10^{-38}～3.4×10^{38}
double	64 位实数，占 8 个字节，取值范围为 1.7×10^{-308}～1.7×10^{308}

3. 布尔型

在 C#语言中，布尔型有两个取值，分别是 true 和 false，即代表“是”和“否”的含义。

4. 字符型

在 C#语言中，字符型的取值和 Unicode 的字符集相对应，通过字符型可以表示世界上所有语言的字符。字符型文本一般用一对单引号来标识，例如‘MM’和‘NN’。

使用字符型中的转义字符可以表示一些特殊字符，常用的 C#转义字符如表 2-4 所示。

表 2-4　C#转义字符列表

转义字符	描　述
\’	转义单引号
\’’	转义双引号
\\	转义反斜杠
\0	转义空字符
\a	转义感叹号
\b	转义退格
\f	转义换页
\n	转义新的行
\r	转义回车
\t	转义水平制表符
\v	转义垂直制表符
\x	后面接 2 个两进制数字，表示一个 ASCII 字符
\u	后面接 4 个两进制数字，表示一个 ASCII 字符

5. decimal 型

在 C#语言中，decimal 是一种高精度的、128 位的数据类型，常用于金融和货币计算项目。Decimal 类型表示 28 或 29 位有效数字，取值范围是$\pm 1.0 \times 10^{-28}$ ~$\pm 7.9 \times 10^{28}$。

6. string 型

在 C#语言中，string 型用来表示字符串，常用于代替文本字符，是字符型对象(char)的连续集合。当创建 string 型的字符串值后，就不能再修改，除非重新赋值。

7. object 型

在 C#语言中，object 是最基础的类型，可以表示任何类型的值。

实例 2-3：使用变量输出文本

源文件路径：daima\2\2-3

实例文件 Program.cs 的主要实现代码如下所示。

```
string name;          //string 变量类型，名为 name
string myString;      //string 变量类型，名为 myString
name = "帅哥";        //变量 name，设置其值为“帅哥”
```

```
myString = "\"我\"是";
//设置变量 myString 的值为 ""我"是"，字符 "\"" 的功能是转义双引号
Console.WriteLine("{0} {1}", myString, name);//输出两个变量的值
```

实例的最终结果是输出两个变量的拼接值，具体如图 2-3 所示。

图 2-3 执行效果

智慧锦囊

在上述实例中，使用了 Visual Studio 2017 工具进行编译并调试运行。除了上述编译方式外，也可以使用 SDK 命令进行编译，并且可以同时使用编译命令对多个.cs 文件进行编译。例如某个项目中有 123.cs 和 456.cs 两个文件，则可以通过“ csc /out:name.exe 123.cs 456.cs”将其编译为一个可执行文件“ name.cs”。上述实例只是使用了“ /out”编译参数，其他编译参数的使用方式和其相似。

2.2.5 变量命名

在 C#语言中，不能随意给变量命名，必须遵循如下两个命名原则。

(1) 变量名的第一个字符必须是字母、下画线“-”或@。

(2) 第一个字符后的字符可以是字母、下画线或数字。

另外，读者还需要特别注意 C#编译器中的关键字，例如关键字 using。如果错误地使用了编译器中的关键字，则程序将会出现编译错误。例如下面的变量名都是正确的：

```
aaaaaa
@aaaaaaa
_aaaaaa
```

而下面的变量名是不正确的：

```
6666aaa
aa-bb
namespace
```

因为 C#是区分大小写的，所以在命名过程中要注意大小写。为了使编写的程序结构更加清晰明了，建议读者根据变量的作用来命名，并区分大小写字母。例如，存储名字的变量命名为 Name，存储用户名的变量命名为 UserName。

2.2.6 变量的声明和赋值

在 C#程序中，声明变量的语法格式如下所示。

```
类型 变量名;
```

例如，下面的代码声明了一个名称为 Name 的 string 类型变量。

```
string Name;
```

在 C#程序中可以同时声明多个变量，各个变量之间需要用逗号“,”隔开。例如，下面的代码同时声明了名为 aa、bb 和 cc 的三个变量。

```
string aa,bb,cc;
```

