

C# 2.0 程序设计教程

郑宇军 编著

清华大学出版社

北 京

内 容 简 介

C# 2.0 是第一门真正将泛型思想和对象技术进行完美融合的高级语言。

本书通过丰富的范例全面系统地讲解了 C# 语言的编程技术，并重点介绍了 C# 2.0 的新增特性，其中包括 .NET 框架、C# 语法基础、类型系统、对象程序设计、泛型编程、匿名方法及商业开发等多个层面的内容。每章之后给出了丰富的练习题，为读者进一步巩固和拓展所学知识提供了广阔空间。

本书面向 C# 语言的初中级读者，可作为大中专院校及各类培训机构的程序设计语言教材，对于专业开发人员而言也不失为一本理想的参考书。

版权所有，翻印必究。举报电话：010-62782989 13901104297 13801310933

本书封面贴有清华大学出版社防伪标签，无标签者不得销售。

本书防伪标签采用清华大学核研院专有核径迹膜防伪技术，用户可通过在图案表面涂抹清水，图案消失，水干后图案复现；或将表面膜揭下，放在白纸上用彩笔涂抹，图案在白纸上再现的方法识别真伪。

图书在版编目 (CIP) 数据

C# 2.0 程序设计教程/郑宇军编著. - 北京：清华大学出版社，2004.11

ISBN 7-302-10153-1

.C... . 郑... . C 语言—程序设计

IV. TP312

中国版本图书馆 CIP 数据核字 (2004) 第 135503 号

出 版 者：清华大学出版社

<http://www.tup.com.cn>

社总机：010-62770175

地 址：北京清华大学学研大厦

邮 编：100084

客户服务：010-62776969

组稿编辑：科海

文稿编辑：何武

封面设计：付剑飞

版式设计：科海

印 刷 者：北京市耀华印刷有限公司

发 行 者：新华书店总店北京发行所

开 本：787×1092 1/16 印张：26.25 字数：639 千字

版 次：2005 年 1 月第 1 版 2005 年 1 月第 1 次印刷

书 号：ISBN 7-302-10153-1/TP·1066

印 数：1-4000

定 价：39.00 元 (1CD)

本书如存在文字不清、漏印以及缺页、倒页、脱页等印装质量问题，请与清华大学出版社出版部联系调换。联系电话：(010) 82896445

目 录

第1章 .NET和C#概述	1	3.2.2 接口	32
1.1 Microsoft .NET技术的由来	1	3.2.3 代表	35
1.2 公共语言架构 (CLI)	2	3.2.4 数组	37
1.3 .NET Framework	2	3.3 类型转换	41
1.3.1 公共语言运行时 (CLR)	3	3.3.1 数值转换	41
1.3.2 .NET类库	4	3.3.2 枚举转换	44
1.4 C#语言简介	5	3.3.3 引用转换	45
1.5 C# 2.0的新增特性	6	3.3.4 装箱和拆箱转换	46
1.6 小结	7	3.3.5 转换检查	49
第2章 C#应用程序初探	8	3.4 小结	49
2.1 第一个C#应用程序	8	3.5 习题	49
2.2 C#程序的基本结构	10	第4章 成员、变量和常量	51
2.2.1 程序集	10	4.1 基本成员类型	51
2.2.2 命名空间	11	4.1.1 字段	52
2.2.3 类型、类和方法	12	4.1.2 方法	53
2.2.4 注释	14	4.1.3 嵌套成员	58
2.3 与用户进行交互	14	4.1.4 成员访问限制	58
2.3.1 Console类	14	4.1.5 静态成员和非静态成员	63
2.3.2 一个交互程序	15	4.2 变量	65
2.4 编写Windows应用程序	17	4.3 常量	67
2.5 小结	19	4.4 小结	70
2.6 习题	19	4.5 习题	70
第3章 数据类型	20	第5章 表达式	72
3.1 值类型	20	5.1 操作符	72
3.1.1 整数类型	20	5.2 算术表达式	73
3.1.2 字符类型	23	5.2.1 基本算术运算	73
3.1.3 实数类型	24	5.2.2 枚举运算	76
3.1.4 布尔类型	26	5.2.3 模运算	76
3.1.5 结构类型	26	5.2.4 其他	78
3.1.6 枚举类型	28	5.3 自增和自减表达式	80
3.2 引用类型	29	5.4 位运算表达式	81
3.2.1 类	29	5.4.1 取补运算	81

