新世纪高职高专实用规划教材

● 计算机系列

Visual C# 程序设计基础教程

Visual C# CHENGXU SHEJI JICHU JIAOCHENG

邵鹏鸣　编著

清华大学出版社

新世纪高职高专实用规划教材　计算机系列

Visual C＃程序设计基础教程

邵鹏鸣　编著

清华大学出版社
北　京

内 容 简 介

微软的.NET战略是一场软件革命，它改变了开发人员开发应用程序的方式及思维方式，使得开发人员能创建出各种全新的应用程序。C#是微软公司推出的新一代编程语言，它功能强大、编程简洁明快，是微软.NET战略的重要组成部分。本书共分12章，通过大量的与实际程序设计有关的实例深入浅出地讲解了C#程序设计的基本方法、技巧及注意事项，并注重培养学生编写实际应用程序的能力，帮助学生关注编写程序的重要环节及过程，养成良好的编程习惯，避免犯某些常见的错误。全书贯穿了面向对象编程的程序设计思想和设计方法，并用一整章的篇幅讨论使用ADO.NET和SQL访问数据库的编程技术。

本书内容丰富、可操作性强、语言生动流畅、没有晦涩的术语，用实例说明，能够使学生轻松愉快地掌握C#的基本编程方法和技巧。

本书可作为高职高专院校计算机专业学生的教材，也可作为初中级读者和培训班学员学习的教材。

图书在版编目(CIP)数据

Visual C#程序设计基础教程/邵鹏鸣编著. —北京：清华大学出版社，2005.4
(新世纪高职高专实用规划教材　计算机系列)
ISBN 7-302-10513-8

Ⅰ.V…　Ⅱ.邵…　Ⅲ.C语言—程序设计—高等学校：技术学校—教材　Ⅳ.TP312

中国版本图书馆CIP数据核字(2005)第012383号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦
http://www.tup.com.cn　　**邮　　编**：100084
社 总 机：010-62770175　　**客户服务**：010-62776969
组稿编辑：林章波
文稿编辑：张　莉
封面设计：陈刘源
排版人员：王　婷
印 刷 者：北京市清华园胶印厂
装 订 者：三河市新茂装订有限公司
发 行 者：新华书店总店北京发行所
开　　本：185×260　**印张**：27.25　**字数**：646千字
版　　次：2005年4月第1版　2005年4月第1次印刷
书　　号：ISBN 7-302-10513-8/TP·7139
印　　数：1～4000
定　　价：35.00元

《新世纪高职高专实用规划教材》序

编写目的

目前，随着教育改革的不断深入，高等职业教育发展迅速，进入到一个新的历史阶段。学校规模之大，数量之众，专业设置之广，办学条件之好和招生人数之多，都大大超过了历史上任何一个时期。然而，作为高职院校核心建设项目之一的教材建设，却远远滞后于高等职业教育发展的步伐，以至于许多高职院校的学生缺乏适用的教材，这势必影响高职院校的教育质量，也不利于高职教育的进一步发展。

目前，高职教材建设面临着新的契机和挑战：

(1) 高等职业教育发展迅猛，相应教材在编写、出版等环节需要在保证质量的前提下加快步伐，跟上节奏。

(2) 新型人才的需求，对教材提出了更高的要求，即教材要充分体现科学性、先进性和实用性。

(3) 高职高专教育自身的特点是强调学生的实践能力和动手能力，教材的取材和内容设置必须满足不断发展的教学需求，突出理论和实践的紧密结合。

有鉴于此，清华大学出版社在相关主管部门的大力支持下，组织部分高等职业技术学院的优秀教师以及相关行业的工程师，推出了一系列切合当前教育改革需要的高质量的面向就业的职业技术实用型教材。

系列教材

本系列教材主要涵盖以下领域：

- 计算机基础及其应用
- 计算机网络
- 计算机图形图像处理与多媒体
- 电子商务
- 计算机编程
- 电子电工
- 机械
- 数控技术及模具设计
- 土木建筑
- 经济与管理
- 金融与保险

