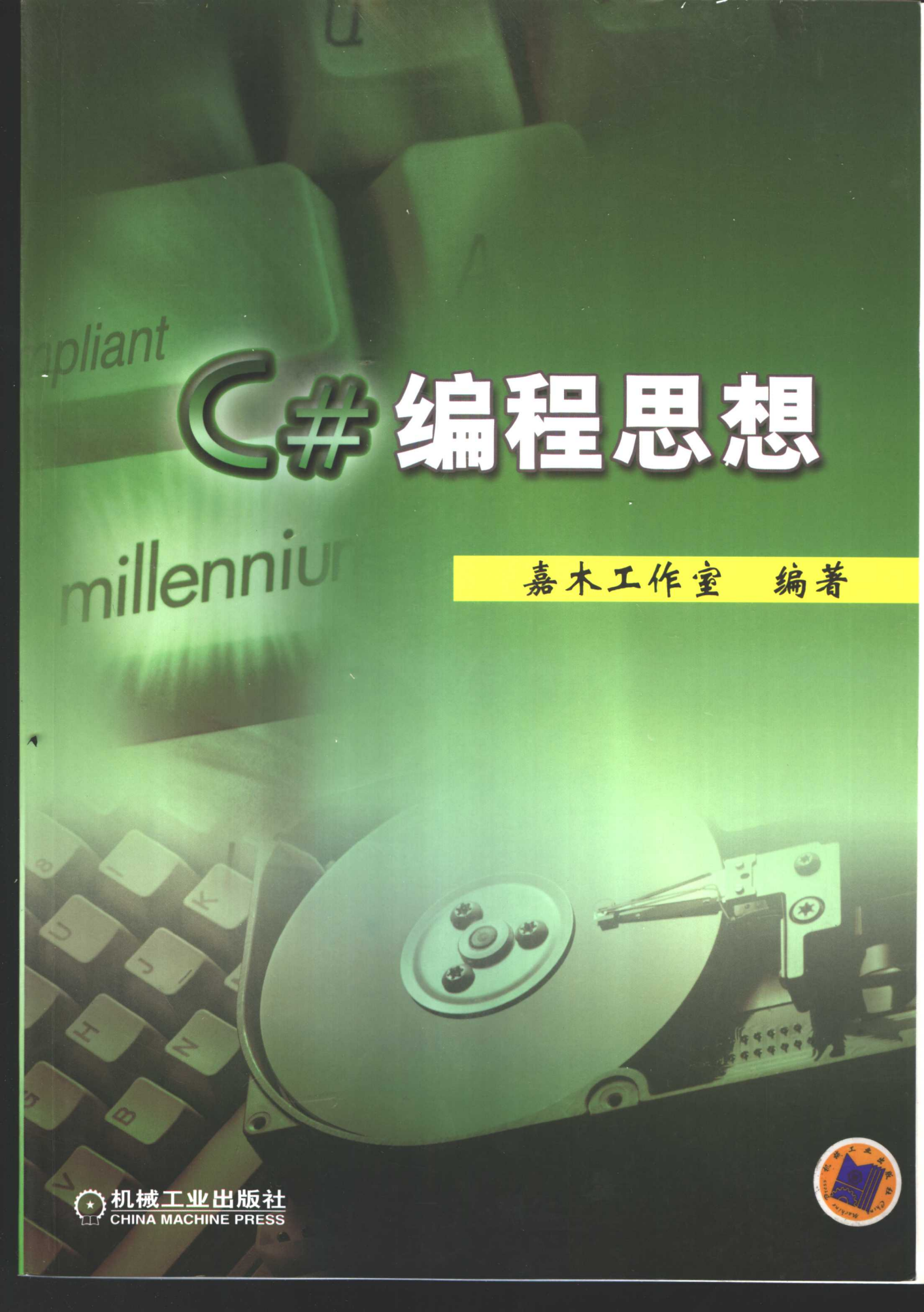
C#编程思想
嘉木工作室 编著
机械工业出版社
CHINA MACHINE PRESS

C#编程思想

嘉木工作室　编著

机 械 工 业 出 版 社

该C#开发语言是包含在Visual Studio.NET框架中的最新应用程序开发语言。目前，随着网络应用程序的广泛开发与应用需求，C#已经成为开发基于计算和通信的最流行的语言。本书从基础入门，结合Web开发的特点，详细介绍C#开发语言的语法，并针对这些语法提供了丰富的例程，以充分发挥C#语言的开发优势。

