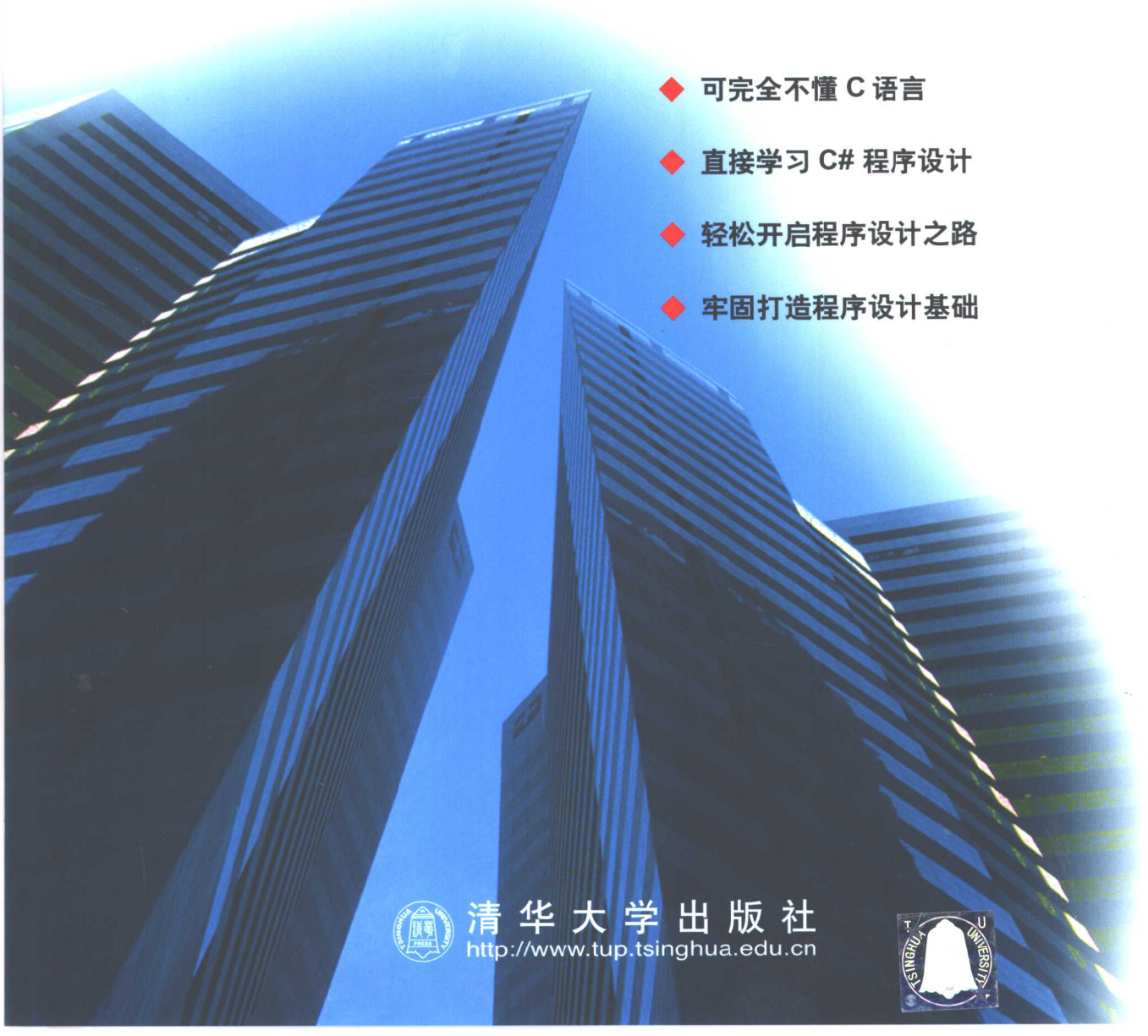
推荐教材与自学用书

Visual C# .NET
程序设计基础

孙永强　戴 锋　陈宗斌　编著



- 可完全不懂 C 语言
- 直接学习 C# 程序设计
- 轻松开启程序设计之路
- 牢固打造程序设计基础

清华大学出版社
http://www.tup.tsinghua.edu.cn

Visual C# .NET
程序设计基础

孙永强　戴　锋　陈宗斌　　编著

清华大学出版社

(京)新登字 158 号

内 容 简 介

Visual C# .NET 是 Microsoft 公司推出的新型的纯面向对象编程语言，它从著名的 Visual C++发展而来。Visual C#抛弃了 C++中各种不安全因素，使程序员能够更有效率地使用 C#开发应用软件。

本书从最基本的概念入手，在前面大部分章节中讲解了 C#语言及使用 C#语言编写应用程序的方法，书中不仅有详尽的 C#语法描述和介绍，更为 C++和 Java 程序员快速掌握 C#提供了语法对照提示。在后面几章中，讲解了 C#的开发应用，通过实例介绍了用.NET Framework 开发 Windows 应用程序和 Web 应用程序的知识。