5.4.2 与、或和异或运算	82	7.2.3 静态构造函数	122
5.4.3 移位运算	83	7.3 属性	123
5.5 赋值表达式	84	7.4 索引函数	126
5.5.1 简单赋值	85	7.5 事件	130
5.5.2 复合赋值	85	7.6 操作符重载	134
5.6 关系表达式	85	7.7 this关键字	138
5.6.1 比较运算	85	7.8 小结	142
5.6.2 类型判断	87	7.9 习题	142
5.7 条件逻辑表达式	89	第8章 字符串类型	144
5.8 其他特殊表达式	90	8.1 构造String类型	144
5.8.1 一元加减表达式	90	8.2 String类的字段、属性和索引函数 ...	144
5.8.2 条件表达式	91	8.3 字符操作和子串操作	145
5.8.3 类型表达式	92	8.3.1 字符操作	145
5.8.4 创建表达式	96	8.3.2 子串操作	146
5.8.5 溢出检查表达式	97	8.4 字符串的比较和连接	148
5.9 小结	99	8.4.1 比较字符串	148
5.10 习题	99	8.4.2 连接字符串	149
第6章 控制结构	101	8.5 字符串的格式化	150
6.1 选择语句	101	8.5.1 字符替换	150
6.1.1 if语句	101	8.5.2 字符填充	151
6.1.2 switch语句	104	8.5.3 字符修剪	152
6.2 循环语句	106	8.5.4 参数格式化	153
6.2.1 while循环语句	106	8.5.5 解析字符串	154
6.2.2 do-while循环语句	108	8.6 StringBuilder类	156
6.2.3 for循环语句	109	8.6.1 引入StringBuilder类	156
6.2.4 foreach循环语句	111	8.6.2 构造StringBuilder对象	158
6.3 跳转语句	113	8.6.3 StringBuilder类的字段、属性 和索引函数	158
6.3.1 break语句	113	8.6.4 StringBuilder类的方法	158
6.3.2 continue语句	114	8.7 小结	160
6.3.3 return语句	114	8.8 习题	161
6.3.4 goto语句	115	第9章 继承和多态	162
6.4 小结	117	9.1 继承	162
6.5 习题	117	9.1.1 基类和派生类	162
第7章 类	118	9.1.2 隐藏基类的成员	166
7.1 面向对象的程序设计基础	118	9.1.3 base关键字	168
7.2 类的生命周期	119	9.1.4 继承中的构造函数和析构函数	169
7.2.1 构造函数	119	9.2 多态性	173
7.2.2 析构函数	121		

9.2.1 虚拟方法和重载方法	173	12.1 引入可空类型	245
9.2.2 抽象类和抽象方法	178	12.2 泛型结构NullableType	248
9.2.3 密封类和密封方法	180	12.2.1 概述	248
9.3 再谈接口与继承	182	12.2.2 判断和取值	249
9.4 小结	191	12.2.3 类型转换	252
9.5 习题	191	12.3 操作符提升	254
第10章 泛型类	194	12.4 可空布尔类型	258
10.1 概述	194	12.5 空值结合操作符	260
10.1.1 引入泛型的原因	194	12.6 小结	261
10.1.2 泛型类的语法定义	196	12.7 习题	261
10.2 泛型类的成员	198	第13章 泛型方法	264
10.2.1 在成员中使用类型参数	198	13.1 概述	264
10.2.2 类型参数的成员和默认值	201	13.1.1 引入泛型方法	264
10.2.3 嵌套泛型类	202	13.1.2 泛型方法的定义	265
10.2.4 静态成员	203	13.1.3 调用泛型方法	267
10.2.5 泛型类中的操作符重载	205	13.1.4 惟一性规则	268
10.3 多参数泛型类	206	13.2 泛型方法的重载	270
10.3.1 多个类型参数的定义和使用	206	13.2.1 概述	270
10.3.2 泛型类中方法的标识	209	13.2.2 示例程序：读写器	271
10.4 类型限制	210	13.3 泛型方法与代表	279
10.5 泛型类之间的继承	212	13.4 小结	281
10.5.1 开放类型与封闭类型	212	13.5 习题	281
10.5.2 普通基类与派生泛型类	213	第14章 遍历器	284
10.5.3 泛型基类与普通派生类	215	14.1 概述	284
10.5.4 泛型基类与泛型派生类	216	14.1.1 foreach语句与遍历	284
10.6 小结	220	14.1.2 Iterator模式	284
10.7 习题	220	14.2 使用可枚举类型	286
第11章 泛型结构和接口	223	14.2.1 IEnumerable和IEnumerable<T> 接口	286
11.1 泛型结构	223	14.2.2 实现多种遍历方式	287
11.2 泛型接口	226	14.2.3 带参遍历	292
11.2.1 泛型接口的定义	226	14.3 使用枚举器	296
11.2.2 惟一性规则	227	14.4 遍历器工作机制	298
11.2.3 泛型接口与继承	231	14.4.1 遍历器代码	298
11.2.4 集合中的泛型	234	14.4.2 遍历流程	300
11.3 小结	241	14.5 示例程序：联系人分类输出	302
11.4 习题	242	14.6 小结	305
第12章 可空类型	245	14.7 习题	305