另外，系列教材还包括大学英语、大学语文、高等数学、大学物理、大学生心理健康等基础教材。所有教材都有相关的配套用书，如实训教材、辅导教材、习题集等。

教材特点

为了完善高等职业技术教育的教材体系，全面提高学生的动手能力、实践能力和职业技术素质，特意聘请有实践经验的高级工程师参与系列教材的编写，采用了一线工程技术人员与在校教师联合编写的模式，使课堂教学与实际操作紧密结合。本系列丛书的特点如下：

(1) 打破以往教科书的编写套路，在兼顾基础知识的同时，强调实用性和可操作性。

(2) 突出概念和应用，相关课程配有上机指导及习题，帮助读者对所学内容进行总结和提高。

(3) 设计了“注意”、“提示”、“技巧”等带有醒目标记的特色段落，使读者更容易得到有益的提示与应用技巧。

(4) 增加了全新的、实用的内容和知识点，并采取由浅入深、循序渐进、层次清楚、步骤详尽的写作方式，突出实践技能和动手能力。

读者定位

本系列教材针对职业教育，主要面向高职高专院校，同时也适用于同等学历的职业教育和继续教育。本丛书以三年制高职为主，同时也适用于两年制高职。

本系列教材的编写和出版是高职教育办学体制和运作机制改革的产物，在后期的推广使用过程中将紧紧跟随职业技术教育发展的步伐，不断吸取新型办学模式、课程改革的思路和方法，为促进职业培训和继续教育的社会需求奉献我们的力量。

我们希望，通过本系列教材的编写和推广应用，不仅有利于提高职业技术教育的整体水平，而且有助于加快改进职业技术教育的办学模式、课程体系和教学培训方法，形成具有特色的职业技术教育的新体系。

教材编委会

新世纪高职高专实用规划教材
计算机系列编委会

前　言

随着21世纪的到来，计算机技术的发展更加迅猛，在各行各业的应用更加广泛，面对日新月异的新技术、新方法，我们必须对现有计算机课程的设置和教学内容进行调整，以适应技术进步和市场变化的需要，使学生所学到的东西是市场上最需要的。

如今，应用程序已由驻留在用户硬盘上的独立可执行文件发展为由 Web 服务器在 Internet 上传送的分布式应用程序；相应地，任何一种开发平台及程序设计语言都必须适应这种变化。微软的.NET 是一种开发平台，C#是微软特别为.NET 平台设计的一种现代编程语言，.NET 有着广阔的应用前景，.NET 的应用必将对整个计算机产业产生重要而深刻的影响。.NET 不但改变了开发人员开发应用程序的方式和思维方式，而且使开发人员能创建出各种全新的应用程序，大幅度提高软件生产率。未来.NET 将无处不在。

C#是一种简单、现代、面向对象且类型安全的编程语言，C# 语言从 C 和 C++ 语言演化而来，C# 同时具备应用程序快速开发(RAD) 语言的高效率和 C++ 固有的强大能力。同时它吸收了 Java 语言的特点和精华，熟悉 Java 的人就觉得它很像 Java，比尔·盖茨曾说过："Java 是最卓越的程序设计语言！"，不过从 C#诞生的那一时刻起，这已成为过去。C#是微软将 Java 集成到.NET 中的产物，它是整个 .NET 平台的基础，是未来主流的编程语言。

我们知道 C 语言是一种代码效率很高但不易于进行快速开发的程序设计语言。OOP 出现后，OOP 与 C 语言融合产生了 C++，继而有了集成开发环境。但 C++的出现，并没有使 C 语言家族在应用开发方面获得突飞猛进的发展。基本上还是占据着 UNIX、Linux 以及底层应用开发的天地，而在 Windows 大型应用软件特别是数据库和 Web 开发上，由于其固有的复杂性和缺乏针对性，就不如 VB 等具有很强针对性的开发工具。使用过包括 C 和 C++ 在内的多种程序设计语言的人，一定会深切体会到它们之间的区别。比如与 Visual Basic 相比，Visual C++程序员为实现同样的功能就要花费更长的开发周期。

