

“Herbert告诉你在现实世界中使用C# 4.0所需要知道的一切！”

——Michael Howard, Microsoft

C# 4.0 完全参考手册

- 全面介绍C#语言的方方面面
- 介绍C# 4.0的新增功能，包括任务并行库、PLINQ、dynamic类型等
- 提供了数百个易上手的示例

Herbert的经典参考手册——针对C# 4.0
作了全面修订和更新。

Herbert Schildt是最畅销编程书籍的作者，其创作的编程书籍在全世界已经实现了数百万的销量。

(美) Herbert Schildt 著
李增民 李立新 译



清华大学出版社

C# 4.0 完全参考手册

清华大学出版社

北 京

Herbert Schildt

C# 4.0: The Complete Reference

EISBN: 978-0-07-174116-3

Copyright © 2009 by The McGraw-Hill Companies, Inc.

All Rights reserved. No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including without limitation photocopying, recording, taping, or any database, information or retrieval system, without the prior written permission of the publisher.

This authorized Chinese translation is jointly published by McGraw-Hill Education (Asia) and Tsinghua University Press. This edition is authorized for sale in the People's Republic of China only, excluding Hong Kong, Macao SAR and Taiwan.

Copyright © 2010 by McGraw-Hill Education (Asia), a division of the Singapore Branch of The McGraw-Hill Companies, Inc. and Tsinghua University Press.

版权所有。未经出版人事先书面许可,对本出版物的任何部分不得以任何方式或途径复制或传播,包括但不限于复印、录制、录音,或通过任何数据库、信息或可检索的系统。

本授权中文简体字翻译版由麦格劳-希尔(亚洲)教育出版公司和清华大学出版社合作出版。此版本经授权仅限在中华人民共和国境内(不包括香港特别行政区、澳门特别行政区和台湾)销售。

版权©2010 由麦格劳-希尔(亚洲)教育出版公司与清华大学出版社所有。

北京市版权局著作权合同登记号 图字: 01-2010-5499

本书封面贴有 McGraw-Hill 公司防伪标签,无标签者不得销售。

版权所有,侵权必究。侵权举报电话: 010-62782989 13701121933

图书在版编目(CIP)数据

C# 4.0 完全参考手册/(美)希尔特(Schildt, H.) 著;李增民,李立新 译. —北京:清华大学出版社,2010.12
书名原文: C# 4.0: The Complete Reference

ISBN 978-7-302-24043-3

I. C… II. ①希… ②李… ③李… III. C 语言—程序设计 IV. TP312

中国版本图书馆 CIP 数据核字(2010)第 207136 号

责任编辑:王 军 张立浩

装帧设计:孔祥丰

责任校对:胡雁翎

责任印制:王秀菊

出版发行:清华大学出版社

地 址:北京清华大学学研大厦 A 座

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

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

印 刷 者:清华大学印刷厂

装 订 者:北京市密云县京文制本装订厂

经 销:全国新华书店

开 本:185×260 印 张:57.75 字 数:1552 千字

版 次:2010 年 12 月第 1 版 印 次:2010 年 12 月第 1 次印刷

印 数:1~4000

定 价:118.00 元

产品编号:036891-01

作者简介

Herbert Schildt 是 C#、C、C++ 和 Java 方面的权威，他的编程书籍在世界范围内的销量已逾数百万册，并且已经翻译成所有主流的外语。Herb 是大量畅销书籍的作者，其著作包括 *Java: The Complete Reference*、*C++: The Complete Reference*、*C: The Complete Reference* 等。虽然对计算的所有方面都有一定兴趣，但是 Herb 主要关注计算机语言，包括编译器、解释器和机器人控制语言，并且对语言标准化也深感兴趣。Herb 获得了伊利诺斯大学的学士学位和硕士学位。可以通过访问网站 www.HerbSchildt.com 与 Herb 取得联系。

技术编辑简介

Michael Howard 是 Microsoft 可信计算(Trustworthy Computing, TwC)小组的安全工程团队的首席安全项目经理，他负责管理跨公司的安全设计、编程和测试技术。Howard 是安全开发生命周期(Security Development Lifecycle, SDL)的体系结构设计师，SDL 用于改进 Microsoft 软件的安全性。

