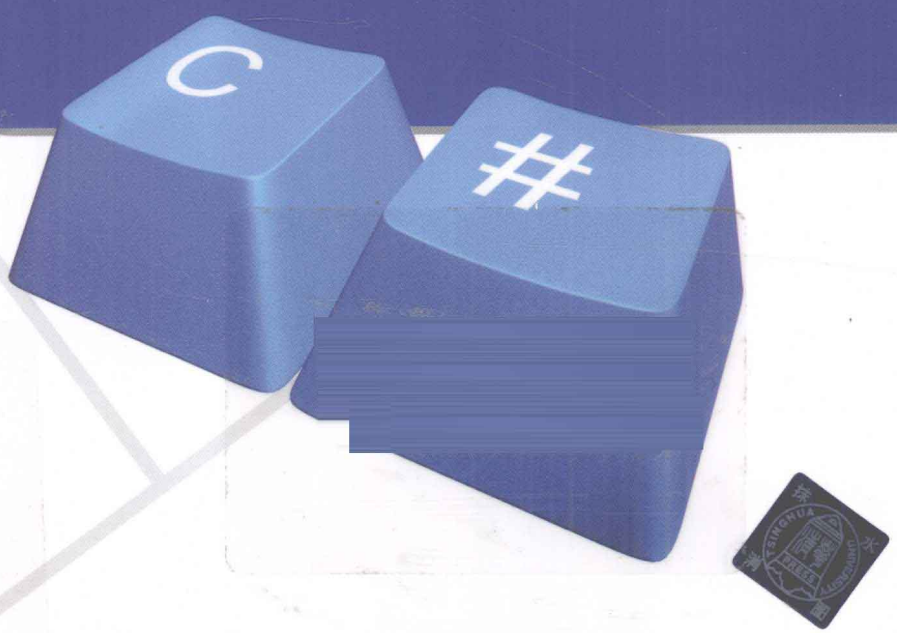
21世纪面向工程应用型
计算机人才培养规划教材

朱二喜 陆红蕾 徐敏 编著

# C# 程序设计与项目实践




清华大学出版社

21世纪面向工程应用型计算机人才培养规划教材

# C#程序设计与项目实践

朱二喜　陆红蕾　徐　敏　编著

清华大学出版社
北京

## 内 容 简 介

本书基于 Visual Studio 2008/. NET Framework 3.5 开发和运行环境，阐述了 C# 3.0 语言的基础知识，以及使用 C# 3.0 语言的实际开发应用实例。具体内容包括 C# 语言基础、对象和类、基本常用类与异常处理、Windows 编程基础、文件操作、ADO. NET 与数据访问、GDI+绘图等。

本书作者结合多年的程序设计、开发经验以及多年的授课经历，参阅大量文献资料，精选了大量的实际项目和示例，由项目到技能，再到知识点，个个击破，由浅入深、循序渐进地介绍了 C# 程序设计的基本知识，让读者更扎实更牢固更系统全面地掌握 C# 语言的理论与应用。

本书适用于高职高专类的学生，也可以作为自学者的初读版本，同时还可作为广大程序设计开发者、爱好者的参考书籍。



**图书在版编目(CIP)数据**

C# 程序设计与项目实践/朱二喜等编著. --北京：清华大学出版社，2012.3
(21 世纪面向工程应用型计算机人才培养规划教材)
ISBN 978-7-302-27567-1

Ⅰ. ①C…　Ⅱ. ①朱…　Ⅲ. ①C 语言－程序设计－高等学校－教材　Ⅳ. TP312

中国版本图书馆 CIP 数据核字(2011)第 270933 号

**责任编辑**：梁　颖　薛　阳
**封面设计**：杨　兮
**责任校对**：焦丽丽
**责任印制**：李红英

**出版发行**：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课 件 下 载：http://www.tup.com.cn,010-62795954
**印 刷 者**：清华大学印刷厂
**装 订 者**：三河市溧源装订厂
**经　　销**：全国新华书店
**开　　本**：185mm×260mm　　**印　张**：25.75　　**字　数**：626 千字
**版　　次**：2012 年 3 月第 1 版　　**印　次**：2012 年 3 月第 1 次印刷
**印　　数**：1～3000
**定　　价**：39.50 元