本书适合从事网络程序设计的程序员、大中专院校相关专业师生和培训学校学生。

图书在版编目（CIP）数据

C# 编程思想/嘉木工作室编著. —北京：机械工业出版社，2003.3

ISBN 7-111-11860-X

Ⅰ.C… Ⅱ.嘉… Ⅲ.C语言—程序设计

Ⅳ.TP312

中国版本图书馆CIP数据核字（2003）第025283号

机械工业出版社（北京市百万庄大街22号 邮政编码100037）

策　　划：边　萌　　责任编辑：边　萌

封面设计：张　静　　责任印制：闫　焱

北京交通印务实业公司印刷·新华书店北京发行所发行

2003年5月第1版第1次印刷

787mm×1092mm×1/16·22印张·554千字

0 001—4000册

定价：36.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

本社购书热线电话（010）68993821、88379646

封面无防伪标均为盗版

前　言

在过去的二十多年内，C 和 C++已经成为在软件开发中应用最广泛的开发语言。但是 C 和 C++的灵活性牺牲了它们的开发效率，如果和其他开发语言相比，相同功能的 C/C++软件通常需要更长的开发周期。正是由于 C/C++开发的复杂性和需要较长的开发周期，许多 C/C++开发人员都在寻找一种可以在功能和开发效率间提供更多平衡的开发语言。作为快速创建和集成 XML Web 服务和应用程序的单一综合工具，Visual Studio .NET 在改善操作的同时极大地提高了开发效率。其中的 Visual C# .NET （C#的发音为 C Sharp）是 Microsoft 推出的新型语言，它可提高 C 和 C++ 开发人员的效率。Visual C# .NET 对具有属性、方法、索引器、特性、版本和事件的组件提供一流的支持，并同时对 .NET 平台提供强有力的和有效的支持。

C#是一种先进的、面向对象的语言，通过 C#可以让开发人员快速建立大范围的基于 MS 网络平台的应用，并且提供大量的开发工具和服务，帮助开发人员开发基于计算和通信的各种应用。由于 C#是一种面向对象的开发语言，所以将 C#可以大范围的适用于高层商业应用和底层系统的开发。即使是通过简单的 C#构造也可以将各种组件方便地转变为基于 Web 的应用，并且能够通过 Internet 被各种系统或其他开发语言所开发的应用调用。

即使抛开上面所提到的优点，C#也可以为 C/C++开发人员提供快速的开发手段而不需要牺牲任何 C/C++语言的特点/优点。从继承角度来看，C#在更高层次上重新实现了 C/C++，熟悉 C/C++开发的人员可以很快的转变为 C#开发人员。C#的优点包括：

- 高的开发效率与安全性；
- 与 Web 开发相结合；
- 减小开发中的错误；
- 提供内置的版本支持来减少开发费用；
- 功能强，易于表现，灵活；
- 更好的结合商业应用中的流程与软件实现；
- 可扩展的协作能力；
- 允许有限制的使用指针。

总之，使用 C#能够利用方便快捷的 Microsoft 网络平台建立各种应用和建立能够在网络间相互调用的 Web 服务。从开发语言的角度来讲，C#可以更好地帮助开发人员避免错误，提高工作效率，并具有 C/C++的强大功能。

该书结构合理、内容翔实、循序渐进，适合各个层次的读者学习参考。由于时间仓促，本书编著过程中难免存在错误和纰漏，恳请广大读者批评指正。

编　者

目　录

第1章 C# 简介

C#是在 C 和 C++基础上发展起来的一门简单、现代、对象定位、类型安全的程序语言。C#（读作“C sharp”）植根于 C 和 C++语言家族，且与 C 和 C++程序非常接近，C#旨在把 System 的多产性与 C++尚未完善的功能结合起来。

C#是 Microsoft Visual Studio.NET 的一部分。除 C#外，Visual Studio 还支持 Visual Basic、Visual C++，以及撰稿语言 VBScript 和 Jscript。所有这些语言都可访问 Microsoft.NET 平台，后者包括一个普通的执行工具和一个丰富的类库。Microsoft.NET 定义一个普通语言的子集（CLS）及一种混合语言，这种混合语言确保 CLS 所属的语言和类库之间能够通用地操作，且不留下任何痕迹。虽然 C#是一门新的语言，但对于 C#开发者来说，它完全可以访问正时兴的工具如 Visual Basic 和 Visual C++所用的丰富的类库。C#本身没有类库。

本章以下几部分将对这门语言的基本特征加以介绍。以后的几章则对其规则及异常现象进行具体的说明，有时也涉及其运算方式。本章的虽然不十分详细，但却简短清晰。其目的是为了向读者简要介绍这门语言，以便读者能够编写简单的程序及对以后几章的阅读。