Howard 于 1992 年在 Microsoft 的新西兰分部开始他的职业生涯，头两年在产品支持服务部(PPS)负责 Windows 和编译器方面的工作，之后进入 Microsoft 顾问咨询服务部(MCS)，给客户提供安全基础设施支持，并帮助设计自定义的解决方案和软件开发。1997 年，Howard 前往美国，在 Windows 部门负责 Internet Information Services(Microsoft 下一代的 Web 服务器)方面的工作，然后于 2000 年开始从事现在的工作。

Howard 是 *IEEE Security & Privacy* 杂志的编辑，他经常在与安全相关的会议上发言，并且定期发表一些有关安全编码和设计的文章。他是 6 本安全书籍的合著者，包括获奖的《软件安全的 24 宗罪——编程缺陷与修复之道》(清华大学出版社引进并出版)、*The Security Development Lifecycle*，其最新的著作是 *Writing Secure Code for Windows Vista*。

前 言

编程人员往往喜欢精益求精，他们总是不断地想方设法提高程序的性能、效率和可移植性。因此，他们对所使用的工具同样也要求甚多，特别是对编程语言的选择。编程语言的种类很多，但只有一小部分是优秀的语言。优秀的编程语言往往功能强大而又灵活，语法简洁且清晰；能够为创建正确的代码提供极大的帮助，而不是设置障碍；能够不断地支持最新的功能，而不是逐渐被淘汰出局。最重要的是，优秀的编程语言注定会有这样一种无形的品质：让程序员在使用时感觉良好。C#就是这样一种编程语言。

C#是 Microsoft 为支持 .NET Framework 而创建的语言，它建立在丰富的编程资源之上，其首席设计师是长期以来公认的软件大师 Anders Hejlsberg。C#继承了至今世界上最成功的两种计算机语言：C 和 C++，它继承了 C 语言的语法、部分关键字和运算符，并以 C++定义的对象模型为基础加以改进。C#还与另一种非常成功的语言 Java 有紧密关系。

C#和 Java 有共同的起源，像双胞胎一样，但是在许多重要方面两者也有所不同，因此 C#和 Java 更像是堂兄弟。例如，两者都支持分布式编程，并且都使用中间代码获得安全性和可移植性，但是两者的实现细节是不同的。它们都提供了大量运行时错误检查、安全性和托管执行，但是同样在细节方面有所区别。然而，与 Java 不同的是，C#提供了对指针的访问——指针是 C++支持的一种功能。因此，C#将 C++的原始功能与 Java 的类型安全性组合在一起。而且，它在安全性和功能之间达到了最佳平衡，并且实现了透明化。

在计算技术变革的历史中，为了适应计算环境的变化、计算机语言的发展以及人们在思维方式和编程方式上的改变，编程语言得到了不断的发展，C#也不例外。在不断的改进、改编和创新过程中，C#已经展示了它能不断满足现代编程人员需求的能力。随着 2000 年 Microsoft 发布最初的 C# 1.0 版本以来不断向该语言添加许多新的功能就是最好的证明。

C# 2.0 是 C#语言的第一个重大修订版本，该版本通过增加一系列的新功能来使编程人员更容易地编写更有弹性、更可靠和更高效的代码。毫无疑问，C# 2.0 中增加的最重要的功能是泛型。通过使用泛型，在 C#中创建类型安全的、可重用的代码成为可能。因此，泛型的添加从根本上扩展了 C#语言的功能和作用范围。

C#语言的第二个重大修订版本就是 C# 3.0。毫不夸张地说，C# 3.0 添加的功能已经重新定

IV C# 4.0 完全参考手册

义了 C# 的核心部分，同时在此过程中提高了计算机语言开发的门槛。在 C# 3.0 的许多创新功能中，最具代表性的两个功能是 LINQ 和 Lambda 表达式。LINQ 代表语言集成查询，它允许通过使用 C# 语言的元素创建数据库样式的查询。Lambda 表达式则实现函数样式的语法，它使用 Lambda 运算符 $\Rightarrow$ ，并且频繁用于 LINQ 表达式中。

C# 语言的第三个重大修订版本就是 C# 4.0，也是本书将要介绍的版本。C# 4.0 构建在前面几个版本的基础之上，提供了大量简化普通编程任务的新功能。例如，它添加了命名实参和可选实参，该功能使得某些类型的方法调用变得更便利。它添加了 `dynamic` 关键字，这使得 C# 可用于数据类型是在运行时获得的情形(如使用 COM 接口或反射时)。C# 原先就支持的协变和逆变功能现在扩展至用于类型形参。通过对 .NET Framework(C# 的类库)进行增强，任务并行库(TPL)和并行 LINQ(PLINQ)提供了对并行编程的支持。使用这些子系统可以很容易创建可自动扩展以更好地利用多核计算机的代码。因此，随着 C# 4.0 的发布，C# 将可以利用高性能的计算平台。