产品编号：044897-01

# 前言

foreword

C#作为一门非常流行的程序设计语言，集中了C、C++和Java等语言的优点。C#程序开发人员可以直接使用.NET Framework中完整且丰富的类库设计出跨平台的软件系统。C#具有简单易学、使用方便的特点，利用它可以开发复杂的软件系统。

本书主要基于Visual Studio 2008/.NET Framework 3.5开发和运行环境，阐述了C# 3.0语言的基础知识，以及使用C# 3.0语言的实际开发应用实例。

本书共分为8章，第1章讲述基本的理论知识，包括C#语言的概述和特点，.NET Framework的概述和结构，以及Visual Studio 2008开发运行环境的介绍和C#编译过程；第2章讲述C#语言的基础知识，包括常量与变量、通用数据类型、运算符、数组、数据类型转换和C#编程规则；第3章讲述类和对象，包括类的声明、内部结构、属性、构造函数、委托与事件、索引器等，同时还介绍了面向对象的编程，包括继承和接口；第4章讲述常用的类和异常处理，包括String类、DateTime结构、Random类等主要的类，同时介绍了异常的概念、处理机制和常见的异常类；第5章讲述Windows基础编程，包括基本控件的属性与事件驱动，同时讲授了菜单栏等窗体部分的制作过程；第6章讲述文件的操作，包括文件的创建、读取与写入等；第7章讲述数据库访问，包括ADO.NET的概述与内容、数据库访问方法、操作数据的类和控件的介绍；第8章讲述GDI+绘图，包括主要图形的绘制及各种绘图要素的使用方法。

本书具有如下特点。

(1) 采用项目驱动式教学，在实际的项目中，讲述技能操作与知识点。

(2) 由浅入深，理论与实际结合，通过大量的项目和示例阐述程序设计的基本原理。

(3) 读者不仅可以在项目中学到理论知识，同时可以为以后的工作积累丰富的开发经验。

(4) 本书中有些示例或项目的源代码可为以后的软件开发提供参考。

由于时间关系及编者的学识经验有限，书中的不足之处在所难免，敬请同行、专家和读者指正。

编　者

2011年9月

# 目录

contents

第1章 chapter 1

# 基本理论知识

C#是微软公司开发的使用简单、功能强大、面向对象、表达能力丰富的语言，也是可用于创建要运行在.NET CLR上的应用程序的语言之一，它是微软公司专门为使用.NET平台而创建的。它不但结合了C++强大的灵活性和Java语言的简洁等特性，而且还吸纳了Visual Basic语言所具有的易用性。C#在.NET Framework框架中扮演着重要角色，它是互联网软件和服务战略的重要内容。我们不能孤立地使用C#语言，而必须和.NET Framework一起考虑。C#编译器专门用于.NET，这表示用C#编写的所有代码总是在.NET Framework中运行。C#的结构和方法反映了.NET基础方法论；C#的特定语言功能取决于.NET的功能，或依赖于.NET基类。

**本章技能学习要点。**

技能1：认识C#语言

技能2：了解.NET框架结构

技能3：认识通用类型系统

技能4：认识程序集

技能5：认识.NET Framework类

技能6：认识命名空间

技能7：Visual Studio 2008的介绍

技能8：掌握创建、编写和调试项目

## 项目一：一个简单的C#程序

本书介绍了C#语言的语法和用法，但不过分强调.NET Framework。这是必需的，因为首先要具备C#编程基础才能灵活使用.NET Framework。下面介绍一个简单的C#程序。

学习某种编程语言，通常采用“Hello，World”程序作为起步。首先打开Visual Studio 2008，在菜单栏中单击“文件”→“新建”→“项目”命令，弹出“新建项目”对话框。此对话框中列出了Visual C#能够创建的应用程序类型。选择“控制台应用程序”作为项目类型，并将应用程序的名称改为“HelloWorld”，单击“确定”按钮，如图1-1所示。