第15章 匿名方法.....	306	17.2.1 与IO操作相关的枚举	350
15.1 方法的命名调用和匿名调用	306	17.2.2 驱动器	352
15.2 深入了解Delegate类	308	17.2.3 目录	355
15.2.1 创建代表对象	308	17.2.4 文件	358
15.2.2 属性	308	17.3 文件流和数据流	361
15.2.3 方法调用	310	17.3.1 抽象类Stream	362
15.3 匿名方法的定义规则	312	17.3.2 文件流FileStream	363
15.4 外部变量	314	17.3.3 流的文本读写器	365
15.5 代表对象作为方法参数和返回值	316	17.3.4 流的二进制读写器	368
15.6 在事件中使用匿名方法	321	17.3.5 常用的其他流对象	369
15.7 小结	325	17.4 程序示例	372
15.8 习题	326	17.4.1 文件加密器	372
第16章 异常处理	328	17.4.2 联系人数据的IO操作	375
16.1 错误和异常	328	17.5 小结	382
16.2 C#中的异常处理结构	331	17.6 习题	382
16.2.1 try-catch语句	331	第18章 代码组织与管理	384
16.2.2 try-catch-finally语句	333	18.1 分布类型 (Partial Type)	384
16.2.3 try-finally语句	336	18.1.1 分布类型的定义	384
16.2.4 throw语句	337	18.1.2 分布泛型	386
16.3 异常的层次结构	339	18.1.3 分布类型的应用	388
16.3.1 异常传播	339	18.2 代码中的预处理器指令	395
16.3.2 Exception类	340	18.2.1 条件编译	395
16.3.3 其他一些常见的异常类	342	18.2.2 编译警告和错误	399
16.4 使用异常的原则和技巧	344	18.2.3 其他的一些预处理器指令	400
16.5 小结	345	18.3 XML代码注释	400
16.6 习题	346	18.3.1 XML简介	401
第17章 文件和流	349	18.3.2 使用XML注释	401
17.1 文件系统概述	349	18.3.3 XML注释标记	403
17.2 驱动器、目录和文件	350	18.4 小结	407
		18.5 习题	408

第 1 章 .NET 和 C#概述

本章将介绍.NET技术的由来和关键组成部分,并提供.NET的核心编程语言C#及其升级版本的基本信息。

1.1 Microsoft .NET技术的由来

Internet技术在20世纪60年代末期才开始起步,却成为人类历史上意义最为深刻的变革之一。随着网络应用需求的飞速增长,以资源共享和协同工作为主要特征的网络分布计算已经成为新一代计算和应用的主流。随着主机计算向网络分布计算的过渡,软件系统的规模和复杂度呈几何级数增加,传统的软件开发方法面临前所未有的挑战。

为突破时间、空间及不同平台的限制,提高软件系统的重用和集成能力,组件化软件开发技术应运而生。它将对象技术和组件技术有机地结合在一起,强调组件的可重用性和互操作性,力图通过组件集成来构建高效的分布式软件系统。20世纪90年代以来出现了3种典型的组件技术标准,它们分别是OMG (Object Management Group, 对象管理组织) 的CORBA、Microsoft的COM/DCOM以及Sun公司的JavaBeans。这些技术各有优劣,都面临着不断改进和发展的要求,谁也没有一统天下的实力。