在声明多个变量的同时，可以对各个变量进行赋值，例如下面的代码：

```
string aa= "我",bb= "是",cc= "谁";
```

因为 C#是一门类型安全的语言，所以当给一个变量赋值时，值的类型必须满足如下条件之一。

- 与变量的类型相同。
- 是一个 C#将执行赋值转换的类型。
- 是一个可以显式转换为正确类型的类型。

由前面的知识了解到，C#变量在命名时必须遵循特定的规则。并且 C#变量在使用前必须进行初始化处理，例如下面的首行代码进行了初始化，而第二行代码进行了赋值。

```
int age;     //初始化操作，定义一个 int 类型变量 age
age=26;      //赋值变量 age 的值是 26
```

2.3 常　　量

在计算机编程语言领域中，通常将值固定不变的量称为常量。在 C#语言中，常量类型只能是下列类型中的一种：sbyte、byte、ushort、short、int、uint、ulong、long、char、float、double、decimal、bool、string 或枚举。

↑扫码看视频

通常将 C#语言中的常量分为两种，分别是文本常量和符号常量。具体说明如下所示。

(1) 文本常量：是输入到程序中的值，例如“12”和“Mr 王”等。

(2) 符号常量：其声明方法和变量的声明方法类似，唯一的区别是常量在声明前必须使用修饰关键字 const 开头，并且常量在定义时被初始化。常量一旦被定义后，在常量的作用域内其本身的名字和初始化值是等价的。

符号常量的命名规则和变量的命名规则相同，但是符号常量名的第一个字母最好是大写字母，并且在同一个作用域内所有的变量名和常量不能重名。例如下面的常量命名是正确的：

```
const double mm=3.14;
const string nn=LiuDeHua;
```

而下面的常量命名是错误的：

```
const double mm;              //没有初始值
string nn=LiuDeHua;           //没有关键字 const
```

实例 2-4：赋值并显示不同的常量

源文件路径：daima\2\2-4

实例文件 Program.cs 的具体实现代码如下所示。

```
int Score = 157;                  //声明一个 int 整型变量，设置初始值是 157
const int WScore = 87;            //声明一个 int 整型常量，初始值是 87
Console.WriteLine("当勇士队发挥好的时候，得分 Score={0}", Score);
 //输出变量 Score 的值
Console.WriteLine("当勇士队发挥不好的时候，得分 WScore={0}", WScore);
 //输出常量 WScore 的值
Score = 1039;                     //重新将变量赋值为 1039
Console.WriteLine("当勇士队发挥逆天的时候，得分 Score={0}", Score);
 //重新输出变量 Score 的值
```

执行效果如图 2-4 所示。

```
当勇士队发挥好的时候，得分Score=157
当勇士队发挥不好的时候，得分WScore=87
当勇士队发挥逆天的时候，得分Score=1039
```

图 2-4　执行效果

2.4　类 型 转 换

在 C#语言中，最简单的数据类型是 char 类型，功能是用一个数字来表示 Unicode 字符集中的一个字符。在默认情况下，不同类型的变量使用不同的模式来表示数据。但是有时需要联合使用不同的类型实现某个功能，此时就需要用到类型转换。C#语言中的类型转换有隐式转换和显式转换两种，在本节将分别介绍这两种转换方式的基本知识。

↑扫码看视频

2.4.1　隐式转换

隐式转换是指不需要特别声明即可默认在所有情况下进行转换，这是系统的默认转换方式。在进行隐式转换时，编译器不需要进行检查就能实现安全的转换处理。C#的隐式转换一般不会失败，也不会导致信息丢失。例如在下面的代码中，将 int 类型变量 mm 隐式地转换为了 long 类型。

```
int mm=20;
long nn=mm;
```