由于 C# 能够快速适应编程领域中不断变化的需求，因此它始终是充满活力的、不断创新的语言。总之，C# 已成为现代计算中最为强大的、功能最为丰富的一种语言，任何程序员都不能忽略它的存在。编写本书的目的就是帮助读者掌握 C# 语言。

内容简介

本书介绍了 C# 4.0，全书分为两个部分。第 I 部分全面讨论了 C# 语言，包括 C# 4.0 版本中增加的新功能。这是全书内容比重最大的一部分，它描述了关键字、语法和一些定义 C# 语言的功能，并且介绍了 I/O 操作、文件处理、反射和预处理器。

第 II 部分研究了 C# 类库，同时也是 .NET Framework 类库。这是一个庞大的类库，由于篇幅有限，本书不可能深入探讨整个 .NET Framework 类库，而只能集中地讨论 System 名称空间中包含的核心类库。另外，该部分还包括对集合、多线程、任务并行库、PLINQ 和联网的介绍。这些是类库中几乎每个 C# 编程人员都会用到的部分。

读者对象

本书的读者无须具有任何编程经验。如果您已掌握 C++ 或者 Java 语言，那么阅读本书时将毫不费力，因为 C# 和这两种语言有很多共同之处。如果之前没有任何编程经验，您也能够通过对本书的学习逐渐掌握 C# 语言，但需要仔细研究每章中的示例。

编程环境

必须使用 Visual Studio 2010 或更高版本编译和运行 C# 4.0 程序。

在线源代码下载

本书中所有程序的源代码都可以从 www.mhprofessional.com 和 www.tupwk.com.cn/downpage 网站上免费获得。

反馈信箱

《C# 4.0 完全参考手册》是 Herb Schildt 系列编程书籍之一，Herbert Schildt 网站(www.HerbSchildt.com)上展示了更多他所编著的书籍，并提供了他的联系方式。最后，请将您的反馈意见发送至：wkservice@tup.tsinghua.edu.cn，我们将不胜感激。

进阶学习

《C# 4.0 完全参考手册》是 Herbert Schildt 系列编程书籍之一，下面列出了其他一些有用的书籍：

要学习 Java 编程，我们推荐下列书籍：

Java: The Complete Reference

Java: A Beginner's Guide

Swing: A Beginner's Guide

The Art of Java

Herb Schildt's Java Programming Cookbook

要学习 C++，下列书籍特别有用：

C++: The Complete Reference

C++: A Beginner's Guide

C++ From the Ground Up

STL Programming From the Ground Up

The Art of C++

Herb Schildt's C++ Programming Cookbook

如果想学习更多有关 C 语言的内容，那么我们推荐如下书籍：

C: The Complete Reference

目 录

第 I 部分 C# 语言

第 1 章 C#的起源	3
1.1 C#的族谱	3
1.1.1 C 语言: 现代编程的开端	3
1.1.2 OOP 和 C++语言的诞生	4
1.1.3 Internet 和 Java 的出现	4
1.1.4 C#的诞生	5
1.1.5 C#的发展	6
1.2 C#如何与 .NET Framework 相关	7
1.3 公共语言运行时的工作原理	7
1.4 托管和非托管代码	8
第 2 章 C#概述	9
2.1 面向对象编程	9
2.1.1 封装	10
2.1.2 多态	10
2.1.3 继承	11
2.2 简单示例一	11
2.2.1 C#命令行编译器 csc.exe	11
2.2.2 使用 Visual Studio IDE	12
2.2.3 逐行分析第一个示例程序	16
2.3 处理语法错误	18
2.4 改写示例一	19
2.5 简单示例二	19
2.6 另一种数据类型	21
2.7 两种控制语句	23
2.7.1 if 语句	23
2.7.2 for 循环	25