另一方面，虽然 C 和 C++为我们带来高度的灵活性，但必须要忍受学习的艰苦和开发的长期性，特别对 VC++来说，大部分的程序结构都被封装在 MFC 中。所以对于初学者来说，程序结构显得十分混乱，学习将变得十分艰苦。而且自从 VC++ 2.0 以来，为了适应不断更新的技术(例如 COM、ATL 等)，又要与前一个版本兼容，VC++在此之后的每一次升级都给 VC++程序员带来一份痛苦——程序结构变得越来越复杂，而且出现了越来越多的变量类型，从而带来了更多的问题。

C#的出现弥补了 C 语言家族的上述不足，它借鉴了 Java、C++、C 甚至 VB 的优点，因而 C#具有 C、C++的强大功能，具有 Java 那样的面向对象机制和虚拟码，具有 VB 开发的高效性和方便性。

学习一种先进的语言和一种先进的编程方法(面向对象的编程方法)将激发学生更大的兴趣。这些知识在他们离开学校，进入一个由 Internet 和 WWW 占据重要地位的世界时可以立即发挥作用。正是这一点激发了他们对知识的学习热情。当今的学生必须同时掌握基础语言、面向对象编程和类库，而 C#课程适合这类知识的学习。运用 C#能够出色地完成任务，所以他们更愿意投入更多的精力和时间。

作者立志于写一本实用、有一定的深度且易读的程序设计语言教材，无论学生有无编程经验，要做到开卷有益，最重要的是能够激发他们的学习兴趣。

本书是从应用和工程实践的角度出发组织和编写的，其主要特点如下：

(1) 新体系、新内容、新手段、新思路。无论是内容体系、编写教材的思路、教学的模式及理念等都具有一定的新意。

(2) 先进性及实用性：本书的内容反映最新、最实用的程序设计方法及技术，顺应并符合新世纪教学发展的规律，书中讲授的程序设计方法与现代编程方法步调一致，具有很强的实用性。

(3) 本书不是采用传统的“提出概念→解释概念→举例说明”的方法，而是以实例为主线通过完成任务的方式学习程序设计知识。本书采用“提出任务→介绍完成任务的方法及步骤→最后归纳出一般规律、概念及知识点”的模式。每一个新概念、知识点的提出都伴随着一个完整的、可实际运行的实用程序及其输入、输出。学生通过完成实例，即可掌握概念和知识点。

(4) 使学生从一开始就编写有用的程序，这有助于保持学生的学习兴趣及动力。书中的实例不是只有几行代码的小程序，这些实例来源于实际应用程序，它们与我们的生活相关，是大家感兴趣的内容。实例包含的知识点可能跨越若干章节。本书包含大量这样的实例，对这些实例及其代码的分析和讨论是本书的精髓。

(5) 逐步展现主题鲜明的各章。本书集中于重要的主题并进行充分论述，而不是肤浅地涉及许多问题，罗列许多概念、术语、语法。本书的内容及体系结构使学生感到它既具有易读性又增进知识，具有很强的实用性。

(6) 学习程序设计语言的目的是为了开发程序，本书不只是讲授C#的语言要素、语法，而是教会学生如何用C#开发程序，书中的每一个实例都说明了开发过程。通过学习如何开发程序来掌握语言要素、语法。

(7) 本书从始至终贯穿面向对象编程的思想，目标之一是使学生习惯于现实世界的程序设计，仅仅知道面向对象的概念是不够的，学生们必须能够运用这些知识开发实际使用的程序。教学概念的核心之一是：在成为对象的设计者之前，必须首先成为对象的用户。换句话说，在有效地设计自己的类之前，必须首先学会使用预定义的类，我们从第一个程序开始就接触如何使用类。学习使用类库中的类和编写自己的类相辅相成，互相促进，学习使用类库中的类有助于学习编写自己定义的类，学习自定义类又有助于学会使用类库中的一些类，并加深对类库中的一些类的理解。当今学生必须同时掌握基础语言、面向对象编程和类库，本书将这三者结合在一起。

(8) 本书要求学生遵循书中介绍的方法和步骤实际建立实例程序，然后将实例程序进行修改或扩展，并通过对实例的代码进行分析和讨论掌握实例背后包含的概念、原理、知识点和方法等，这是学生学习程序设计最稳妥、最有效、最快捷的途径。