本书可作为编程初学者自学 C#程序设计语言的入门书籍，也适合希望快速掌握 C#程序设计的 Visual C++和 Java 程序员阅读。

书　　名：Visual C＃ .NET 程序设计基础
作　　者：孙永强　戴　锋　陈宗斌
责任编辑：许瑛琪
出 版 者：清华大学出版社(北京清华大学学研大厦，邮编 100084)
http://www.tup.tsinghua.edu.cn
印 刷 者：北京市人民文学印刷厂
发 行 者：新华书店总店北京发行所
开　　本：787×1092 1/16　**印张**：24.75　**字数**：586 千字
版　　次：2002 年 5 月第 1 版　2002 年 5 月第 1 次印刷
书　　号：ISBN 7-302-05384-7/TP・3165
印　　数：0001～5000
定　　价：30.00 元

前　言

Visual C# .NET（简称 C#，读作 C Sharp）语言是 Microsoft 公司为推行.NET 战略而发布的一种全新的编程语言。它是一种全新的、纯面向对象的编程语言，具有清晰明了的语法结构、优秀的编程开发环境和高效率的编译工具。C#语言从 C++语言发展而来，继承了 C++语言的所有优点，同时根据.NET 战略的需要增强了自身的功能，抛弃了 C++中各种不安全因素，从而使程序员能够更有效率地使用 C#开发应用软件。

本书的一个显著特点是：将 C#这种基础编程语言放在与之相应的开发工具中介绍，让用户在学会 C#语言的同时，熟悉 Visual Studio .NET 集成开发环境，为用户进一步深入学习 Visual C# .NET 打下牢固的基础。这是本书与市场上其他介绍 C#的书籍的一个显著的不同之处。本书的第二个显著特点是：明确定位初级读者，让对编程怀有满腔热情、而又不知从何学起的读者深入掌握基础编程语言。本书从程序设计最基础的知识开始介绍，用户可完全不懂程序设计，直接在 Visual Studio .NET 开发环境中学习 C#语言。此外，本书从一开始就向用户介绍面向对象的程序设计知识，把结构化程序设计的知识融合到面向对象的知识中去介绍，改变了以往在介绍面向对象知识之前先介绍面向过程知识的老套路，因为实践证明，对面向过程知识理解得越深，接受面向对象编程方法就越困难。所以本书从一开始就介绍面向对象的概念和知识，让用户能更好地接受这一最先进的程序设计方法。这可以说是本书第三个显著的特点。

本书经过精心组织，合理安排，将全书内容划分为 18 章，各章节内容大致安排如下：

第 1 章先带领用户熟悉 Visual Studio .NET 集成开发环境，然后借助一个简短的程序，分析了 C#语言编写的程序结构。

第 2 章～第 10 章由浅入深介绍了 C#语言的类型系统，考虑到面向对象程序设计的特点，重点介绍了类这种类型，让用户深切领略面向对象程序设计的特点。

第 11 章～第 16 章介绍了 C#语言的一些高级编程知识，其中很多是 C#语言添增的功能。通过对这些章节的学习，能扩展用户的编程思路，进一步学习最先进的编程思想和手段。

第 17、18 章带领用户用 C#语言开发简单的 Windows 应用程序和 ASP.NET Web 应用程序，通过这两章的学习，能为用户下一步深入学习 Visual C# .NET 打下良好的基础。

全书结构合理、思路清晰，内容安排重点突出。本书在介绍各个知识点时，考虑到初学者的需要，绝大多数知识点都给出了完整的程序示例，这些程序示例与所介绍的知识点结合非常紧密，并且例子本身力求短小、精悍。本书非常适合广大程序设计初学者阅读，对于有一定经验的编程人员，也可以作为深入领略 C#编程语言的工具。