2.8 代码块	26
2.9 分号、定位和缩进	27
2.10 C#语言的关键字	28
2.11 标识符	29
2.12 .NET Framework 类库	30
第 3 章 数据类型、字面量和变量	31
3.1 数据类型的重要性	31
3.2 C#的值类型	31
3.3 整数类型	32
3.4 浮点类型	34
3.5 decimal 类型	36
3.6 字符类型	37
3.7 布尔类型	38
3.8 自定义输出格式	39
3.9 字面量	42
3.9.1 十六进制字面量	42
3.9.2 字符转义序列	43
3.9.3 字符串字面量	43
3.10 变量	45
3.10.1 初始化变量	45
3.10.2 动态初始化变量	46
3.10.3 隐式类型的变量	46
3.11 变量的作用域和生命周期	48
3.12 类型转换和强制类型转换	50
3.12.1 自动类型转换	50
3.12.2 强制转换不兼容的类型	51
3.13 表达式中的类型转换	54

第 4 章 运算符	59	第 7 章 数组和字符串	131
4.1 算术运算符	59	7.1 数组	131
4.2 关系和逻辑运算符	63	7.2 多维数组	135
4.3 赋值运算符	67	7.2.1 二维数组	135
4.4 按位运算符	69	7.2.2 三维或更多维的数组	137
4.4.1 按位与、或、异或和一元取 反运算符	69	7.2.3 初始化多维数组	137
4.4.2 移位运算符	75	7.3 交错数组	139
4.4.3 按位复合赋值	78	7.4 数组引用赋值	141
4.5 问号(?)运算符	78	7.5 Length 属性	142
4.6 空格和圆括号	79	7.6 隐式类型的数组	145
4.7 运算符优先级	80	7.7 foreach 循环	147
第 5 章 程序控制语句	81	7.8 字符串	150
5.1 if 语句	81	7.8.1 构造 string 类型对象	151
5.1.1 if 语句嵌套	82	7.8.2 操作字符串	151
5.1.2 if-else-if 阶梯结构	83	7.8.3 字符串数组	155
5.2 switch 语句	84	7.8.4 字符串是不可改变的	157
5.3 for 循环	88	7.8.5 在 switch 语句中使用 字符串	158
5.4 while 循环	96	第 8 章 方法和类	159
5.5 do-while 循环	97	8.1 控制对类成员的访问	159
5.6 foreach 循环	98	8.1.1 C# 的访问修饰符	159
5.7 使用 break 语句退出循环	98	8.1.2 公有访问和私有访问的 应用	161
5.8 使用 continue 语句	101	8.1.3 访问控制: 案例研究	161
5.9 goto 语句	102	8.2 给方法传递引用	166
第 6 章 类和对象	105	8.3 使用 ref 和 out 参数	170
6.1 类的基础知识	105	8.3.1 使用 ref 关键字	170
6.1.1 类的通用形式	105	8.3.2 使用 out 关键字	172
6.1.2 定义类	106	8.3.3 对引用参数使用 ref 和 out 关键字	174
6.2 创建对象的方式	110	8.4 使用数量可变的参数	176
6.3 引用类型的变量和赋值	111		
6.4 方法	111		
6.4.1 给 Building 类添加方法	112		
6.4.2 从方法返回	114		
6.4.3 返回值	115		
6.4.4 使用参数	117		
6.4.5 给 Building 类添加带参数的 方法	119		
6.4.6 避免产生不可到达的代码	121		
6.5 构造函数	121		
6.5.1 带参数的构造函数	122		
6.5.2 给 Building 类添加构造 函数	123		
6.6 new 运算符	124		
6.7 垃圾回收和析构函数	125		
6.8 this 关键字	127		