2000年6月22日, Microsoft正式宣布了其新一代计算计划,即Microsoft .NET (读作“ dot-net ”), 这是Microsoft针对第三代Internet推出的全新设计思想,其目标之一就是希望.NET能够取代COM,进而成为Windows应用和Web应用的主流开发模型。这可以说是Microsoft继使用Windows取代DOS操作系统之后又一项战略性的举措。从技术角度理解, .NET是一个全新的计算平台,它的主要特点有:

- 面向异构网络、硬件平台和操作系统,为软件提供最大限度的开放性、可扩展性、互操作性及可重用性。例如,它能够将个人电脑上运行的软件方便地移植到手机、掌上电脑等各种终端之上。
- 实现软件系统之间的智能交互和协同工作,提高整个网络的效率和利用率,特别是实现企业级的系统集成和资源优化,给开放型企业的生产力水平带来质的飞跃。其中的一个关键组件就是以XML和SOAP协议为基础的Web服务。
- 提供一个标准化的、安全的、一致的模型和环境,简化分布式应用程序的开发难度,从而大幅提高软件系统的质量和生产率。例如, .NET支持以不同编程语言开发的组件之间进行无缝的交互和集成。

.NET在理念中包含了对操作系统和计算机网络设计思想的延伸,即将Internet本身作为构建新一代操作系统的基础。Microsoft的宏伟目标是让.NET彻底改变软件的开发方式、发

行方式和使用方式，构建第三代Internet平台，解决各种协同合作问题，实现信息的高效沟通和分享，让整个Internet为人们提供最全面的服务。

Microsoft .NET技术可以分为两大部分：规范和实现。

1.2 公共语言架构（CLI）

Microsoft为.NET技术制定了一套完整的规范，并将它提交到ECMA等标准化组织，以促进.NET的广泛应用。这套规范就是公共语言架构（CLI ,Common Language Infrastructure）。它的组成部分包括：

- 通用类型系统（CTS ,Common Type System）。定义了一套类型系统的框架，规定了数据类型的声明、使用和管理方法。.NET中的任何数据类型都必须遵守该框架中的约定。
- 公共语言规范（CLS ,Common Language Specification）。一组语言规则的集合。如果某种编程语言符合其中的所有规则，它就是标准的.NET编程语言，可以实现和其他.NET编程语言的跨语言集成；如果某个组件中的代码使用了其中规定的功能，它就是标准的.NET组件，可以实现和其他.NET组件的交互。
- 通用中间语言（CIL ,Common Intermediate Language）。一种中性语言，更准确地说是一套处理器无关的指令集合。可以针对不同的平台将通用中间语言翻译为可执行的机器指令（二进制代码），而任何.NET编程语言所编写的代码可以被编译成通用中间语言指令集。
- 其他相关的标准化文件、格式、规定等。

1.3 .NET Framework

公共语言架构是.NET的技术规范，Microsoft依据该规范在Windows平台上开发了一个完整的实现，这就是.NET Framework，它包括.NET类库和公共语言运行时（CLR ,Common Language Runtime）两大部分，其整体环境结构如图1.1所示。

其中，.NET类库定义了大量的可重用对象和组件，开发人员可以利用这些现成的对象和组件快速开发各种服务和应用。不论是.NET类库中预定义的对象和组件，还是开发人员自行创建的对象和组件，它们都运行在公共语言运行时这个平台上，而底层平台的操作细节由公共语言运行时负责实现。