由于时间仓促，加上作者水平有限，书中难免会出现错误疏漏，欢迎广大读者批评指正。

目　录

第 1 章　初识 C#.....1
1.1　C#语言简介.....1
1.1.1　程序设计与程序设计语言.....1
1.1.2　C#语言产生的背景.....2
1.1.3　C#语言的特点.....3
1.2　熟悉 Visual Studio .NET 集成开发环境.....3
1.2.1　集成开发环境.....3
1.2.2　打开 Visual Studio .NET 集成开发环境.....3
1.2.3　Visual Studio .NET 集成开发环境界面介绍.....4
1.3　C#版的 Helloworld 程序.....4
1.3.1　在集成开发环境下建立 Helloworld 程序.....5
1.3.2　在命令行环境下编译运行 Helloworld 程序.....6
1.3.3　C#程序结构分析.....7
第 2 章　C#程序设计基础知识.....9
2.1　面向对象方法简介.....9
2.2　Unicode.....12
2.3　标识符.....13
2.4　简单类型和常数.....15
2.4.1　数值类型.....15
2.4.2　数值常数.....17
2.4.3　字符类型和字符常数.....18
2.4.4　布尔类型和布尔型常数.....20
2.5　字符串.....20
2.5.1　字符串类型.....20
2.5.2　字符串常数.....20
2.6　类型的实例.....21
2.6.1　变量的定义.....21
2.6.2　变量的赋值.....23
2.6.3　变量的初始化.....25
2.6.4　常量.....26
2.6.5　C#的类型系统简介.....27
2.6.6　装箱和拆箱.....29
2.7　运算符与表达式.....31
2.7.1　算术运算.....31
2.7.2　关系运算.....33
2.7.3　条件逻辑运算和逻辑非运算.....34
2.7.4　位运算.....35
2.7.5　赋值表达式.....37
2.7.6　字符串连接.....38
2.7.7　C#的其他运算符和表达式.....38
2.7.8　运算符优先级.....39
2.8　语句.....40
2.9　练习题.....41
第 3 章　类与 C#程序.....42
3.1　类及其构成.....42
3.2　类的数据成员.....44
3.2.1　常量成员.....44
3.2.2　字段成员.....46
3.3　类的方法成员.....49
3.4　类的属性成员.....50
3.5　C#程序的构成.....53
3.6　练习题.....54
第 4 章　程序流程.....56
4.1　顺序结构.....56
4.2　分支结构.....58
4.2.1　if 语句.....58
4.2.2　switch 语句.....62
4.3　循环结构.....65
4.3.1　while 语句.....65
4.3.2　do... while 语句.....67
4.3.3　for 语句.....68

4.3.4 foreach 语句 71
4.3.5 无条件跳转语句 73
4.4 练习题 76
第 5 章 类的方法成员 79
5.1 方法概述 79
5.2 定义及调用类的方法 79
5.2.1 方法定义 80
5.2.2 方法的返回值 81
5.2.3 方法调用 82
5.3 方法的参数 83
5.3.1 无参方法 83
5.3.2 有参方法 84
5.4 方法重载 88
5.4.1 重载的必要性 88
5.4.2 调用重载的方法 89
5.4.3 方法重载的注意事项 90
5.5 递归 91
5.5.1 递归方法的概念 92
5.5.2 递归方法的求解过程 92
5.5.3 用非递归方法代替递归方法 93
5.6 练习题 94
第 6 章 构造完整的类 97
6.1 构造函数与析构函数 97
6.1.1 创建并初始化类的实例 97
6.1.2 构造函数 99
6.1.3 析构函数 106
6.2 字段成员 107
6.2.1 只读字段 107
6.2.2 实例字段的初始化 109
6.3 属性 110
6.3.1 属性访问函数 110
6.3.2 使用属性 112
6.4 索引指示器 113
6.4.1 定义索引指示器 114
6.4.2 使用索引指示器访问对象 115
6.5 类的静态成员 116
6.5.1 静态成员与实例成员的区别 116
6.5.2 静态字段成员 117
6.5.3 静态方法 121
6.5.4 静态构造函数 122
6.5.5 静态属性 124
6.6 运算符重载 126
6.6.1 运算符重载的必要性 126
6.6.2 重载运算符的方法 126
6.6.3 重载单目运算符 128
6.6.4 重载双目运算符 130
6.7 练习题 132
第 7 章 继承与多态 135
7.1 C#的继承机制 135
7.1.1 继承的基本知识 135
7.1.2 继承的工作方式 136
7.1.3 派生类的构造与析构 140
7.1.4 base 关键字 141
7.1.5 隐藏基类成员 143
7.2 多态性 145
7.2.1 多态性概述 145
7.2.2 虚方法 148
7.2.3 多态的实现 150
7.3 抽象类和抽象方法 152
7.3.1 抽象类 152
7.3.2 抽象方法 154
7.4 密封类和密封方法 155
7.4.1 密封类 155
7.4.2 密封方法 156
7.5 练习题 157
第 8 章 接口 160
8.1 概述 160
8.2 接口的声明 162
8.3 接口成员的声明 164
8.4 接口成员的访问 166
8.5 接口的实现 168
8.5.1 显式接口成员实现 169
8.5.2 接口映射 171
8.5.3 接口实现的继承 179
8.5.4 接口的重新实现 181