由于水平和时间的关系，书中的错误在所难免。如果发现不当之处，欢迎提出宝贵意见，并将信息反馈给我们，将不胜感激。

邵鹏鸣

2004年10月

目　录

第 1 章　认识 C# 1

1.1　第一个简单的控制台应用程序 1
实例：打印一行文字 1
1.2　简单的 Windows 应用程序 3
实例：在对话框中显示一行文字 3
实例：在文本框中显示一行文字 5
1.3　习题 9

第 2 章　变量与数据类型 10

2.1　变量与常量 10
2.1.1　变量的含义 10
2.1.2　变量声明 11
实例：计算路程 11
2.1.3　常数 15
2.2　基本数据类型 15
2.2.1　整型 17
实例：整数相乘 17
2.2.2　字符数据类型 21
实例：字符检查 22
2.2.3　非整型 24
实例：浮点数相乘 24
实例：贷款计算器 28
2.2.4　隐式数值转换 33
实例：隐式数值转换 34
2.2.5　显式转换 36
2.2.6　算术溢出及显式转换溢出 36
实例：算术溢出及显式转换溢出 36
2.2.7　布尔型 38
实例：数值比较 39
2.3　习题 40

第 3 章　程序流控制 42

3.1　选择语句 42
3.1.1　if 语句 42
实例：考试结果分析 42
3.1.2　if...else 语句 45
实例：猜数游戏 45
3.1.3　条件运算符 47
实例：显示时间 47
3.1.4　if...else if ...else 语句 49
实例：工资发放 49
3.1.5　if 语句的嵌套 51
实例：求数的绝对值 51
3.1.6　switch 语句 54
实例：计算器 54
3.1.7　复合赋值运算符 57
3.1.8　条件逻辑运算符和逻辑运算符 57
3.2　循环语句 58
3.2.1　while 语句 58
实例：计算复利存款(1) 58
3.2.2　do/while 语句 61
实例：计算复利存款(2) 61
3.2.3　for 语句 63
实例：打印字母表及对应的 ASCII 码(1) 63
3.2.4　嵌套循环 66
实例：打印字母表及对应的 ASCII 码(2) 66
3.2.5　增量运算符与减量运算符 67
实例：自增运算 67
3.3　跳转语句 68
3.3.1　goto 语句 68
实例：查询 68

实例：自动售货机 70
3.3.2 break 语句 71
实例：打印字母表及对应的 ASCII 码(3) 71
3.3.3 continue 语句 72
实例：找数 72
3.3.4 运算符的优先级 73
3.4 习题 74

第 4 章 方法与数组 76

4.1 方法 76
4.1.1 方法的定义 76
实例：求 n! 76
4.1.2 方法定义的格式 78
4.2 数组 80
实例：日常开销 80
4.2.1 数组初始化 83
实例：显示月名称 83
4.2.2 数组元素访问 85
实例：显示 0～n 的值 85
4.2.3 数组对象的赋值运算 87
实例：数组对象的赋值 87
4.2.4 值类型与引用类型 90
4.3 向方法传递数组 93
4.4 传递参数：传值方式和传引用方式 93
4.4.1 传值方式 93
实例：传值方式 93
4.4.2 传引用方式 95
实例：传引用方式 96
实例：out 输出参数 97
4.5 多维数组 99
4.5.1 多维数组的声明和创建 100
4.5.2 多维数组初始化 100
实例：三维数组 100
实例：学生考试成绩统计 102
4.6 习题 104

第 5 章 类与对象 106

5.1 类、对象、字段、属性和方法 106
实例：定义 Person 类(1) 106
实例：贷款分析 111
实例：定义矩形(1) 117
5.2 实例构造函数 120
实例：定义 Person 类(2) 120
5.2.1 默认实例构造函数 121
5.2.2 字段初始化 122
5.2.3 构造函数声明 123
实例：定义矩形(2) 123
5.3 实例构造函数重载 125
实例：定义矩形(3) 125
5.4 析构函数 128
实例：定义 size 类 128
5.5 对象成员与类的成员 131
实例：定义 Student2 131
5.6 对象参数与返回值为对象 134
5.6.1 对象参数 135
实例：定义矩形(4) 135
实例：定义矩形(5) 138
5.6.2 返回值为对象 140
实例：定义矩形(6) 140
5.7 方法的重载 142
实例：定义矩形(7) 143
5.8 习题 147