8.5 返回对象	178	10.2 属性	255
8.6 方法重载	182	10.2.1 自动实现属性	260
8.7 构造函数重载	187	10.2.2 对属性使用对象 初始化器	260
8.8 对象初始化器	193	10.2.3 属性限制	261
8.9 可选实参	194	10.3 对存取器使用访问修饰符	261
8.9.1 可选实参与重载	196	10.4 使用索引器和属性	264
8.9.2 可选实参与多义性	196		
8.9.3 可选实参的实际示例	197		
8.10 命名实参	198	第 11 章 继承	271
8.11 Main()方法	200	11.1 继承基础	271
8.11.1 从 Main()方法返回值	200	11.2 成员访问和继承	274
8.11.2 给 Main()方法传递 参数	200	11.3 构造函数和继承	278
8.12 递归	202	11.4 继承和名称隐藏	285
8.13 理解 static 关键字	205	11.5 创建多层的层次结构	288
8.14 static 类	210	11.6 构造函数的调用	291
第 9 章 运算符重载	213	11.7 基类引用和派生对象	292
9.1 运算符重载基础	213	11.8 虚方法和重写	296
9.1.1 重载二元运算符	214	11.8.1 重写方法的原因	300
9.1.2 重载一元运算符	216	11.8.2 应用虚方法	300
9.2 处理针对 C#内置类型的 运算	220	11.9 使用抽象类	304
9.3 重载关系运算符	225	11.10 使用 sealed 关键字阻止 继承	308
9.4 重载 true 和 false	227	11.11 object 类	308
9.5 重载逻辑运算符	229	11.11.1 装箱和拆箱	310
9.5.1 重载逻辑运算符的简单 方式	229	11.11.2 object 类型是否是通用 数据类型	312
9.5.2 使用短路运算符	231		
9.6 转换运算符	236	第 12 章 接口、结构和枚举	315
9.7 运算符重载的注意事项	240	12.1 接口	315
9.8 运算符重载的另一个示例	241	12.2 使用接口类型的引用	320
第 10 章 索引器和属性	245	12.3 接口属性	322
10.1 索引器	245	12.4 接口索引器	324
10.1.1 创建一维索引器	245	12.5 接口的继承	326
10.1.2 索引器重载	249	12.6 接口继承引起的名称隐藏	327
10.1.3 索引器不需要底层的 数组	251	12.7 显式实现	327
10.1.4 多维索引器	252	12.8 在接口和抽象类之间选择	330
		12.9 .NET 标准接口	330
		12.10 结构	330
		12.11 枚举	335
		12.11.1 初始化枚举	337

12.11.2	指定枚举的基本类型	338	14.4	文件数据流和面向字节的文件 I/O 操作	376
12.11.3	使用枚举	338	14.4.1	打开和关闭文件	376
第 13 章	异常处理	341	14.4.2	使用 FileStream 类从文件中读取字节	378
13.1	SystemException 类	341	14.4.3	写入文件	380
13.2	异常处理的基础	341	14.4.4	使用 FileStream 类复制文件	382
13.2.1	使用 try 和 catch 关键字	342	14.5	基于字符的文件 I/O 操作	383
13.2.2	一个简单的异常示例	342	14.5.1	使用 StreamWriter 类	383
13.2.3	另一个异常示例	344	14.5.2	使用 StreamReader 类	385
13.3	未捕获异常的后果	345	14.6	重定向标准数据流	386
13.4	使用异常可以妥善地处理错误	347	14.7	读取和写入二进制数据	388
13.5	使用多条 catch 子句	347	14.7.1	BinaryWriter 类	388
13.6	捕获所有的异常	348	14.7.2	BinaryReader 类	389
13.7	嵌套 try 块	350	14.7.3	二进制 I/O 操作的程序示例	390
13.8	抛出异常	351	14.8	随机访问文件	395
13.9	finally 语句	353	14.9	使用 MemoryStream 类	396
13.10	进一步分析 Exception 类	355	14.10	StringReader 和 StringWriter 类	399
13.11	派生异常类	358	14.11	File 类	400
13.12	捕获派生类异常	361	14.11.1	使用 Copy() 方法复制文件	400
13.13	checked 和 unchecked 关键字	363	14.11.2	使用 Exists() 和 GetLastAccessTime() 方法	401
第 14 章	I/O 系统	367	14.12	将数值型字符串转换为内部表示格式	402
14.1	C# 的 I/O 依赖于数据流	367	第 15 章	委托、事件和 Lambda 表达式	407
14.1.1	字节数据流和字符数据流	367	15.1	委托	407
14.1.2	预定义数据流	367	15.1.1	委托的方法组转换	410
14.2	数据流类	368	15.1.2	使用实例方法作为委托	410
14.2.1	Stream 类	368	15.1.3	多播委托	412
14.2.2	字节数据流类	369	15.1.4	协变和逆变	414
14.2.3	字符数据流包装类	369	15.1.5	System.Delegate 类	416
14.2.4	二进制数据流	371			
14.3	控制台 I/O	371			
14.3.1	读取控制台输入	371			
14.3.2	使用 ReadKey() 方法	373			
14.3.3	写入控制台输出	375			