8.6 练习题 184
第 9 章 结构类型和枚举类型 188
9.1 结构类型 188
9.1.1 定义结构类型 188
9.1.2 访问结构成员 189
9.1.3 结构与类的区别 190
9.2 枚举类型 192
9.2.1 定义枚举类型 192
9.2.2 枚举成员的赋值 193
9.2.3 访问枚举成员 196
9.2.4 System.Enum 类 199
9.3 练习题 202
第 10 章 数组 204
10.1 一维数组 204
10.1.1 一维数组的定义 204
10.1.2 一维数组的初始化 205
10.1.3 访问一维数组的元素 207
10.2 多维数组 209
10.2.1 二维数组的定义 209
10.2.2 二维数组的初始化 209
10.2.3 访问二维数组的元素 211
10.2.4 作为参数传递数组 214
10.3 锯齿状数组 215
10.3.1 二维锯齿状数组的定义 215
10.3.2 初始化二维锯齿状数组 216
10.3.3 访问二维锯齿状数组元素的方法 217
10.4 数组应用：排序 219
10.4.1 冒泡排序法 219
10.4.2 快速排序法 222
10.5 数组应用：矩阵的有关计算 224
10.5.1 矩阵元素求和 224
10.5.2 矩阵的加法 225
10.5.3 矩阵的乘法 227
10.6 练习题 228
第 11 章 代理 231
11.1 代理的声明 231
11.2 代理的实例化 233
11.3 MulticastDelegate 类 234
11.4 代理的调用 240
11.5 代理和事件 242
11.5.1 创建包含事件成员的类 243
11.5.2 创建包含事件处理方法的类 247
11.6 练习题 250
第 12 章 命名空间 256
12.1 概述 256
12.2 命名空间的声明 258
12.3 使用 using 语句 260
12.4 .NET Framework 类库中的命名空间 263
12.5 示例程序 266
12.6 练习题 269
第 13 章 属性 272
13.1 概述 272
13.1.1 属性目标 273
13.1.2 属性的参数 275
13.2 全局属性 276
13.3 保留属性 279
13.3.1 ConditionalAttribute 属性 279
13.3.2 ObsoleteAttribute 属性 280
13.4 自定义属性 282
13.4.1 Attribute 类 282
13.4.2 创建自定义属性 282
13.5 检索属性信息 287
13.6 练习题 292
第 14 章 异常处理 297
14.1 概述 297
14.2 .NET Framework 中的异常类 298
14.2.1 Exception 类 299
14.2.2 常用异常类 299
14.3 C#中的异常处理 300
14.3.1 使用 try/catch 语句来捕获异常 300
14.3.2 使用 throw 语句抛出异常 302
14.3.3 使用 finally 语句抛出异常 304

14.3.4 创建自定义的异常类 305
14.4 异常处理中的注意事项.....305
14.5 练习题 306
第 15 章 指针和不安全代码.....310
15.1 不安全上下文.....311
15.2 unsafe 关键字.....311
15.3 fixed 语句.....313
15.4 指针 315
15.4.1 访问指针成员 316
15.4.2 指针运算 316
15.4.3 在调用栈中分配内存 318
15.4.4 指针类型转换 318
15.5 练习题 322
第 16 章 输入和输出.....325
16.1 控制台输入和输出.....325
16.1.1 控制台输入 325
16.1.2 格式化字符串 327
16.1.3 控制台输出 330
16.2 文件输入/输出 331
16.2.1 Stream 类.....331
16.2.2 FileStream 类.....333
16.2.3 用于读写数据的类 334
16.2.4 文件和目录类 337
16.2.5 Path 类.....341
16.2.6 示例程序 343
16.3 练习题 347
第 17 章 开发 Windows 应用程序 350
17.1 Windows 编程基础 350
17.2 .NET Framework 体系结构.....351
17.2.1 公共语言运行库..... 351
17.2.2 .NET Framework 类库 352
17.2.3 .NET Framework 编程语言 353
17.3 编写简单的 Windows 应用程序..... 353
17.3.1 设计应用程序窗体..... 353
17.3.2 向窗体中添加控件..... 356
17.3.3 为控件添加事件..... 360
17.3.4 通用对话框 363
第 18 章 开发 ASP. NET Web 应用程序..... 367
18.1 ASP.NET Web 应用程序简介 367
18.2 创建一个简单的 Web 应用程序..... 368
18.2.1 创建 Web 应用程序窗体 368
18.2.2 向窗体中添加控件..... 369
18.2.3 为控件添加功能..... 370
18.2.4 运行 Web 应用程序 371
18.3 Web 窗体简介 372
18.3.1 Web 窗体概述 373
18.3.2 Web 窗体代码模型 374
18.3.3 Web 窗体页面处理过程 376
18.3.4 Web 窗体状态管理 379
18.4 ASP.NET 服务器控件及事件处理.. 381
18.4.1 ASP.NET 服务器控件介绍 381
18.4.2 向 Web 窗体页面中添加服务器控件 382
18.4.3 ASP.NET 服务器控件事件处理 382

第 1 章　初识 C#

本章简单介绍了 C#语言的背景和特点，并且对 Visual Studio .NET 的集成开发环境做了介绍。在此基础上，通过一个实际的 C#程序“Helloworld!”，介绍了编辑、编译、运行 C#程序的方法，让读者对 C#有个直观的印象。