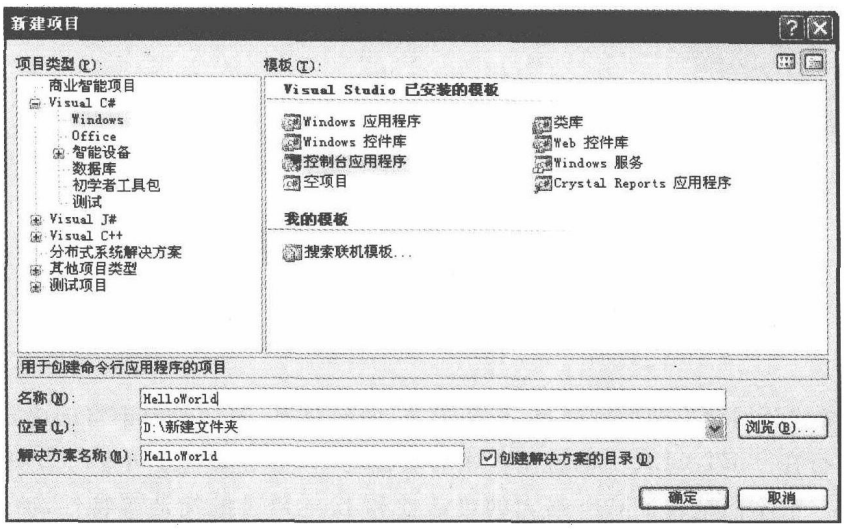



图 1-1

在代码编辑窗口中输入以下 C# 代码。

```
using System;
using System.Collections.Generic;
using System.Text;

namespace HelloWorld
{
    class Program
    {
        static void Main(string[] args)
        {
        Console.WriteLine("Hello,World!");
        }
    }
}
```

编写好程序代码后，C# 源程序将被存储在以 cs 作为扩展名的文件中。假定“Hello, World”源程序文件被存为 HelloWorld.cs，那么，使用下面的命令行就能通过 Microsoft C# 编译器编译这个程序：

```
csc HelloWorld.cs
```

或者单击 Visual Basic 2008 开发平台下的工具栏中的“启动”按钮 或者按 F5 键来编译。它将产生一个名为 HelloWorld.exe 的可执行程序集。当程序运行时，输出结果如图 1-2 所示。

“Hello, World”程序开头是 using 指令，引用了 System 命名空间(namespace)。命名空间提供了 C# 程序和类库分层次的组织手段。命名空间包括类型和其他命名空间，例如，System 命名空间包括若干类型(如程序中引用的 Console 类)，以及若干其他命名空间(如 IO 和 Collection)。如果通过 using 指令引用给定命名空间，就可以对命名空间的成员进行非限定的使用。正是由于程序中使用了 using 指令，才能够将 System.Console.WriteLine 简写为 Console.WriteLine。

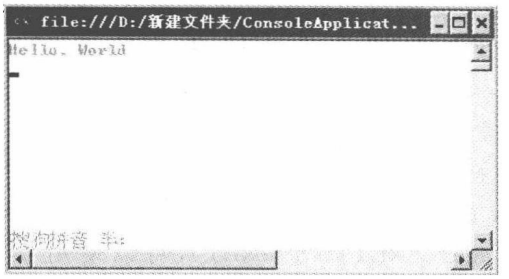



图 1-2

“Hello, World”程序中声明的 Hello 类中只有一个成员，即名为 Main 的方法。Main 方法是用 static 修饰符声明的。静态方法不同于实例方法，后者需要使用关键字 this 来引用特定的对象实例，而静态方法的操作不需要引用特定的对象。作为约定，被命名为 Main 的静态方法充当程序的入口点。

程序输出是由 System 命名空间下 Console 类的 WriteLine 方法产生的。这个类是由 .NET框架类库提供的，默认情况下，类库被 Microsoft C＃编译器自动引用。注意，C＃本身没有单独的运行时类库。事实上，.NET 框架是 C＃的运行时类库。

## 技能 1：认识 C＃语言

C＃语言是在 C、C++和 Java 语言基础上重新构建的，它的语法与 C++和 Java 语言非常相似，是一种基于.NET Framework、完全面向对象的、类型安全的编程语言。C＃为应用程序开发人员提供了快速开发手段，并且保持了 C++语言的特点和优点。从语法形式和易用性上讲，C＃语言几乎综合了目前流行的所有高级语言的优点，提供了一种语法简洁、功能完善而又容易使用的外在表现形式。使用 C＃开发应用程序比使用其他语言相对简单，因为其语法比较简单。C＃是一种强大的语言，在 C＃中能完成意想不到的功能任务。例如直接访问和处理系统内存，但只能在标记为“不安全”的代码中使用。

C＃是一种类型安全的语言，这表示一旦为某些数据指定了类型，就不能转换为另一个不相关的类型。所以，在类型之间转换时，必须遵守严格的规则。执行相同的任务时，用 C＃编写的代码通常比其他语言要长。但 C＃代码更健壮，调试也比较简单，.NET 总是可以随时跟踪数据的类型。