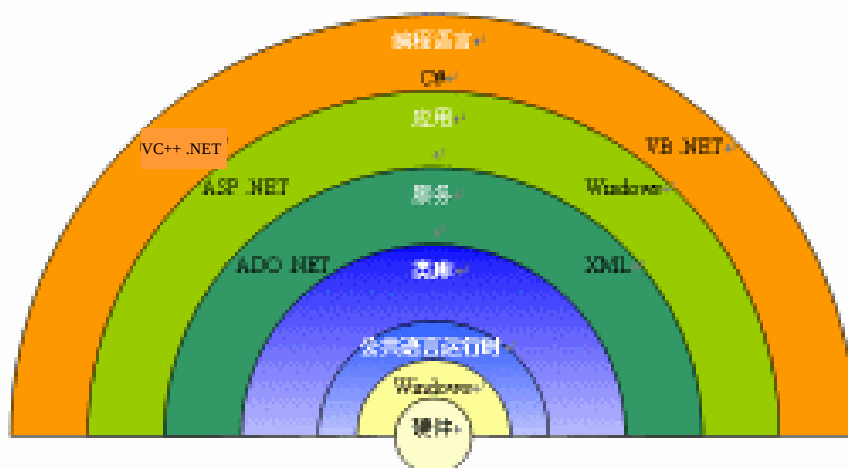


图 1.1 .NET Framework 环境结构

1.3.1 公共语言运行时（CLR）

准确地说，公共语言运行时就是一个虚拟机，它为各种.NET应用提供了一个高性能的、抽象于底层操作系统和硬件的运行时环境。公共语言运行的主要功能体现在3个方面：

（1）管理代码的执行

各类.NET应用程序的代码被编译为通用中间语言。在程序执行时，公共语言运行时将通用中间语言翻译为具体的机器指令，负责加载所需的元数据类型、组件及其他各种资源，并在执行过程中提供安全性控制、错误处理、垃圾回收等服务；在必要时，公共语言运行时还可以进行程序的版本处理、代码调试等工作。

由于公共语言运行时对程序执行细节进行了封装，开发人员就可以专注于程序的业务和功能流程。此外，公共语言运行时提供了一致的实现模型，各类组件和程序能够以统一的方式进行交互。因此，公共语言运行时是在.NET Framework上实现程序高度的互操作性、集成性以及可重用性的关键所在。

为了充分利用公共语言运行时所提供的强大功能，Microsoft推荐使用托管代码（Managed Code）来开发各种.NET应用。托管代码是指满足公共语言规范要求、所使用元数据类型均属于通用类型系统、并使用针对公共语言运行时的语言编译器所开发的代码。使用托管代码开发的对象，其整个生命周期都在公共语言运行时的管理范围当中，因而能够享受跨语言集成、跨语言异常处理、增强的安全性、版本控制和部署支持、简化的组件交互模型、调试和分析等多种服务。

然而，.NET也允许开发人员使用非托管代码来绕过公共语言运行时的某些服务，或自行实现公共语言运行时的某些功能。托管代码和非托管代码之间可以相互调用。

（2）提供通用类型系统

公共语言运行时定义了一套完整的、符合通用类型系统约定的元数据类型集合，包括

值类型和引用类型两部分。这些元数据类型为组件的资源控制、版本管理，以及组件之间的交互提供了关键的信息。

公共语言运行时利用通用类型系统对代码进行严格的验证，以保证组件和程序的可靠性。开发人员所定义的各种新类型都从公共语言运行时的元数据类型中派生，以保证这些类型在生成和使用的过程中完全遵守通用类型系统的约定。以公共语言运行时为基础，现有的及新增的.NET语言都能够以一致的方式进行编程。这样，使用不同编程语言的开发人员就可以共同编写代码，构建同一个应用程序。例如，在C#语言编写的程序中，可以继承使用VB .NET语言编写的类，调用它的对象和方法。

（3）提供系统服务

公共语言运行时屏蔽了底层操作系统和硬件的特性，应用程序是在这个虚拟机上而不是在操作系统上直接运行。在需要进行资源管理、内存分配、系统API访问等工作时，由公共语言运行时与操作系统打交道，.NET组件和应用程序则使用公共语言运行时提供的统一接口，这不仅简化了开发难度，而且还使得程序能够在不同的平台上方便地移植。

1.3.2 .NET类库

.NET类库是Microsoft开发的一个面向对象的可重用类型集合，旨在帮助开发人员进行各种Windows应用和Web应用开发。这些预定义类型实现了丰富的功能，并提供了良好的可重用性。类库中的主要内容有：

- 基础类型定义，包括各种数学类型及其运算、字符串类型、对象类型等；
- 数据结构封装，包括集合、链表、队列、堆栈等；
- Windows界面要素，包括标签、按钮、菜单、工具栏、文本框、列表框、数据绑定和导航控件等；
- Web界面要素，能够实现Windows界面所具有的大部分功能，还包括Web客户端所特有的要素，如验证控件、客户端缓存等；
- Web Service，包括服务描述、消息与响应、协议等；
- XML文档处理，包括XML文件、属性、元素、节点、读写器、解析器等；
- 文件输入输出，包括驱动器、目录、文件、流、读写器等；
- 数据访问，包括数据连接、适配器、数据集、表格、记录、主键、关联等；
- 类型反射，包括程序集、类和对象、方法、属性、字段等的元数据类型信息获取；
- 异常处理，包括系统、应用程序、数学运算、安全性限制等引发的异常。