学习重点：

- 认识 C#语言
- 熟悉 Visual Studio .NET 集成开发环境
- 创建第一个 C#程序

1.1　C#语言简介

C#是在 C++的基础上发展起来的一种纯面向对象的跨语言平台的程序设计语言。C#是 Microsoft .NET 平台的首选开发语言。

1.1.1　程序设计与程序设计语言

计算机系统由硬件和软件组成。软件控制着硬件，完成各种功能。

现代的软件理论认为，计算机软件是计算机程序和与程序配套的技术文档、用户文档的统一体，但从功能上说，控制计算机硬件电路完成各种任务的只有计算机程序。编写计算机程序，就能控制计算机，让计算机为我们服务。

计算机程序是使用计算机程序设计语言书写的，而计算机程序设计语言是一套用来控制计算机的命令集，可以分为机器语言、汇编语言和高级语言 3 种。

1. 低级语言

机器语言和汇编语言统称低级语言，都是面向具体硬件系统的编程语言，在不同的硬件环境下，低级语言的命令集是不同的。所以，用低级语言在一种 CPU 系统上编写的程序就无法在另一种 CPU 上运行。

在各种语言编写的程序里，用机器语言编写的程序能被计算机直接识别。机器语言程序由二进制机器码组成，很难学习和记忆，但是运行速度最快；汇编语言是机器语言的变种，它采用某些简单易记的符号代表难记的机器码。相对于机器语言来说，汇编语言学习起来较为容易。但是，汇编语言运行时需要有一个翻译程序，先将汇编语言写的程序翻译成机器语言，然后由计算机直接运行。低级语言的优点在于运行速度快，但是低级语言程序不容易编写、不容易排错，几乎无法移植。所以，除了某些对时序要求很高的场合外，

一般不提倡使用低级语言编写程序。

2. 高级语言

高级语言接近于数学语言，也混合了一些人类语言的元素，具有易学易用的优点。但是，用高级语言编写的程序是不能被机器直接识别的。要运行用高级语言编写的程序，要么必须通过编译软件将其“翻译”成机器语言程序然后再运行；要么通过解释软件边解读程序，边告诉计算机该做什么。这两种运行高级语言程序的方法分别称为编译执行和解释执行。使用高级语言编写的程序文件被称为“源代码文件”，简称“源程序”或“源代码”。

一般地说，采用编译执行的方法运行程序在速度上要比采用解释执行的方法快，但采用解释执行的方法比较容易进行程序调试。目前，各种高级语言普遍采用了编译执行的方法，只有少数几种采用了解释执行的方法。

相对于低级语言而言，高级语言的缺点在于程序运行速度慢，并且对系统的资源(内存空间)占用大。但是，随着硬件技术的提高、硬件价格的下降，这些缺点早已不是程序设计中最主要的矛盾，所以，现代的应用软件开发使用的基本上是高级语言。

从计算机诞生到现在的近 50 年间，计算机科学家设计出了大量不同的高级语言，以满足不同领域的需要。在早期的高级程序设计语言中，著名的有面向初学者的 Basic 语言、工程计算用的 Fortran 语言、具有数学般严谨风格的 Pascal 语言和高效灵活的 C 语言等，其中影响面最广的当属 C 语言。

C 语言最初的版本是贝尔实验室的 Dennis Ritchie 等在 1972 年完成的。设计 C 语言的初衷是为了改写 UNIX 操作系统，由于 UNIX 系统的成功，C 语言也就随之流行开来。C 语言的主要特点在于语法简洁、灵活，运行效率高，甚至可同低级语言相媲美，再加上丰富的运算符和数据结构，使 C 语言成为深受广大程序员喜爱的高级程序设计语言。

近年来，随着面向对象方法的兴起，又出现了如 C++、Smalltalk、Java 等面向对象的程序设计语言，但从面向对象的角度来看，它们都是不完整的。而 C#是目前最新的、真正面向对象的程序设计语言。

1.1.2 C#语言产生的背景

在因特网技术高速发展的今天，网络与计算机技术成了世界的中心，计算机网络成了人类生活中必不可少的组成部分。

作为世界上最著名的操作系统开发商，Microsoft 公司提出了.NET 战略，希望把计算机软件变为因特网上的一种服务。下一代的 Windows 操作系统将全面装备.NET 平台，让个人计算机同因特网连接更加紧密。

C#语言就是在这样的背景下推出的。作为.NET 平台下的首选开发语言，C#的设计者为开发人员提供了一个简单、现代、面向对象的语言环境，为开发稳定、坚固和高效的应用软件奠定了基础。

C#语言的前身是 C++语言。直到今天，C++语言仍然是面向对象程序设计语言事实上的标准。但是，C++语言的很多优点也是它致命的缺点，比如指针操作引起的不安全因素，内存回收需要程序员介入等，使得用 C++开发软件的困难程度比其他语言要高得多。C#抛

弃了 C++中大多数复杂的、不安全的元素，牺牲了 C++的某些灵活性换取了软件开发的高效性和简单性。

1.1.3　C#语言的特点

首先，C#是真正的面向对象程序设计语言。在 C#中，不再有函数或者过程，只有类的方法，这同面向过程/面向对象混合型的 C++有着很大的不同。纯面向对象意味着程序员可以完全使用面向对象的思维方法来设计系统，从而为开发大型软件提供可能。