在 C#语言中，绝大多数的简单数据类型可以实现隐式转换，但是其中的 bool 和 string 是不能进行隐式转换的。在 C#语言规范中，编译器可以隐式执行类型转换的类型如表 2-5 所示。

表 2-5　可以隐式转换的数值类型列表

类　型	可转换为
byte	short、ushort、uint、int、ulong、long、float、double、decimal
sbyte	short、int、long、float、double、decimal
short	int、long、double、decimal
ushort	uint、int、ulong、long、float、double、decimal
int	long、float、double、decimal
uint	ulong、long、float、double、decimal
long	float、double、decimal
ulong	float、double、decimal
float	double
char	ushort、uint、int、ulong、long、float、double、decimal

知识精讲

因为不存在 char 类型的隐式转换，所以其他整型值不会自动转换为 char 类型。另外读者不需要强记表 2-5 的内容，只需牢记各类型的取值范围即可。因为对于任何类型 A，只要其取值范围被完全包含在类型 B 的取值范围内，就可以隐式地将类型 A 转换为类型 B。

实例 2-5：使用隐式转换

源文件路径：daima\2\2-5

实例文件 Program.cs 的主要实现代码如下所示。

```
byte aa = 100;       //定义变量 aa，设置类型为 byte，并设置初始值为 100
int bb = aa;         //定义变量 bb，隐式设置 aa 类型为 int，并设置 bb 的值为 100
long cc = bb;        //定义变量 cc，隐式设置 bb 的类型为 long，并设置 cc 的值为 100
double dd = cc;      //定义变量 dd，隐式设置 cc 的类型为 double，并设置 dd 的值为 100
Console.WriteLine("mm = {0}", aa); //输出变量 aa 的值
Console.WriteLine("nn = {0}", bb); //输出变量 bb 的值
Console.WriteLine("zz = {0}", cc); //输出变量 cc 的值
Console.WriteLine("ff = {0}", dd); //输出变量 dd 的值
```

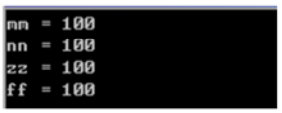


图 2-5　实例执行效果

实例的最终输出结果是 4 个变量的值相同，实现了隐式转换。执行效果如图 2-5 所示。

2.4.2　显式转换

显式转换是一种强制性的转换方式，在使用显式转换时，必须在代码中明确声明要转

换的类型。在C#语言中，使用显式转换的语法格式如下所示。

```
类型 1 变量名=(类型 2)变量名
```

其中“类型1”表示待转换的类型，“类型2”表示准备转换成的类型。下面的两段代码可区分显式转换和隐式转换的区别，其中实现隐式转换的代码如下所示。

```
int mm=20;
long nn=mm;                          //隐式转换
```

实现显式转换的代码如下所示。

```
int mm=20;
long nn=(long)mm;                    //显式转换，将 int 类型强制转换成 long 类型
```

从一个数值的类型向另一个数值类型进行转换的过程被称为显式数值转换。在使用显示转换时，虽然在转换过程中会出现数据丢失，却也总是能强制将表达式从任何数值类型转换为任何其他类型。

知识精讲

虽然C#允许变量进行显式转换，但需要注意如下所示的两点。

(1) 隐式转换是显式转换的一种特例，所以允许把隐式转换书写成显式转换格式。

(2) 显式转换不安全，因为不同类型的变量取值范围不同，所以如果强制执行显式转换，可能会造成数据丢失。

在C#语言中，编译器可以显式执行的数值转换类型如表2-6所示。

表2-6 C#可以显式转换的数值类型列表

类 型	可转换为
byte	sbyte、char
sbyte	byte、ushort、uint、ulong、char
short	sbyte、byte、ushort、uint、ulong、long、char
ushort	sbyte、byte、short、char
int	sbyte、byte、short、ushort、uint、ulong、char
uint	sbyte、byte、short、ushort、int、char
long	sbyte、byte、short、ushort、uint、int、ulong、char
ulong	sbyte、byte、short、ushort、uint、int、long、char
float	sbyte、byte、short、ushort、uint、int、ulong、long、char、float
char	sbyte、byte、short
decimal	sbyte、byte、short、ushort、uint、int、ulong、long、char、float、double
double	sbyte、byte、short、ushort、uint、int、ulong、long、char、float、decimal