15.1.6 使用委托的原因	416	16.2.2 #if 和 #endif	461
15.2 匿名函数	416	16.2.3 #else 和 #elif	462
15.3 匿名方法	416	16.2.4 #undef	464
15.3.1 给匿名方法传递参数	417	16.2.5 #error	464
15.3.2 从匿名方法中返回值	418	16.2.6 #warning	465
15.3.3 在匿名方法中使用外部 变量	419	16.2.7 #line	465
15.4 Lambda 表达式	421	16.2.8 #region 和 #endregion	465
15.4.1 Lambda 运算符	421	16.2.9 #pragma	465
15.4.2 表达式 Lambda	422	16.3 程序集和 internal 访问 修饰符	466
15.4.3 语句 Lambda	424		
15.5 事件	427	第 17 章 运行时类型标识、反射和 特性	469
15.5.1 多播委托事件的示例	428	17.1 运行时类型标识	469
15.5.2 作为事件处理程序的实例 方法和 static 方法的 区别	430	17.1.1 使用 is 运算符测试 类型	469
15.5.3 使用事件存取器	432	17.1.2 使用 as 运算符	470
15.5.4 事件的其他功能	437	17.1.3 使用 typeof 运算符	472
15.6 对事件使用匿名方法和 Lambda 表达式	437	17.2 反射	473
15.7 .NET 事件的指导原则	438	17.3 使用反射	475
15.8 事件的应用: 案例研究	441	17.3.1 获取方法的相关信息	475
第 16 章 名称空间、预处理器和 程序集	445	17.3.2 GetMethods()方法的另 一种形式	478
16.1 名称空间	445	17.3.3 使用反射调用方法	479
16.1.1 名称空间的声明	445	17.3.4 获取 Type 对象的构造 函数	481
16.1.2 名称空间可以避免名称 冲突	448	17.3.5 从程序集获得类型	486
16.1.3 using 指令	449	17.3.6 全自动类型查询	491
16.1.4 using 指令的另一种 形式	451	17.4 特性	493
16.1.5 名称空间的合成	453	17.4.1 特性基础	494
16.1.6 嵌套名称空间	454	17.4.2 位置形参和命名形参	497
16.1.7 全局名称空间	456	17.5 3 个内置特性	501
16.1.8 使用名称空间别名 限定符(::)	456	17.5.1 AttributeUsage 特性	501
16.2 预处理器	460	17.5.2 Conditional 特性	501
16.2.1 #define	460	17.5.3 Obsolete 特性	502
		第 18 章 泛型	505
		18.1 泛型的概念	505
		18.2 一个简单的泛型示例	506

18.2.1 泛型类型基于类型实参而有所不同.....	509	18.15.3 委托变体.....	558
18.2.2 泛型如何实现类型安全.....	509	18.16 泛型类型的实例化.....	560
18.3 使用两个类型形参的泛型类.....	512	18.17 使用泛型时的一些限制.....	561
18.4 泛型类的通用形式.....	513	18.18 泛型小结.....	561
18.5 类型约束.....	513	第 19 章 LINQ.....	563
18.5.1 基类约束.....	514	19.1 LINQ 的基础知识.....	564
18.5.2 接口约束.....	522	19.1.1 简单查询.....	564
18.5.3 new()构造函数约束.....	525	19.1.2 查询可以多次执行.....	566
18.5.4 引用类型约束和值类型约束.....	527	19.1.3 查询中的数据类型的关联方式.....	567
18.5.5 使用约束建立两个类型形参之间的关系.....	529	19.1.4 查询的通用形式.....	568
18.5.6 使用多个约束.....	530	19.2 使用 where 子句筛选值.....	568
18.6 创建类型形参的默认值.....	531	19.3 使用 orderby 子句排序结果.....	570
18.7 泛型结构.....	533	19.4 深入讨论 select 子句.....	573
18.8 创建泛型方法.....	534	19.5 使用嵌套的 from 子句.....	577
18.8.1 调用泛型方法时显式地指定类型实参.....	536	19.6 使用 group 子句分组结果.....	578
18.8.2 为泛型方法指定约束.....	536	19.7 使用 into 子句创建延续.....	580
18.9 泛型委托.....	537	19.8 在查询中使用 let 子句创建变量.....	582
18.10 泛型接口.....	538	19.9 使用 join 子句连接两个序列.....	583
18.11 比较同一个类型形参的实例.....	542	19.10 匿名类型.....	586
18.12 泛型类的层次结构.....	546	19.11 创建组连接.....	588
18.12.1 使用泛型基类.....	546	19.12 查询方法.....	590
18.12.2 泛型派生类.....	548	19.12.1 基本查询方法.....	591
18.13 重写泛型类中的虚方法.....	549	19.12.2 使用查询方法创建查询.....	592
18.14 重载使用类型形参的方法.....	550	19.12.3 查询语法与查询方法的对比.....	594
18.15 泛型类型形参中的协变和逆变.....	552	19.12.4 更多与查询相关的扩展方法.....	594
18.15.1 在泛型接口中使用协变.....	552	19.13 延期执行查询和立即执行查询.....	596
18.15.2 在泛型接口中使用逆变.....	555	19.14 表达式树.....	597
		19.15 扩展方法.....	599
		19.16 PLINQ.....	601