第 6 章 继承 149

6.1 直接基类与派生类 149
实例：定义 Person 149
实例：定义 Student(1) 150
6.2 派生类实例构造函数声明 153
实例：复数加法 154
实例：复数减法 155
6.3 隐藏从基类继承的成员 158
实例：隐藏继承字段 158
6.4 含直接基类构造函数的构造函数声明 159

实例：定义 Student(2)....159
6.5 虚拟方法与重写方法....161
实例：多级继承层次结构
—图形....162
6.6 习题....168
第 7 章 多态性....169
7.1 抽象方法与抽象类....169
7.1.1 抽象方法....169
实例：多态性及实现(1)....169
7.1.2 抽象类....178
实例：多态性及实现(2)....178
7.2 接口....180
实例：薪水发放系统....180
实例：用接口实现不同的度
量衡系统....186
7.3 委托....188
7.3.1 使用委托....188
实例：使用委托实现运算....188
7.3.2 组合委托....191
实例：使用组合委托实现运算....191
实例：用委托排序数组....194
7.4 事件....197
实例：进度指示器....198
实例：具有取消功能的进度
指示器....201
7.5 习题....203
第 8 章 常用控件....205
8.1 滚动条....205
实例：调色板....205
8.1.1 滚动条常用属性....209
8.1.2 滚动条常用事件....210
8.1.3 用户自定义颜色....210
8.2 PictureBox 图片框控件....211
实例：滚动图像....211
8.2.1 PictureBox 的常用属性....217
8.2.2 PictureBox 的常用事件....218
8.2.3 Image 的 FromFile 方法....218
8.3 文本框控件与按钮控件....218
实例：登录程序....219
8.3.1 TextBox 的常用属性....224
8.3.2 TextBox 的常用事件....226
8.3.3 Button 按钮的常用属性....226
8.3.4 Button 按钮的常用事件....227
8.4 Windows 窗体事件及事件
处理程序....227
实例：计算器....227
8.5 复选框和单选按钮....232
实例：font 程序....233
8.5.1 复选框的常用属性....235
8.5.2 复选框的常用事件....236
8.5.3 单选按钮的常用属性....236
8.5.4 单选按钮的常用事件....236
8.6 列表框....236
实例：ListDemo 程序....237
8.6.1 列表框控件的常用属性....242
8.6.2 列表框控件的常用属性
和方法....243
8.6.3 列表框控件的常用事件....244
8.7 带复选框的列表框....244
实例：CheckedListBoxTest 程序....244
8.7.1 复选列表框控件的
常用属性....247
8.7.2 复选列表框控件的常用
方法和事件....248
8.8 习题....248
第 9 章 GDI+图形....250
9.1 第一个绘图程序....250
实例：第一个绘图程序
——画直线....250
9.2 创建 Graphics 对象....251
9.2.1 用 CreateGraphics 方法
创建 Graphics 对象....252
实例：在标签和图像框上画图....252
9.2.2 Paint 事件处理程序
中的 PaintEventArgs....253