.NET类库中的类型主要包括类、接口和值类型，它通过命名空间对这些类型进行层次化的组织和管理。

1.4 C#语言简介

20世纪80年代以来,C和C++一直是使用最为广泛的商业应用开发语言。这两种语言在带来强大控制能力和高度灵活性的同时,代价是相对较长的学习周期和较低的开发效率,同时对控制能力和灵活性的滥用也给程序的安全性带来了潜在的威胁。C++语言过度的功能扩张也破坏了面向对象的设计理念。软件行业迫切地需要一种全新的现代程序设计语言,它能够在控制能力与生产效率之间求得良好的平衡,特别是将快速应用开发与对底层平台所有功能的访问紧密结合在一起,并与Web标准保持同步,C#(读作“C-Sharp”)语言就是这一历史使命的承担者。

C#语言从C和C++发展而来,它汲取了包括C、C++、Java在内的多种语言的精华,是一种简单、完备、类型安全和完全面向对象的高级程序设计语言。它的设计目标就是在继承C和C++强大功能的同时,兼有RAD(Rapid Application Development,快速应用程序开发)语言的简易和高效。作为.NET的核心编程语言,C#充分享受了公共语言运行时所提供的优势,能够与其他应用程序方便地集成和交互。下面对它的几个突出特点进行了描述:

- 简洁的语法。C#取消了指针,也不定义烦乱的伪关键字,它使用有限的指令、修饰符和操作符,语法上几乎不存在任何冗余,整个程序结构十分清晰。初学者可以轻松快速地掌握C#的基本特性,而C和C++程序员转入C#则几乎不会有任何障碍。
- 精心的面向对象设计。C#具有面向对象的语言所应有的基本特性:封装、继承和多态性。它禁止多继承,禁止各种全局方法、全局变量和常量。C#以类为基础来构建所有的类型,并通过命名空间对代码进行层次化的组织和管理,减少了发生命名冲突的可能性。
- 与Web的紧密结合。借助Web服务框架,C#使得网络开发和本地开发几乎一样简单。开发人员无需了解网络的细节,可以用统一的方式来处理本地的和远程的C#对象,而C#组件能够方便地转变为Web服务,并被其他平台上的各种编程语言调用。
- 完整的安全性及错误处理。C#符合通用类型系统的类型安全性要求,并用公共语言运行时所提供的代码访问安全特性,从而能够在程序中方便地配置安全等级和用户权限。此外,垃圾收集机制自动管理对象的生命周期,开发人员无须再负担内存管理的任务。应用程序的可靠性进一步得到了提高。
- 版本管理技术。C#在语言中内置了版本控制功能,并通过接口和继承来实现应用的可扩展性。应用程序的维护和升级更加易于管理。
- 灵活性与兼容性。C#中允许使用非托管代码,能够与各种现有的组件和程序(包括COM组件、WIN32 API等)进行集成和交互。它还可以通过代表(delegate)来模拟指针的功能,通过接口来模拟多继承的实现。

为了吸引软件开发人员和合作伙伴对.NET的认同,Microsoft推出了新一代的集成开发环境——Microsoft Visual Studio .NET。该环境提供了对C#语言编程的可视化支持,使得开发人员能够方便地创建、运行、调试和发布C#程序,从而针对.NET平台快速地构建广泛的应用。

1.5 C# 2.0的新增特性

作为一个新生事物，.NET技术很快取得了巨大的成功。Internet上仿佛一夜之间出现了不计其数的Web服务组件，大量的网站（特别是电子商务网站）升级到了.NET平台，包括Borland在内的许多开发工具厂商纷纷推出了自己的.NET编程语言。据统计，在美国有一半以上的开发人员在使用Visual Studio .NET进行开发，而C#则成为其中最受欢迎的开发语言。然而，.NET前进的脚步并没有因此而停止。Microsoft在2003年初将.NET Framework从1.0版本升级到了1.1版本，主要内容是对.NET类库的修改和完善，并通过.NET Framework精简版强化了对各种智能设备编程的支持。Microsoft还在新一代操作系统Windows 2003中内置了.NET Framework 1.1。

作为Microsoft .NET战略计划的关键一步，.NET Framework 2.0版本的推出则标志着.NET技术一次全新的重大升级。.NET Framework 2.0中新增的主要功能有：