实例2-6：使用C#显式转换

源文件路径：daima\2\2-6

实例文件的主要实现代码如下所示。

```
double mm = 10000;                    //定义变量 mm 的类型为 double，初始值为 10000
long nn = (long)mm;                   //将 mm 转换为 long 数值类型，并赋值给变量 nn
int zz = (int)nn;                     //将 nn 转换为 int 数值类型，并赋值给变量 zz
byte ff = (byte)zz;                   //将 zz 转换为 byte 数值类型，并赋值给变量 ff
Console.WriteLine("mm = {0}", mm); //输出变量 mm 的值
Console.WriteLine("nn = {0}", nn); //输出变量 nn 的值
Console.WriteLine("zz = {0}", zz); //输出变量 zz 的值
Console.WriteLine("ff = {0}", ff); //输出变量 ff 的值
```

实例执行后的效果如图 2-6 所示。

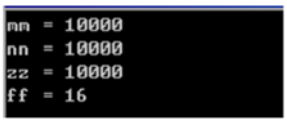


图 2-6　实例执行效果

2.5　实践案例与上机指导

通过本章的学习，读者基本可以掌握 C#语言基础语法的知识。其实 C#基础语法的知识还有很多，这需要读者通过课外渠道来加深学习。下面通过练习操作，达到巩固学习、拓展提高的目的。

↑扫码看视频

2.5.1　枚举

本书前面讲解过的各种变量类型，除 string 类型外基本上都有明确的取值范围。但是在现实应用中，可能只需要变量取值范围内的一个或几个值，此时就可以使用枚举来实现。在 C#语言中，使用关键字 enum 来定义枚举，具体语法格式如下所示。

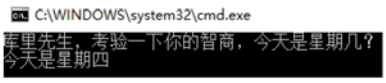
```
enum 枚举名称:类型{
      枚举值 1,
      枚举值 2,
      …
      枚举值 n
}
```

实例 2-7：使用枚举保存数据

源文件路径：daima\2\2-7

实例文件的主要实现代码如下所示。

```
enum MyDate{                          //使用 enum 创建枚举
        Sun = 0,                      //设置枚举值名称 Sun，枚举值为 0
        Mon = 1,                      //设置枚举值名称 Mon，枚举值为 1
        Tue = 2,                      //设置枚举值名称 Tue，枚举值为 2
        Wed = 3,                      //设置枚举值名称 Wed，枚举值为 3
        Thi = 4,                      //设置枚举值名称 Thi，枚举值为 4
        Fri = 5,                      //设置枚举值名称 Fri，枚举值为 5
        Sat = 6                       //设置枚举值名称 Sat，枚举值为 6
    }
    static void Main(string[] args){
        int k = (int)DateTime.Now.DayOfWeek; //获取代表星期几的返回值
        Console.WriteLine("库里先生，考验一下你的智商，今天是星期几？");
        switch (k){
            //如果 k 等于枚举变量 MyDate 中的 Sun 的枚举值，则输出“今天是星期日”
            case (int)MyDate.Sun: Console.WriteLine("今天是星期日"); break;
            //如果 k 等于枚举变量 MyDate 中的 Mon 的枚举值，则输出“今天是星期一”
            case (int)MyDate.Mon: Console.WriteLine("今天是星期一"); break;
            //如果 k 等于枚举变量 MyDate 中的 Tue 的枚举值，则输出“今天是星期二”
            case (int)MyDate.Tue: Console.WriteLine("今天是星期二"); break;
            //如果 k 等于枚举变量 MyDate 中的 Wed 的枚举值，则输出“今天是星期三”
            case (int)MyDate.Wed: Console.WriteLine("今天是星期三"); break;
            //如果 k 等于枚举变量 MyDate 中的 Thi 的枚举值，则输出“今天是星期四”
            case (int)MyDate.Thi: Console.WriteLine("今天是星期四"); break;
            //如果 k 等于枚举变量 MyDate 中的 Fri 的枚举值，则输出“今天是星期五”
            case (int)MyDate.Fri: Console.WriteLine("今天是星期五"); break;
            //如果 k 等于枚举变量 MyDate 中的 Sat 的枚举值，则输出“今天是星期六”
            case (int)MyDate.Sat: Console.WriteLine("今天是星期六"); break;
        }
        Console.ReadLine();
    }
```