实例：填充矩形 254
9.3 笔、画笔和颜色....255
9.3.1 笔 255
实例：创建笔 255
9.3.2 画笔 257
实例：用不同的画笔画图....257
实例：使用图案绘图....259
实例：绘制颜色渐变图形....260
9.4 绘制线条和形状....262
9.4.1 绘制线条 262
实例：绘制直线 262
实例：绘制抛物线 264
9.4.2 绘制多边形和折线....265
9.4.3 绘制矩形 267
实例：绘制由坐标对、宽度和高度指定的矩形....267
实例：绘制一系列由 RectangleF 结构指定的矩形....268
9.4.4 绘制椭圆 270
实例：绘制椭圆 270
9.4.5 绘制弧线和扇形....272
实例：绘制弧线 272
实例：绘制扇形 273
9.4.6 绘制文本字符串....275
实例：绘制格式化字符串....277
9.5 用 GDI+呈现图像 280
实例：呈现图像 280
实例：在指定位置按指定大小绘制指定的图象....282
9.6 画点 283
实例：画点 283
9.7 坐标系统 284
9.7.1 坐标系统类型....284
实例：在不同的坐标系中画直线 286
9.7.2 全局变形和局部变形....288
实例：全局变形图形....288
实例：局部变形图形....290
9.8 全局变形可与局部变形合并....292
实例：全局变形可与局部变形合并....293
9.9 习题 295
第 10 章 与用户交互....297
10.1 菜单....297
10.1.1 创建菜单....297
实例：随机画矩形....297
10.1.2 Timer 控件....301
10.1.3 MainMenu 控件常用属性....301
10.1.4 快捷菜单....301
实例：实现快捷菜单....302
10.2 鼠标事件....303
10.2.1 实例：用鼠标画图....303
10.2.2 鼠标事件....307
10.3 键盘事件处理....309
实例：键盘事件程序....309
10.4 通用对话框....316
10.4.1 【打开文件】对话框....316
实例：打开文件....316
10.4.2 【保存文件】对话框....319
实例：保存文件....319
10.4.3 【字体】对话框....321
实例：改变文本的字体....321
10.4.4 【颜色】对话框....323
实例：改变文本颜色....323
10.5 编写多文档界面应用程序....324
实例：字处理器....324
10.6 习题....334
第 11 章 用流进行文件输入和输出....335
11.1 文件与流....335
11.1.1 FileStream....335
实例：使用 FileStream....335
11.1.2 定位操作....338
实例：定位操作....338
11.1.3 向文件追加数据....340
实例：向文件追加数据....340

11.1.4 StreamReader 和 StreamWriter.....341
实例：电话号码簿.....341
11.2 二进制读取器和写出器.....344
实例：学生名册.....344
11.3 序列化对象.....347
实例：将对象写入文件.....347
11.4 顺序访问文件.....351
实例：员工工资发放程序.....351
实例：创建员工工资发放程序.....358
11.5 随机访问文件.....362
实例：银行客户帐号管理程序.....362
实例：新建一个项目以使用自定义的类库.....371
11.6 习题.....381
第 12 章 使用 ADO.NET 进行数据库编程.....382
12.1 Connection 和 Command 对象.....382
实例：基于 C/S 的产品信息管理.....382
12.1.1 Connection 对象.....383
实例：创建和打开一个到 SQL Server 的连接.....383
12.1.2 Command 对象.....385
实例：使用 Command 对象.....385
12.2 使用 ADO.NET DataReader.....387
12.2.1 使用 SqlDataReader 装载列表框.....387
12.2.2 创建普通的列表框类.....390
12.2.3 在 ListLoad()方法中使用 PDSAListItemNumeric 类.....391
12.2.4 显示产品的详细信息.....392
12.2.5 装载组合框.....394
12.2.6 在组合框中查找值.....396
12.2.7 修改数据.....399
12.3 使用 DataAdapter、DataTables 和 DataSets.....400
12.3.1 使用 DataTable 对象装载组合框.....401
12.3.2 装载 Categories 组合框.....403
12.3.3 创建 DataSet 对象.....404
12.3.4 使用数据集装载列表框.....405
12.3.5 在数据集的表中查找特定的行.....406
12.4 修改数据.....408
12.4.1 使用数据集向数据库表中添加一行.....408
12.4.2 修改数据集中表中的行.....410
12.4.3 在数据集中删除行.....412
12.4.4 DataAdapter 与 CommandBuilder 的进一步说明.....413
12.5 习题.....414

第 1 章　认识 C#

本章目标

- 熟悉 Visual Studio.NET 集成开发环境(IDE)
- 学会使用 C#创建、编译和执行简单的.NET 应用程序
- 使用输入和输出
- 初步认识和了解窗体、控件、事件和方法