- 对64位处理器和操作系统的全面支持；
- 对IPv6网络协议的支持；
- 可轻松实现权限管理的访问控制列表编程方法；
- 全新的数据访问策略ADO .NET；
- 新增的数据保护编程接口；
- 全新的Web应用开发平台ASP .NET 2.0；
- 高安全性的加密数据流；
- 增强的COM互操作性；
- 增强的证书管理功能；
- 更为强大的调试功能；
- 网络连接中的变化检测；
- 包括FTP客户端请求、HTTP资源缓存等在内的分布式计算新功能的实现；
- 在通用类型系统对泛型的支持；
- 事务管理和线程管理。

与此同时，Microsoft也将C#语言升级到了C# 2.0，用以全面支持在.NET Framework 2.0上的编程。在新版本的Visual Studio 2005中也包含了对C# 2.0可视化编程的支持。C# 2.0中的新增特性主要有：

- 泛型（Generic Type）。这是从C++的模板（Templates）发展而来的一个特性，它允许在类和方法的定义中使用参数来指代抽象的数据类型，需要时可以用不同具体的数据类型来取代该参数，从而在更高的层次上实现代码的可重用性。C#将类的继承机制和泛型有机地结合在一起，把面向对象的方法推向了一个新的高度。泛型的概念适用于类（请参看第10章）、结构和接口（请参看第11章），也适用于单独的方法（请参看第13章）。
- 可空类型（Nullable Type）。这是泛型的一个具体应用，它允许在变量中包含未确

定的值，例如一个布尔变量的值可以是真或假，也可以未确定。这就为值类型的处理带来了更大的方便（请参看第12章）。

- 遍历器（Iterator）。允许使用foreach循环结构来遍历集合中的每个元素。遍历器所实现的功能并不复杂，但它是面向对象设计模式的一个典型应用。.NET类库采用了完美的面向对象设计，其中用到了多种设计模式，遍历器算是其中一个杰出的代表（请参看第14章）。
- 匿名方法（Anonymous Method）。可以把一段代码直接作为参数使用，而无需显式地定义一个方法，这样不仅减少了编程的工作量，提高了代码的可维护性，更极大地方便了程序中的各种运算方式（请参看第15章）。
- 分布类型（Partial Type）。允许将一个类型的定义分布到多个代码段乃至多个源文件中，从而提高代码的可读性和可维护性（请参看第18章）。

1.6 小 结

Microsoft .NET是面向Internet的计算平台，它将软件开发和应用的观念、方式和模型都推向了一个全新的境界。C#则是Microsoft专门针对.NET而设计的高级编程语言，它在保持强大功能的同时，能够显著地提高现代软件的开发效率。C# 2.0是对C#语言的一次全面升级，使用它编写的程序具有更为优越的可重用性、可维护性和灵活性。本书随后的各章节将对C#语言的主要功能特性逐一进行深入的讲解。

第2章 C#应用程序初探

本章将从一个最基本的例子开始，带领读者逐步走进C#的编程世界。本章中使用的例子虽然简单，但却不失完整地展示了应用程序的基本结构及C#程序的鲜明特点。

2.1 第一个C#应用程序

首先介绍一个最简单的C#应用程序：向用户输出一行文本。

在代码编辑器中输入下面的内容：

```
//程序清单P2_1.cs :
using System;
namespace P2_1
{
    public class FirstProgram
    {
        public static void Main(string[] args)
        {
            Console.WriteLine("Welcome to .NET");
        }
    }
}
```

以扩展名.cs保存文件，这里取P2_1.cs。如果使用微软提供的Visual C#命令行编译器，可以使用下面的编译命令：

```
csc P2_1.cs
```

这样就会生成一个名为P2_1.exe的可执行程序（默认与源文件前名相同）。输入命令行P2_1并回车执行该程序，将在控制台窗口输出一行预定的文本“Welcome to .NET”，如图2.1所示。

注意下面这行代码：

```
public static void Main(string[] args)
```

每个C#可执行程序都必须有一个执行入口，这个入口由一个名为Main的方法定义。正是该方法中包含的代码实现了输出文本的功能：

```
Console.WriteLine("Welcome to .NET");
```