C＃只是.NET 开发的一种语言，但在程序员看来，这是最好的一种语言。C＃的优点是，它是唯一为.NET Framework 设计的语言，是在移植到其他操作系统上的.NET 版本中使用的主要语言。要使语言（如 VB.NET）尽可能类似于其以前的语言，且仍遵循 CLR（通用语言运行库），就不能完全支持.NET 代码库的某些功能。但 C＃能使用.NET Framework 代码库提供的每种功能。.NET 的最新版本还对 C＃语言进行了几处改进，这是为了满足开发人员的要求，使之更强大。

### 1. 用 C＃能编写什么样的应用程序

.NET Framework 没有限制应用程序的类型。C＃使用.NET Framework，所以也没有限制应用程序的类型。这里仅讨论几种常见的应用程序类型。

- Windows 应用程序。这些应用程序如 Microsoft Office,有人们很熟悉的 Windows 外观和操作方式,使用. NET Framework 的 Windows Forms 模块就可以生成这种应用程序。Windows Forms 模块是一个控件库,其中的控件(例如按钮、工具栏、菜单等)可以用于建立 Windows 用户界面(UI)。
- Web 应用程序。这些是 Web 页,可以通过任何 Web 浏览器查看。. NET Framework 包括一个动态生成 Web 内容的强大系统,允许个性化、实现安全性等。这个系统叫做 Active Server Pages . NET(ASP. NET),可以使用 C # 通过 Web Forms 创建 ASP. NET应用程序。
- Web 服务。这是创建各种分布式应用程序的新方式,使用 Web 服务可以通过 Internet 虚拟交换数据。无论使用什么语言创建 Web 服务,也无论 Web 服务驻留在什么系统上,都使用一样简单的语法。

这些类型也需要某种形式的数据库访问,这可以通过. NET Framework 的 Active Data Object . NET(ADO. NET)部分来实现。也可以使用许多其他资源,例如创建联网组件、输出图形、执行复杂教学任务的工具。

**2. 认识 C# 程序结构**

C # 中程序结构的关键概念为程序、命名空间、类型、成员和程序集。C # 程序包括一个或者多个源文件。程序中声明类型,类型包含成员并能够被组织到命名空间中。类和接口是类型的例子。字段、方法、属性和事件则是成员的例子。当 C # 程序被编译时,它们被物理地打包到程序集中。程序集的文件扩展名一般为. exe 或者. dll,这取决于它们是实现为应用程序(application),还是类库(library)。

**示例:**

```
using System;

namespace Acme.Collection
{
    public class Stack
    {
        Entry top;
        public void Push(object data)
        {
            top = new Entry(top.data);
        }

        public object Pop()
        {
            if (top == null) throw new InvalidOperationException();
            object result = top.data;
            top = top.next;
            return result;
        }

        class Entry
        {
            public Entry next;
```

```
            public object data;

            public Entry(Entry next, object data)
            {
                this.next = next;
                this.data = data;
            }

        }
    }
}
```

在叫做 Acme. Collection 的命名空间下，声明名为 Stack 的类，这个类的完全限定名就是 Acme. Collection. Stack。它包括几个成员：一个名为 top 的字段，两个分别命名为 Push 和 Pop 的方法，以及一个名为 Entry 的嵌套类。Entry 类又进一步包括三个成员：一个名为 next 的字段，一个名为 data 的字段，以及一个构造函数。假定这个示例的源程序被存为 acme. cs 文件，命令行为：

```
csc /t:library acme.cs
```

将这个示例编译为类库(不带 Main 入口点的代码)，并且产生一个名为 acme. dll 的程序集。

程序集包括中间语言(Intermediate Language，IL)指令形式的可执行代码，符合元数据(metadata)形式的信息。在它执行之前，程序集的 IL 代码将被. NET 公共语言运行库(Common Language Runtime，CLR)自动转换成特定处理器的代码。

由于程序集是自描述的功能单元，它既包括代码，也包括元数据，因此，在 C# 中不需要 #include 指令和头文件。假如某个 C# 程序需要引用特定程序集中的公共类型和成员，那么只在编译时简单地引用那个程序集就可以了。例如，下面的程序使用来自 acme. dll 程序集中的 Acme. Collection. Stack 类。

```
using System;
using Acme.Collection

class Test
{
    static void Main()
    {
        Stack s = new Stack();
        s.Push(1);
        s.Push(10);
        s.Push(100);
        Console.WriteLine(s.Pop());
        Console.WriteLine(s.Pop());
        Console.WriteLine(s.Pop());
    }
}
```