C#是一种比较简单的程序设计语言，它没有 C++中那些容易引发问题的编程难点，也比 C++容易学习和掌握。

C#是在.NET 平台下实现的编程语言，所以，它在很多方面同.NET 平台是相辅相成的。比如，跨语言的异常处理、遵守公用语言规范、使用公共语言运行环境等。

在.NET 平台下，程序被编译成中间语言代码，当程序运行时，才由中间语言代码编译成目标机器上的机器码运行。这种方式被称为“即时编译(Just-in-time compilation)”，是公共语言运行环境的一大特色。通过这一技术，C#写的程序只需要经过一次编译，就可在不同平台上运行。

虽然 Java 程序也只需要经过一次编译即可在不同平台上运行，但是同 Java 的虚拟机技术相比，即时编译技术是真正的编译，从而能够获得高速的程序运行结果。

1.2　熟悉 Visual Studio .NET 集成开发环境

Visual Studio .NET 集成开发环境是.NET 应用程序的开发平台，它是一个功能强大的集源程序编辑、编译、调试、管理等功能为一身的集成工具，能够为.NET Framework 支持的任何一种语言，如 Visual Basic .NET、Visual C++ .NET、Visual C#提供开发环境。

1.2.1　集成开发环境

集成开发环境是目前计算机语言产品都具备的一种工作环境，是进行程序设计的工作场所。在集成开发环境中，程序员可以对源程序进行编辑和编译，对目标程序进行调试运行。

Visual Studio .NET 集成开发环境将 Visual Basic .NET、Visual C++ .NET 等开发工具集合在同一个界面中，从而可以方便地编写、编译和调试各种程序。

通常，除了集成开发环境外，计算机语言产品还会提供命令行环境，程序员可以在命令行界面下通过编译命令对某个程序进行编译。集成开发环境提供了很多便利，但是运行时占用的资源也多；而命令行环境则相对来说显得小巧，但是不方便使用。

1.2.2　打开 Visual Studio .NET 集成开发环境

本书不介绍 Visual Studio .NET 的安装。用户可参考 Visual Studio .NET 的用户文档安

装 Visual Studio .NET。Visual Studio .NET 安装成功后，【程序】菜单中会出现 Microsoft Visual Studio .NET 子菜单。

要启动 Visual Studio .NET 的集成开发环境，可选择【程序】|Microsoft Visual Studio .NET|Microsoft Visual Studio .NET 命令。Visual Studio .NET 集成开发环境的界面如图 1.1 所示。

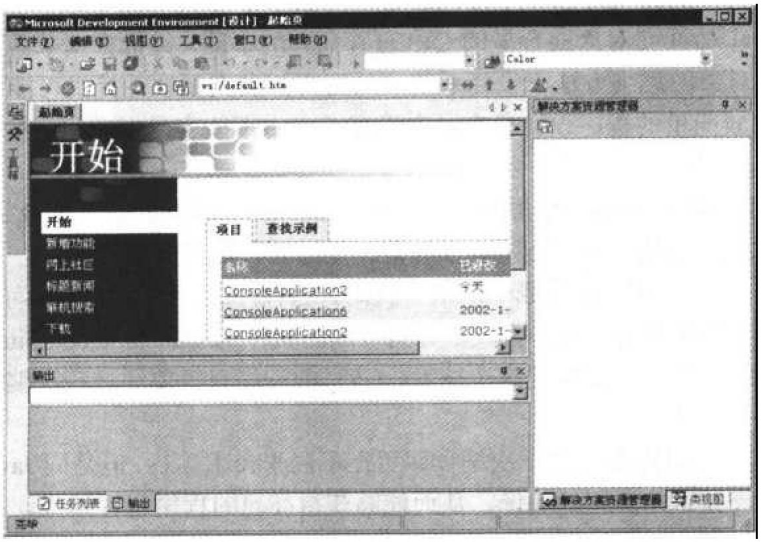


图 1.1 Visual Studio .NET 集成开发环境界面

1.2.3 Visual Studio .NET 集成开发环境界面介绍

Visual Studio .NET 集成开发环境中最重要的是【解决方案资源管理器】窗口。在【解决方案资源管理器】窗口中，可以从不同角度对整个项目的文件进行管理。

在对界面上的元素进行编辑时，【属性】窗口将列出该元素的各种属性，并提供修改属性的手段。

【输出】窗口将在程序编译、运行时，输出编译和调试信息。

Visual Studio .NET 界面上的各个窗口都是可停靠的窗口，可以方便地将其停靠在界面上，或者让其作为普通窗口显示。

1.3 C#版的 Helloworld 程序

Helloworld 程序是一个很简单的程序，它将在命令行提示符下显示“Hello World！”这几个字符。

以下通过建立一个 C#版的 Helloworld 程序，来熟悉使用 Visual Studio .NET 集成开发环境建立、编译、运行一个 C#程序和使用命令行环境来编译运行一个 C#程序的方法，并且初步了解 C#程序的结构。