Microsoft C#编译器csc的主要编译选项见表2.1。

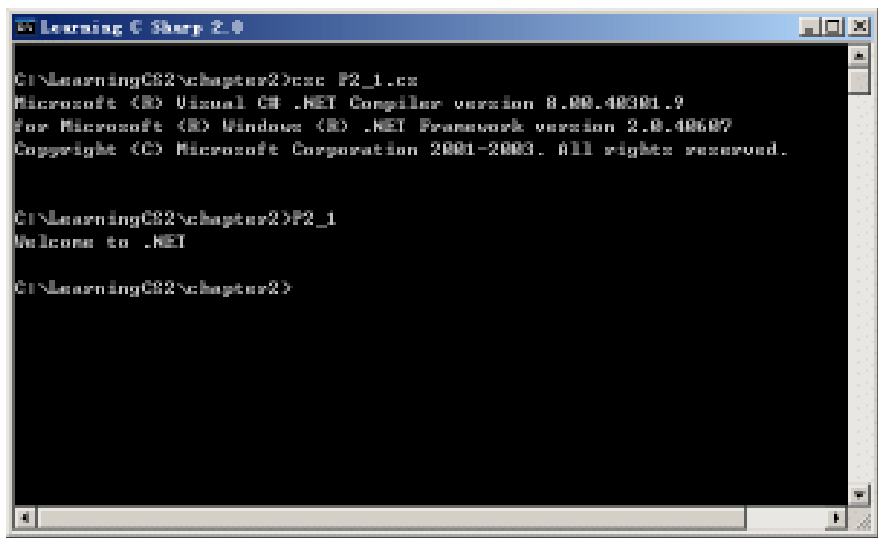


图 2.1 程序 P2_1.cs 的运行结果

表2.1 csc编译器主要选项

选项	用途
输出选项	
/out: <file>	指定生成文件的路径名
/target: exe	生成控制台应用程序格式
/target: winexe	生成Windows应用程序格式
/target: dll	生成动态链接库格式
/target: module	生成模块格式
/doc: <file>	指定同时生成的XML文档的路径名
/platform: <string>	指定运行代码的目标平台
输入选项	
/reference: <file list>	指定一组要引用的程序集文件
/addmodule: <file list>	指定一组要添加的模块文件
/resource: <resinfo>	编译时嵌入指定资源
/linkresource: <resinfo>	编译时链接指定资源
/win32res	指定Win32资源文件
/win32icon	指定Win32图标文件
信息选项	
/debug [+/-]	指定是否生成调试信息
/optimiz: [+/-]	指定是否进行编译优化

(续表)

选项	用途
/incremental [+/-]	指定是否对源文件进行递增式编译
/warn: <n>	设置警告级别, 可为1~4级
/warnaserror [+/-]	指定是否把警告信息视为错误
/warnaserror [+/-]: <warn	指定是否把列出的警告信息视为错误
/nowarn: <warn list>	编译过程中不生成列出的警告信息
/checked [+/-]	指定是否将算术运算溢出视为异常
/unsafe [+/-]	指定是否允许使用不安全的代码
其他选项	
/nologo	编译时不包含版权信息
/noconfig	编译时不包含配置文件信息
/help	列出编译命令选项
/codepage: <n>	指定所使用的代码页
/main: <type>	指定某个类的Main方法作为程序入口

例如, 下面的命令将在编译时进行代码优化, 生成名为FirstProgram.exe的文件, 并在程序中加入调试信息:

```
csc /debug+ /optimize+ /out:FirstProgram.exe P2_1.cs
```

2.2 C#程序的基本结构

接下来以程序P2_1.cs为例, 对C#应用程序的基本结构进行简要分析。

2.2.1 程序集

人们最早进行程序设计时需要从零开始编写代码, 包括自己定义所需的所有类型和方法。随着程序设计技术的发展特别是高级语言的出现, 开发人员开始在自己的程序中调用许多其他程序模块中的元素, 从而提高开发效率。这些程序模块可能是自己或他人所设计的, 也可能是系统中预定义的。

.NET则更进一步, 以程序集 (Assembly) 为单位来对应用程序进行组织。一个程序集可以看作是一个或多个物理程序模块的组合, 它是.NET应用程序打包、运行和发布的最小单位。 .NET 类库中的对象和组件都包含在各个程序集中, 而最基本的一个程序集名为System, 其中包括了大量的基础类型定义, 如对象类object, 以及字符、字符串、整数、浮点数等基本数据类型。程序P2_1.cs中使用的Console类也属于该程序集。

在使用csc编译程序时, 已经隐式地引用了System程序集。要显式地引用某个程序集,