如果程序被存为 test. cs 文件，那么，在 test. cs 被编译时，acme. dll 可以通过/r 选项被

引用。

```
csc /r: acme.dll test.cs
```

这样可以创建一个名为 test.exe 的可执行程序集。

C#允许一个程序的原文本被存为几个源文件。当多文件的C#程序被编译时，所有的源文件被一起处理，并且各个源文件从概念上能够自由地相互引用，就如同在处理之前，所有的源文件被连接成一个大文件。在C#中向前声明是没有必要的，原因就是声明的顺序无关紧要。C#不限制一个源文件只能声明一个公共类型，也不要求源文件必须与该文件中的类型相匹配。

## 技能2：了解.NET框架结构

.NET是微软公司推出的面向网络的一套完整的开发平台。如图1-3所示，.NET框架结构的核心是.NET Framework，.NET框架在操作系统之上为程序员提供了一个编写各种应用程序的高效的工具和环境。.NET框架结构的顶层是用各种语言所编写的应用程序，这些应用程序由公共语言运行库控制执行。.NET框架试图解决Microsoft Windows环境中许多与应用程序开发和部署有历史联系的问题。例如，COM已经通过允许已编译组件之间经二进制协议进行会话来试图减轻这种不便。然而，COM也有缺陷。COM没有提供运行时发现组件所提供的服务的明白易懂的方法。.NET框架则通过所谓的“映像”概念提供了解决这一问题的机制。错误处理是该框架解决的另一个问题。根据所做出的API调用，此API调用可能产生一个错误或返回一个错误码。如果返回错误码，程序员必须具有可能返回的常见错误方面的知识。该框架通过为所有错误产生一个异常简化了错误处理。Framework库提供了对传统上属于C++程序员领域的低级特性的访问。Windows服务、COM+对象共享以及对如HTTP、SMTP和FTP之类的Internet协议的访问现在已在C#开发者的牢牢掌握之中。

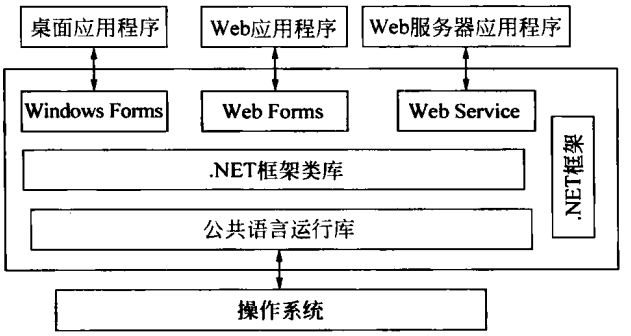



图 1-3

.NET框架为运行于其环境内的应用程序提供了许多瞄准执行领域的服务。所有为.NET编写的程序都运行在被称为公共语言运行库(Common Language Runtime，CLR)的环境内。为运行于CLR内编写的应用程序被看做是托管代码。托管代码可利用CLR提供的服务。某些服务如垃圾收集是自动提供的，其他服务如软件的版本编号则要求程序员干预。

### 1. C#与.NET的关系

C#是一种相当新的编程语言，C#的重要性体现在以下两个方面。

- 它是专门为与Microsoft的.NET Framework一起使用而设计的(.NET Framework是一个功能非常丰富的平台，可开发、部署和执行分布式应用程序)。
- 它是一种基于现代面向对象设计方法的语言，在设计它时，Microsoft还吸取了其他类似语言的经验，这些语言是近20年来面向对象规则得到广泛应用后才开发出来的。

有一个很重要的问题要弄明白：C#就其本身而言只是一种语言，尽管它是用于生成面向.NET环境的代码，但它本身不是.NET的一部分。.NET支持的一些特性，C#并不支持。而C#语言支持的另一些特性，.NET却不支持(例如运算符重载)。

但是，因为C#语言是和.NET一起使用的，所以如果要使用C#高效地开发应用程序，理解Framework就非常重要，所以本节将介绍.NET的内涵。

### 2. C#语言的编译运行

C#程序在.NET Framework上运行，被编译为中间代码(IL)。IL代码与资源(例如位图和字符串)一起作为一种称为程序集的执行文件存储在磁盘上，通常具有扩展名.exe或.dll(库)。