第 20 章 不安全代码、指针、可空类型、dynamic 类型和其他主题

其他主题	603
20.1 不安全代码	603
20.1.1 指针的基础知识	604
20.1.2 使用 unsafe 关键字	605
20.1.3 使用 fixed 修饰符	606
20.1.4 通过指针访问结构成员	607
20.1.5 指针运算	607
20.1.6 指针的比较	609
20.1.7 指针和数组	609
20.1.8 指针和字符串	611
20.1.9 多重间接寻址	612
20.1.10 指针数组	613
20.1.11 sizeof	613
20.1.12 stackalloc	613
20.1.13 创建固定大小的缓冲区	614
20.2 可空类型	615
20.2.1 可空类型基础	616
20.2.2 表达式中的可空对象	617
20.2.3 “??” 运算符	618
20.2.4 在可空对象上使用关系和逻辑运算符	619
20.3 部分类型	620
20.4 部分方法	621
20.5 使用 dynamic 关键字创建 dynamic 类型	622
20.6 COM 互操作性	626
20.7 友元程序集	627
20.8 其他关键字	627
20.8.1 lock 关键字	627
20.8.2 readonly 关键字	628
20.8.3 const 和 volatile 关键字	629
20.8.4 using 语句	629
20.8.5 extern 关键字	630

第 II 部分 C#类库研究

第 21 章 研究 System 名称空间	637
21.1 System 名称空间的成员	637
21.2 Math 类	639
21.3 与内置值类型对应的 .NET 结构	644
21.3.1 整型结构	645
21.3.2 浮点类型结构	647
21.3.3 Decimal 结构	650
21.3.4 Char 结构	653
21.3.5 Boolean 结构	658
21.4 Array 类	659
21.4.1 排序和搜索数组	667
21.4.2 反转数组	670
21.4.3 复制数组	671
21.4.4 使用谓词	672
21.4.5 使用 Action 委托	673
21.5 BitConverter 类	674
21.6 使用 Random 类产生随机数	676
21.7 内存管理和 GC 类	677
21.8 Object 类	679
21.9 Tuple 类	679
21.10 IComparable 和 IComparable<T>接口	679
21.11 IEquatable<T>接口	680
21.12 IConvertible 接口	680
21.13 ICloneable 接口	680
21.14 IFormatProvider 和 IFormattable 接口	682
21.15 IObservable<T>和 IObservable<T>接口	683
第 22 章 字符串和格式化	685
22.1 C#中的字符串	685
22.2 String 类	685
22.2.1 字符串构造函数	686
22.2.2 String 类的字段、索引器和属性	686

22.2.3	字符串运算符.....	687	23.8	使用 Wait()、Pulse()和 PulseAll()方法实现线程 通信	747
22.2.4	字符串方法.....	687	23.9	死锁和竞争条件	751
22.2.5	填充和剪裁字符串.....	704	23.10	使用MethodImplAttribute 特性	751
22.2.6	插入、删除和替换.....	706	23.11	使用互斥锁和信号量.....	753
22.2.7	改变字母大小写.....	707	23.11.1	互斥锁	753
22.2.8	使用 Substring()方法.....	708	23.11.2	信号量	757
22.2.9	字符串扩展方法.....	708	23.12	使用事件	760
22.3	格式化	708	23.13	Interlocked 类	762
22.3.1	格式化概述	709	23.14	.NET 4.0 添加的同步类.....	764
22.3.2	数字格式说明符.....	709	23.15	通过 Abort()方法终止 线程	764
22.3.3	理解实参编号.....	711	23.15.1	Abort()方法的另一种 形式	766
22.4	使用 String.Format()和 ToString()方法格式化数据	712	23.15.2	取消 Abort()方法	767
22.4.1	使用 String.Format()方法 格式化值	712	23.16	挂起和恢复线程	769
22.4.2	使用 ToString()方法格式 化数据	714	23.17	判断线程的状态	769
22.5	自定义数字格式	715	23.18	使用主线程	769
22.6	格式化日期和时间	719	23.19	.NET 4.0 添加的其他多线程 功能	771
22.6.1	自定义日期和时间 格式	721	23.20	多线程提示	771
22.6.2	格式化时间间隔	723	23.21	启动独立的任务	771
22.7	格式化枚举	725			
第 23 章	多线程编程第 1 部分	727	第 24 章	多线程编程第 2 部分: 研究 任务并行库和 PLINQ.....	773
23.1	多线程基础	727	24.1	并行编程的两种方式	773
23.2	Thread 类	728	24.2	Task 类	774
23.2.1	创建和启动线程.....	729	24.2.1	创建任务	774
23.2.2	一些简单的改进	731	24.2.2	使用任务 ID.....	777
23.2.3	创建多个线程	732	24.3	使用等待方法	778
23.3	确定线程结束的时间	734	24.4	使用 TaskFactory 类启动 任务	781
23.4	为线程传递实参	737	24.5	将 Lambda 表达式用作 任务	782
23.5	IsBackground 属性	739	24.6	创建任务延续	783
23.6	线程优先级	739	24.7	从任务返回值	785
23.7	同步	741			
23.7.1	实现同步的另一种 方式	745			
23.7.2	Monitor 类和 lock 关键字 的关系	746			