1.3.1　在集成开发环境下建立 Helloworld 程序

建立 Helloworld 程序步骤如下：

(1) 启动 Visual Studio .NET 集成开发环境。

(2) 选择【文件】|【新建】|【项目】命令，弹出【新建项目】对话框，如图 1.2 所示。

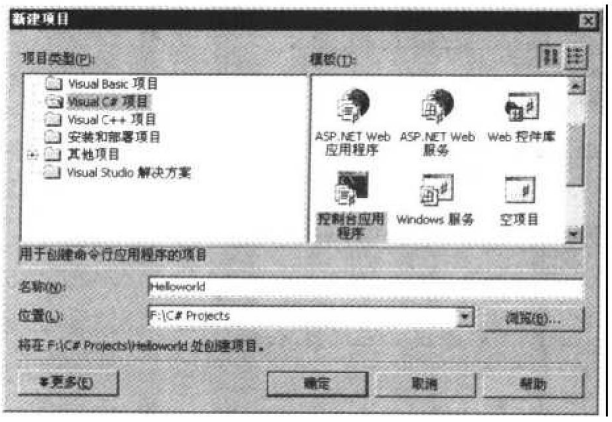


图 1.2　【新建项目】对话框

(3) 在【新建项目】对话框的【项目类型】列表中选择【Visual C#项目】选项，然后在【模板】列表中选择【控制台应用程序】选项。在【名称】文本框中，为新建立的项目起一个名字，这里可以输入“Helloworld”，在【位置】下拉列表框中，可选择一个保存项目文件的文件夹。

(4) 单击【确定】按钮，Visual Studio .NET 集成开发环境即生成一个名为 Helloworld 的新解决方案(也就是项目)，这时候 Visual Studio .NET 集成开发环境的界面如图 1.3 所示。

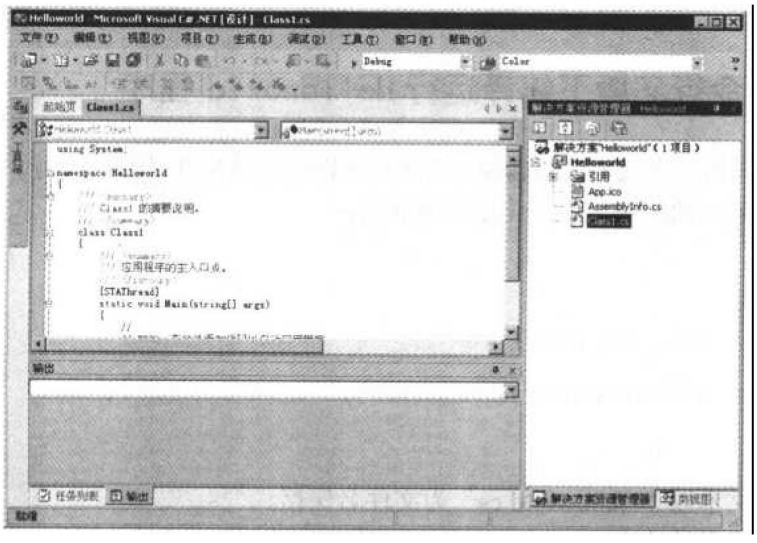


图 1.3　Helloworld 项目

(5) 在 Class1.cs 程序编辑窗口中，Visual Studio .NET 自动生成了一个代码框架。按照下面列出的程序代码，修改 Class1.cs 程序编辑窗口中的代码。其中**加粗显示**的是新添加的代码。

```
namespace helloworld
{
    using System;

    /// <summary>
    ///    Summary description for Class1.
    /// </summary>
    public class Class1
    {
        public Class1()
        {
            //
            // TODO: Add Constructor Logic here
            //
        }

        public static int Main(string[] args)
        {
            //
            // TODO: Add code to start application here
            //
                Console.Write("Hello World!");
            return 0;
        }
    }
}
```

(6) 选择【项目】|【生成】命令，Visual Studio .NET 将开始编译 Helloworld 项目。在【输出】窗口中，将显示解决方案的生成过程(也就是对程序的编译过程)。

(7) 编译成功后，选择【调试】|【开始】命令，Helloworld 程序将被执行。可以看到命令行窗口被打开，字符串"Hello World!"显示在命令行窗口中，然后命令行窗口关闭。

1.3.2 在命令行环境下编译运行 Helloworld 程序

首先，使用任何一个文本编辑器输入 1.3.1 小节的步骤 5 中列出的代码。实际上，1.3.1 小节的步骤 5 中列出的代码可以简化成如下代码：

```
using System;
class Class1
{
      public static void Main()
      {
            Console.Write("Hello World!");
      }
}
```