执行C#程序时，程序集将加载到CLR中，然后根据程序集清单中的信息执行不同的操作。如果符合安全要求，CLR执行即时编译(Just-In-Time(JIT)Compilation)将IL代码转换为本机机器指令并执行。CLR还提供与自动垃圾回收、异常处理和资源管理有关的其他服务。C#程序编译过程如图1-4所示。

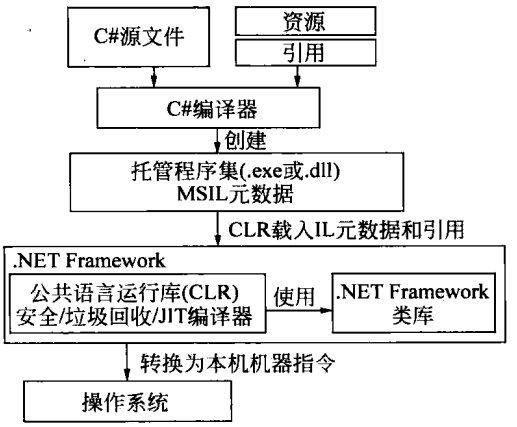



图 1-4

### 3. 公共语言运行库

.NET Framework的核心是其运行库的执行环境，称为公共语言运行库(CLR)或.NET运行库。通常将在CLR的控制下运行的代码称为托管代码(Managed Code)。但是，在CLR执行开发的源代码之前，需要编译它们(在C#中或其他语言中)。在.NET中，编译分为以下两个阶段。

- 把源代码编译为Microsoft中间语言(IL)。

• CLR 把 IL 编译为平台专用的代码。

在这两阶段的编译过程非常重要，因为 Microsoft 中间语言(托管代码)是. NET 的许多优点的关键。

Microsoft 中间语言与 Java 字节代码共享一种理念：它们都是一种低级语言，语法很简单(使用数字代码，而不是文本代码)，可以非常快速地转换为内部机器代码。对于代码来说，这种精心设计的通用语言，有很重要的优点。

(1) 平台无关性。

这意味着包含字节代码指令的同一文件可以放在任一平台中，运行时编译过程的最后阶段可以很容易完成，这样代码就可以运行在该特定的平台上，换言之，编译为中间语言就可以获得. NET 平台无关性，这与编译为 Java 字节代码会得到 Java 平台无关性是一样的。

(2) 即时编译。

即时编译器(Just In Time Compiler)负责将 MSIL 指令转换成本地机器码。它只在第一次调用对象的方法时执行该任务。一旦激活，JIT 就在内存中保留转换后的 MSIL。随后对方法的调用会直接进入本地机器码。

(3) 语言的互操作性。

使用 IL 不仅支持平台无关性，还支持语言的互操作性。简言之，就是能将任何一种语言编译为中间代码，编译好的代码可以与从其他语言编译过来的代码进行交互操作。

**4. 详细介绍中间语言和即时编译器**

通过前面的学习，我们了解了 Microsoft 中间语言显然在. NET Framework 中有非常重要的作用。C# 开发人员应明白，C# 代码在执行前要编译为中间语言(实际上，C# 编译器仅编译为托管代码)，这是有意义的，现在介绍一下 IL 的主要特征，因为面向. NET 的所有语言在逻辑上都需要支持 IL 的主要特征。

中间语言具有以下主要特征。

• 面向对象和使用接口。
• 值类型和引用类型之间的巨大差别。
• 强数据类型。
• 使用异常来处理错误。
• 使用特性(attribute)。

下面详细讨论这些特征。

(1) 面向对象和接口的支持。

. NET 的语言无关性还有一些实际的限制。中间语言在设计时就打算实现某些特殊的编程方法，这表示面向它的语言必须与编程方法兼容，Microsoft 为 IL 选择的特定道路是传统的面向对象的编程，带有类的单一继承性。除了传统的面向对象编程外，中间语言还引入了接口的概念，它们显示了在带有 COM 的 Windows 下的第一个实现方式。. NET 接口与 COM 接口不同，它们不需要支持任何 COM 基础结构，例如，它们不是派生自 IUnknown，也没有对应的 GUID。但它们与 COM 接口共享下述理念：提供一个契约，实现给定接口的类必须提供该接口指定的方法和属性的实现方式。