1.1 开始（Start）

典型的“hello,world”程序的编写方法如下：

```
using System;
class Hello
{
    static void Main() {
          Console.WriteLine("hello, world");
    }
}
```

C#程序的源代码一般储存在一或多个带有扩展名.CS 的文本文件如 hello.cs 中。使用 Visual Studio 提供的命令行编译函数，这个程序编译时可带有命令行指令：

```
CSC     hello.cs
```

该指令产生一个 hello.exe 的可执行程序。这个程序输出：

```
Hello,world
```

此程序严格的检测过程如下：

- 指令 Using System 引用一个由 Microsoft.NET Frame Work 类库提供的名字空间 System。这个名字空间含有方法 Main 中的类 Console。名字空间提供类库中不同元素的等级含义。指令“using”使得作为名字空间成员的类型的非权名使用成为可

能。在程序“hello,world”中使用的 Console.WriteLine 就是 System.Console.WriteLine 的编写形式。

- 方法 Main 是类 hello 的成员，它具有 static 修改函数。因此，它是类 hello 而不是该类实例的一个方法。
- 程序的主要入口——即被调用开始执行程序的方法——总是静态方法 Main。
- “ hello,world ”通过使用一个类库进行输出。这种语言本身没有类库，而是使用一个也被其他语言如 Visual Basic 和 Visual C＋＋使用的普通的类库。C 和 C++的开发者对于那些在程序“ hello,world ”中出现的语句很感兴趣。
- 该程序不使用 Main 的总方法。总体水平不支持方法和变量；这样的元素总是包含在类型声明过程中（如类和结构的声明过程）。
- 该程序既不使用操作符“::”,也不使用“－>”。“::”实际上不是一个操作符，而操作符“－>”只在一小部分程序中使用。分隔符“.”被用在复合名字如：Console.WriteLine 中。
- 该程序没有前声明。也没有必要使用前声明，因为声明顺序无关紧要。
- 该程序不用＃include 引进程序文本。程序的从属关系是根据符号而不是根据文本确定的。此系统消除了用不同语言所编写的程序之间的障碍。例如，类 console 也可用其他语言编写。

1.2 类型（Types）

C#支持两种类型：Value types 和 reference types。值类型包括简单类型（如 char，int 和 float）、枚举类型和结构类型。引用类型包括类类型、接口类型、委托类型和数组类型。

值类型与引用类型的不同点在于，值类型的变量直接包括它们的数据，而引用类型的变量则把引用储存到对象中。引用类型的两个变量可以引用同一个对象。这样，对一个变量的操作就可能影响另一个变量所引用的对象。值类型的每一个变量都具有它们自己的数据拷贝，因此对一个变量的操作不可能影响另一个变量。下面的例子就表明了这种差异。

```
using System;
class Class1
{
public int Value = 0;
}
class Test
{
   static void Main() {
          int val1 = 0;
          int val2 = val1;
          val2 = 123;
          Class1 ref1 = new Class1();
          Class1 ref2 = ref1;
```

```
            ref2.Value = 123;
            Console.WriteLine("Values: {0}, {1}", val1, val2);
            Console.WriteLine("Refs: {0}, {1}", ref1.Value, ref2.Value);
    }
}
```

该程序输出：

```
Values: 0, 123
Refs: 123, 123
```

对局部变量 val1 的赋值不影响局部变量 val2，因为两个局部变量属于同一个值类型（即类型 int），而值类型的每一个局部变量都单独储存。在常量中，赋值表达式 ref2.value=123；既影响 ref1 所引用的对象，也影响 ref2 所引用的对象。以下几行则另当别论：

```
Console.WriteLine("Values: {0}, {1}", val1, val2);
Console.WriteLine("Refs: {0}, {1}", ref1.Value, ref2.Value);
```

因为它们表示 Console.WriteLine 的一些字符串的格式化过程，这个过程所需要的自变量数其实是可变的。第一个自变量是一个字符串，它可能含有数个座位如{0}和{1}。每一个座位表示一系列自变量，如{0}表示第二个自变量，{1}表示第三个自变量等等。在输出结果被传送到控制台之前，每一个自变量值都被其相应自变量格式化的值取代。

开发者可以通过枚举或结构声明定义新的值类型，也可以通过类、接口以及委托声明定义新的引用类型。

下面的例子：

```
using System;
public enum Color
{
   Red, Blue, Green
}
public struct Point
{
   public int x, y;
}
public interface IBase
{
   void F();
}
public interface IDerived: IBase
{
```