将编辑好的文本以“Helloworld.cs”为文件名保存。

进入命令行提示符状态(如果在 Win 9x 下请打开 MS-DOS 方式)，输入：

```
csc helloworld.cs
```

C#编译器将编译 Helloworld 程序，并生成可执行文件 Helloword.exe。

运行 Helloworld.exe，可看到同集成开发环境下相同的效果。

1.3.3　C#程序结构分析

下面用 1.3.2 节中列出的简化代码来分析 Helloworld 程序，并借此来分析一下 C#应用程序的结构。

在 Helloworld 程序中，在屏幕上显示“Hello World!”这几个字符的程序段就是代码列表中**加粗显示**的代码：

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

这里的 Console 是位于 System 命名空间中的一个对象，它代表系统控制台，即字符界面的输入和输出。Write 是 Console 对象的一个方法，它的作用是在屏幕上显示字符串。通过调用 Console 对象的 Write 方法，就能将任意字符串显示在屏幕上。

为了显示“Hello World!”，使用“Hello World!”字符串作为参数，调用 Console.Write 方法，当执行到这条代码时，“Hello World!”就能显示在屏幕上。

C#程序用命名空间来组织代码，要访问某个命名空间中的类或对象，必须用如下语法：

```
命名空间.类名
```

由于 Console 对象位于 System 命名空间中，所以实际上用户在访问 Console 对象时，完整的写法应该是：

```
System.Console
```

但是，在程序的第一行，使用了：

```
using System;
```

这条语句告诉 C#编译器，本程序中可以直接使用 System 命名空间中的类和对象，所以要访问 Console 对象，就可以不用写 System.Console，直接写 Console 即可。

程序的入口从下面的代码开始：

```
public static void Main()
```

这行代码所定义的其实是类 class1 的一个静态方法，C#规定，名字为 Main 的静态方法就是程序的入口。当程序执行时，就直接调用这个方法。这个方法包含一对花括号“{”和“}”，在这两个括号间的语句就是该方法所包含的可执行语句，也是该方法所要执行的功能，本例中该方法要执行的功能就是输出“Hello World!”字符串。方法的执行从左括号“{”开始，到右括号“}”结束。

从上面的程序中还可看到，Main 方法的所有部分都是包含在另一对花括号中的，这是类 class1 所带的一对括号，该括号中所有部分都是 class1 类的成员。与 C++不同的是，在 C#中，表示程序执行入口点的 Main 方法必须包含在一个类中，因而每个可执行的 C#程序都至少会包含一个类和一个 Main 方法。

通过上面的分析，可以看出 C#程序的基本结构如下：

```
using System;                                    //引入命名空间
class Class1                                     //定义类
```

```
{
public static void Main()                    //定义Main()方法
{
    Console.Write("Hello World!");           //标准输出
}
}
```

第 2 章　C#程序设计基础知识

本章讲述同 C#程序设计有关的基础知识，包括面向对象理论和 C#语言中最基本的语法结构，比如常数和变量等。

本章所讲述的知识中有很大一部分只是简单介绍，在以后的章节中，这些知识点还将重复出现。这是由于 C#是一种从 C++进化而来的语言，比较适合已经具备 C++编程基础的程序员使用，对于从未接触过面向对象程序设计的初级用户，有些内容如果不提前介绍，就连最简单的 C#程序也无法讲解。在知识的系统性和循序渐进性的选择上，本书只能选择后者，以方便大多数读者的需要。因而本章最开始部分介绍了一下面向对象程序设计方法的知识，以为读者进一步深入学习 C#打下基础。

学习重点：

- 认识面向对象程序设计方法
- 了解 Unicode 编码
- 掌握标识符的概念
- 掌握 C#中简单类型、常数及字符串的含义及声明格式
- 了解 C#中的各种类型及其实例
- 掌握 C#中的各种运算符及表达式

2.1　面向对象方法简介

C#是面向对象的程序设计语言。面向对象的程序设计语言是建立在面向对象方法理论基础上的。在学习 C#之前，有必要了解一些同面向对象方法有关的知识，如类、对象、属性和方法、继承、多态等。这些看似深奥的理论知识其实只是对客观世界的一种抽象，只要同现实生活中的例子相结合，就很容易理解和掌握它们。

1. 类

在现实生活中，常给某一类事物冠以同样的名字。如“电子计算机”一词是对所有使用电子电路完成数据采集、运算(加工)和存储的机器的总称。虽然不同的电子计算机可能拥有不同的特征，比如作者现在用来写书的这台计算机是一台装备了 Windows 操作系统的微型计算机，而气象台中用来进行天气分析的计算机可能安装的是 UNIX 操作系统，其运算速度也要比作者的计算机快许多，但是这两台计算机是同属于一种类型的事物：“电子计算机”，它们有区别于其他事物的共同特征：“使用电子电路完成数据采集、运算(加工)和存储”。

在面向对象理论中，类(class)就是对具有相同特征的一类事物所做的抽象(或者说，归