24.8	取消任务和使用 AggregateException 异常	787	25.8.2	为泛型集合实现 IComparable<T>接口	861
24.9	其他一些任务功能	790	25.9	使用 IComparer 接口	863
24.10	Parallel 类	790	25.9.1	使用非泛型的 IComparer 接口	863
24.10.1	通过 Invoke()方法 并行化任务	791	25.9.2	使用泛型的 IComparer<T> 接口	865
24.10.2	使用 For()方法	793	25.10	使用 StringComparer 类	866
24.10.3	使用 ForEach()方法	799	25.11	通过枚举器访问集合	866
24.11	研究 PLINQ	800	25.11.1	使用枚举器	867
24.11.1	ParallelEnumerable 类	800	25.11.2	使用 IDictionary Enumerator 接口	868
24.11.2	使用 AsParallel()方法 并行化查询	801	25.12	实现 IEnumerable 和 IEnumerator 接口	869
24.11.3	使用 AsOrdered() 方法	802	25.13	使用迭代器	871
24.11.4	取消并行查询方法	802	25.13.1	停用迭代器	873
24.11.5	其他 PLINQ 功能	804	25.13.2	使用多条 yield 指令	874
24.11.6	有关 PLINQ 的注意 事项	804	25.13.3	创建命名迭代器	874
25.13.4			25.13.4	创建泛型迭代器	876
第 25 章	集合、枚举器和迭代器	805	25.14	集合初始化器	877
25.1	集合概述	805	第 26 章	使用 System.Net 名称空间 通过 Internet 联网	879
25.2	非泛型集合	806	26.1	System.Net 名称空间的 成员	879
25.2.1	非泛型接口	806	26.2	统一资源标识符	881
25.2.2	DictionaryEntry 结构	811	26.3	Internet 访问基础	881
25.2.3	非泛型集合类	811	26.3.1	WebRequest 类	882
25.3	使用 BitArray 类存储位	827	26.3.2	WebResponse 类	884
25.4	专用集合	829	26.3.3	HttpRequest 和 HttpWebResponse 类	885
25.5	泛型集合	830	26.3.4	第一个简单的示例	885
25.5.1	泛型接口	830	26.4	处理网络错误	887
25.5.2	KeyValuePair <TKey,TValue>结构	834	26.4.1	Create()方法产生的 异常	887
25.5.3	泛型集合类	834	26.4.2	GetResponse()方法产生 的异常	887
25.6	并行集合	853			
25.7	在集合中存储用户自定义 的类	857			
25.8	实现 IComparable 接口	860			
25.8.1	为非泛型集合实现 IComparable 接口	860			

26.4.3	GetResponseStream()方法		26.6.3	使用 LastModified	
	产生的异常	888		属性	894
26.4.4	使用异常处理	888	26.7	MiniCrawler: 案例研究	895
26.5	Uri 类	890	26.8	使用 WebClient 类	898
26.6	访问附加的 HTTP 响应		附录 A	文档注释快速参考	903
	信息	891	A.1	XML 注释标记	903
26.6.1	访问报头	892	A.2	编译文档注释	904
26.6.2	访问 cookie	893	A.3	XML 文档示例	905