1.1　第一个简单的控制台应用程序

实例：打印一行文字

打印一行文字，实现如图 1.1 所示效果。

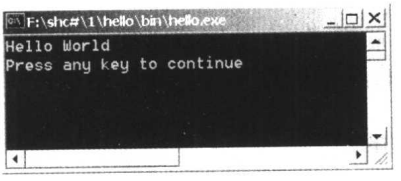

图 1.1　打印一行文字

◆ 实现步骤

(1) 启动 Visual Studio。

(2) 从【文件】菜单上选择【新建】|【项目】打开【新建项目】对话框。

(3) 在【项目类型】窗格中选择【Visual C#项目】，然后在【模板】窗格中选择【空项目】。

(4) 在【名称】框中，键入 Hello 作为该项目的名称。在【位置】框中，输入要将项目保存到的目录，或单击【浏览】按钮选择目录。

(5) 单击【确定】。

Visual Studio 将创建一个新项目“Hello”，并显示解决方案资源管理器。若没有显示解决方案资源管理器，请选择【视图】|【解决方案资源管理器】，或按 Ctrl+Alt+L，解决方案资源管理器即可出现。

(6) 在解决方案资源管理器中，右击解决方案下的 Hello 项目，在弹出的快捷菜单中选择【添加】|【添加新项】。

(7) 在【类别】窗格中选择【本地项目】，然后在【模板】窗格中选择【代码文件】。在【名称】框中，键入 TESTHello。

(8) 单击【打开】。然后在空的 TESTHello 代码文件中输入如下代码：

```
//第一个简单的C#控制台应用程序
class Hello
{
    static void Main()
    {
        System.Console.WriteLine("Hello, World ");
    }
}
```

(9) 按 F5 键运行该应用程序，在命令行窗口中显示“Hello, World!”。

(10) 单击任意键结束程序运行。

◆ 代码分析与讨论

(1) 代码注释

第一行包含注释语句：

```
//第一个简单的C#控制台应用程序
```

其中“//”字符将这行的其余内容转换为注释内容。可以将整行作为注释，或者可以在其他语句的结尾追加一个注释，如下所示：

```
System.Console.WriteLine("Hello World ")  //输出Hello World
```

程序员在程序中加入注释，可以提高程序的可读性，使程序易于阅读和理解。计算机在执行程序时不会执行注释行。以“//”开始的注释叫“单行注释”，它只对当前行有效。以“/*”开始并以“*/”结束的注释称为多行注释。例如：

```
/* 这是多行注释，
  第一个简单的C#控制台应用程序 */
```

(2) 定义类

C#的每一个程序包括至少一个自定义类。这些类称为程序员自定义类或用户自定义类。在 C#中用关键字 class 引导一个类的定义，其后接着类的名称(本例中是 Hello)。关键字是 C#的保留字。class Hello 后的左侧“{”表示开始一个类的定义，对应的右侧“}”用来结束类的定义(如果花括号不成对出现，会出现编译错误)。例如：

```
class Hello
{
    ...
}
```

(3) Main 方法

C#程序必须包含一个 Main 方法，而且必须按第 3 行那样定义，Main 方法是程序的入口点，程序控制在该方法中开始和结束。该方法用来执行任务，并在任务完成后返回信息。void 关键字表明该方法执行任务后不返回任何信息。

Main 方法在类的内部声明，它必须具有 static 关键字，表明是静态方法，第 5 章将讨论静态方法。在 "Hello World!" 示例中，Main 方法是 Hello 类的成员。

左侧“{”开始定义方法的主体内容，对应的右侧“}”用来结束方法的定义。

(4) 输入和输出

程序通常使用 .NET 框架的运行时库提供的输入/输出服务。Main 方法中有语句：

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

该语句的作用是使计算机打印双引号之间的字符串，我们将双引号之间的字符通常称为字符串。

该语句中使用了 WriteLine 方法，它是类库中 Console 类的输出方法之一，WriteLine 方法在命令窗口中显示一行文字后，自动将光标移动到下一行。