执行后的效果如图 2-7 所示。

图 2-7　实例执行效果

2.5.2　结构

结构是一种值类型数据结构，能够使一个单一变量存储各种数据类型的相关数据。在C#语言中，使用关键字 struct 来定义结构，具体语法格式如下所示。

```
struct 名称 {
        结构变量 1;
        结构变量 2;
        …
        结构变量 n;
            }
```

实例 2-8：计算矩形的面积

源文件路径：daima\2\2-8

实例文件 Program.cs 的主要实现代码如下所示。

```
        public struct Rect {                          //定义一个矩形结构
            public double width;                      //矩形的宽
```

```
    public double height;                                    //矩形的高
    //构造方法 Rect，初始化矩形的宽和高
    public Rect(double x, double y) {
        width = x;     //矩形的宽
        height = y;    //矩形的高
    }
    //计算矩形面积
    public double Area() {
        return width * height;     //返回宽和高的积
    }
}
static void Main(string[] args){
    Rect rect1;                                              //实例化矩形结构
    rect1.width = 5;                                         //为矩形宽赋值
    rect1.height = 3;                                        //为矩形高赋值
    Console.WriteLine("主持人：库里先生，都知道你篮球打得好，平时学习怎么样？");
    Console.WriteLine("库里：还行吧！");
    Console.WriteLine("主持人：出个问题考考你吧！");
    Console.WriteLine("库里：宽 5 高 3 的矩形面积为：" + rect1.Area());
    Rect rect2 = new Rect(6, 4);       //使用构造函数实例化矩形结构
    Console.WriteLine("库里：宽 6 高 4 的矩形面积为：" + rect2.Area());
}
```

执行后的效果如图 2-8 所示。

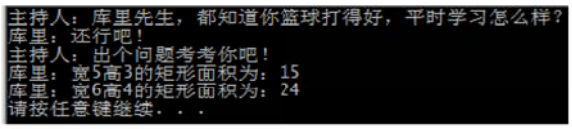


图 2-8　实例执行效果

2.6　思考与练习

本章详细讲解了 C#基础语法的知识，循序渐进地讲解了语句、注释、变量、常量、类型转换和枚举、结构等知识。在讲解过程中，通过具体实例介绍了使用 C#基础语法的方法。通过本章的学习，读者应该熟悉使用 C#基础语法的知识，掌握其使用方法和技巧。

一、选择题

(1) 在 C#语言中，内置类(　　)的功能是以静态的方法提供数学函数的计算方法。

A. Maths　　B. Math　　C. math

(2) 类型 byte 不可以显式转换为(　　)。

A. sbyte　　B. char　　C. ushort

二、判断对错

(1) 在 C#程序中，方法 Write()能够输出控制台内的指定数据，但是不能在字符的后面自动输出一个换行符。　　(　　)

(2) 方法 Read()能够从控制台的输入流中读取下一个字符，如果没有字符则返回-1。当读操作结束后，这个方法才会被返回。 ( )

三、上机练习

(1) 转换字母的大写或小写。

(2) 实现字母与 ASCII 码的转换。