如下代码段使用了 ReadLine 方法：

```
string  Str;
Str=System.Console. ReadLine ( );
```

ReadLine 方法是运行时库中 Console 类的输入方法之一，它用于输入一字符串，按回车键结束输入。其他 Console 方法用于不同的输入和输出操作。

如果在程序开头包含以下 using 语句：

```
using System;
```

则可直接使用 Console 类和方法，无需使用完全限定名。例如：

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

using System 语句引用一个由 Microsoft .NET 框架类库提供的、名为 System 的命名空间。此命名空间包含 Main 方法中引用的 Console 类。命名空间提供了一种分层方法来组织一个或多个程序的元素。using 语句可以非限定地使用属于命名空间的类。“hello，world”程序代码中使用 Console.WriteLine 作为 System.Console.WriteLine 的简写形式。

(5) 编译并运行程序

从 IDE 编译并运行程序。按 F5 生成并运行(也可选择【调试】菜单中的【启动】)。

1.2　简单的 Windows 应用程序

前面一个程序是在命令行窗口中显示输出，但大多数 C#程序使用窗口或对话框显示输出。

实例：在对话框中显示一行文字

在对话框中显示一行文字，如图 1.2 所示。

图 1.2　在对话框中显示一行文字

◆ 实现步骤

(1) 启动 Visual Studio。

(2) 在【文件】菜单中选择【新建】|【项目】打开【新建项目】对话框。

(3) 在【项目类型】窗格中选【Visual C#项目】，然后在【模板】窗格中选择【空项目】。

(4) 在【名称】框中，键入 Hello 作为该项目的名称。在【位置】框中，输入要将项目保存到的目录，或单击【浏览】按钮以选择目录。

(5) 单击【确定】。Visual Studio 将创建一个新项目“Hello”，并显示解决方案资源

管理器。

(6) 在解决方案资源管理器中，右击解决方案下的 Hello 项目，在弹出的快捷菜单中选择【添加】|【添加新项】。

(7) 在【类别】窗格中选择【本地项目】，然后在【模板】窗格中选择【代码文件】。在【名称】框中，键入 TESTHello。

(8) 单击【打开】。然后在空的 TESTHello 代码文件中输入如下代码：

```
using  System.Windows.Forms;
class TestHello
{
    static void Main()
    {
        MessageBox.Show("Hello,World!");
    }
}
```

(9) 在解决方案资源管理器中，右击 TestMessageBox 项目下的【引用】，在弹出的快捷菜单中选择【添加引用】。

(10) 选择.NET 选项卡，拖动滚动条，找到 System.Windows.Forms.dll，然后双击它。

(11) 单击【确定】按钮。

(12) 右击 TestMessageBox 项目，在弹出的快捷菜单中选择【属性】。在 TestMessageBox 项目的属性页中，将输出类型设置为【Windows 应用程序】。

(13) 按 F5 键运行该应用程序，可得到图 1.2 所示的输出。

(14) 关闭对话框并返回 Visual Studio。

◆ 代码分析与讨论

(1) C#程序员既要考虑自定义类，也要考虑重用框架类库(FLC)中的类。

C#程序中要使用类库中的类，必须添加包含要使用的类库的程序集的引用。类库中的每个类都从属于特定的命名空间，我们使用 using 语句定位所使用类库中的类的命名空间，以便我们在程序中直接使用该类。例如，要使用类库中的 MessageBox 类，由于 MessageBox 类在命名空间“System.Windows.Forms”中，我们在程序中必须使用 using 语句：

```
using  System.Windows.Forms;
```

以定位 MessageBox 类的命名空间，这样在程序中即可直接使用 MessageBox 类，如：

```
MessageBox.Show("Hello,World!");
```

而不需要像下面这样使用完全限定名：

```
System.Windows.Forms.MessageBox.Show("Hello,World!");
```

若不使用上述 using 语句而直接使用 MessageBox 类，会发生编译错误：“MessageBox 未定义”。

而另一方面，using System .Windows.Forms 必须得到与之相匹配的动态链接库的支持，即必须首先添加“System .Windows. Forms.dll”的引用，否则编译环境就会无法识别。从中可以看到命名空间与.dll 文件的某种关联。

(2) 在 Main 方法中，语句：

```
MessageBox.Show("Hello,World!");
```

表明一个对类 MessageBox 中的方法 Show 的调用，它的作用是在对话框中显示一字符