

Schildt 的经典编程著作——针对 C# 3.0 进行了全面修订和更新

C# 3.0

The Complete Reference

C# 3.0

完全参考手册

“要找到在现实世界中使用 C# 的答案，只需购买这本不可或缺的参考手册即可！”

——Michael Howard, Microsoft

(美) Herbert Schildt
赵利通

著
译

- ▲ 全面透彻地介绍 C# 语言
- ▲ 涵盖 C# 3.0 的新功能，包括 LINQ、拉姆达表达式和匿名类型
- ▲ 包括数百个浅显明了的示例

免费
源代码
下载

Herbert Schildt 是最优秀的编程畅销书作者，其创作的编程书籍在全球的销量已逾 350 万册。



清华大学出版社

C# 3.0 完全参考手册

(美) Herbert Schildt 著
赵利通 译

清华大学出版社

北 京

Herbert Schildt

C# 3.0: The Complete Reference

EISBN: 978-0-07-158841-6

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 edition 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。

本书封面贴有 McGraw-Hill 公司防伪标签,无标签者不得销售。

北京市版权局著作权合同登记号: 01-2009-2130

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# 3.0 完全参考手册/(美)希尔特(Schildt, H.)著;赵利通译. —北京:清华大学出版社, 2010.3
书名原文: C# 3.0: The Complete Reference

ISBN 978-7-302-22072-5

I. ①C… II. ①希… ②赵… III. ①C 语言—程序设计—技术手册 IV. ①TP312-62

中国版本图书馆 CIP 数据核字(2010)第 026257 号

责任编辑:王 军 李 阳

装帧设计:孔祥丰

责任校对:成凤进

责任印制:何 芊

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:三河市新茂装订有限公司

经 销:全国新华书店

开 本:185×260 印 张:54.75 字 数:1435 千字

版 次:2010 年 3 月第 1 版 印 次:2010 年 3 月第 1 次印刷

印 数:1~3000

定 价:128.00 元

本书如存在文字不清、漏印、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)62770177 转 3103 产品编号:031786-01

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，但只有一小部分可称之为“伟大”。伟大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方便，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，伟大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#是微软为支持.NET Framework 而创建的语言，它建立在丰富的程序设计资源之上，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#还和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，所以 C#和 Java 更像是堂兄弟。例如，两者都支持分布式程序设计，并且都使用中间代码来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，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程序设计方式上的改变，程序设计语言得到了不断的发展，C#也不例外。在不断的提炼、适应和创新过程中，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 年发布最初的 1.0 版本以来不断向 C#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、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，这是 C#的最新版本，也是本书将要介绍的版

本。毫不夸张地说，C# 3.0 添加的功能已经重新定义了 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 和拉姆达表达式。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。拉姆达表达式则实现功能样式的语法，它使用 $\Rightarrow$ 拉姆达运算符，并且频繁用于 LINQ 表达式中。

在学习本书的过程中可以了解到，LINQ 和拉姆达表达式的组合代表 C# 中极端强大的功能子集。此外，它们都是创新的功能，重新定义了为许多不同类型的编程任务(而不仅是数据库查询)设计解决方案的方式。从本质上来说，这些功能允许按照新方式处理老问题，使用它们不仅可以简化解决方案，而且可以帮助从不同的角度形成关于问题的概念。这些功能改变了人们从事编程工作的思维方式。

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# 3.0，全书分为两个部分。第 I 部分全面讨论了 C# 语言，包括 C# 3.0 版本中增加的新特性。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、反射和预处理程序。

第 II 部分研究了 C# 类库，同时也是 .NET Framework 类库。这是一个巨大的类库，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，本部分还包括对集合、多线程处理和网络连接和 Windows Forms 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

本书的读者无需具有任何编程经验。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 2008 或更高版本来编译和运行 C# 3.0 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

本书是 Herbert Schildt 系列编程书籍之一，Herbert Schildt 网站(www.HerbSchildt.com)上展示了他所编著的更多书籍，并提供了他的联系方式。最后，请将您的反馈意见发送至：wkservice@vip.163.com，我们将不胜感激。

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: 现代程序 设计的开端	3
1.1.2 OOP 和 C++语言的创建	4
1.1.3 Internet 和 Java 的出现	4
1.1.4 C#的创建	5
1.1.5 C#的发展	6
1.2 C#如何与 .NET Framework 相关	7
1.3 公共语言运行库的工作原理	7
1.4 托管和非托管代码	8
第 2 章 C#概述	9
2.1 面向对象程序设计	9
2.1.1 封装	10
2.1.2 多态性	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	12
2.2.2 使用 Visual Studio IDE	13
2.2.3 逐行分析第一个示例程序	17
2.3 处理语法错误	18
2.4 改写示例一	19
2.5 简单示例二	20
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	28
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、直接量和变量	31
3.1 数据类型的重要性	31
3.2 C#的数据类型	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 直接量	42
3.9.1 十六进制直接量	42
3.9.2 字符转义序列	43
3.9.3 字符串直接量	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	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	6.4.5 给 Building 类添加带 参数的方法	119
3.13 表达式中的类型转换	54	6.4.6 避免产生不可到达的代码	121
第 4 章 运算符	59	6.5 构造函数	121
4.1 算术运算符	59	6.5.1 带参数的构造函数	122
4.2 关系和逻辑运算符	63	6.5.2 给 Building 类添加 构造函数	123
4.3 赋值运算符	67	6.6 new 运算符	124
4.4 按位运算符	68	6.7 垃圾回收和析构函数	125
4.4.1 按位与、或、异或和 取反运算符	69	6.8 this 关键字	127
4.4.2 移位运算符	75	第 7 章 数组和字符串	131
4.4.3 按位复合赋值	77	7.1 数组	131
4.5 问号(?)运算符	78	7.2 多维数组	135
4.6 空白符和圆括号	79	7.2.1 二维数组	135
4.7 运算符优先级	79	7.2.2 三维或更多维的数组	136
第 5 章 程序控制语句	81	7.2.3 初始化多维数组	137
5.1 if 语句	81	7.3 交错数组	138
5.1.1 if 语句嵌套	82	7.4 数组引用赋值	141
5.1.2 if-else-if 阶梯结构	83	7.5 Length 属性	142
5.2 switch 语句	84	7.6 隐式类型的数组	145
5.3 for 循环	88	7.7 foreach 循环	147
5.4 while 循环	96	7.8 字符串	150
5.5 do-while 循环	97	7.8.1 构造字符串	151
5.6 foreach 循环	98	7.8.2 操作字符串	151
5.7 使用 break 语句退出循环	98	7.8.3 字符串数组	154
5.8 使用 continue 语句	101	7.8.4 字符串是不可变的	156
5.9 goto 语句	102	7.8.5 在 switch 语句中使用 字符串	157
第 6 章 类和对象	105	第 8 章 方法和类	159
6.1 类基础	105	8.1 控制对类成员的访问	159
6.1.1 类的基本形式	105	8.1.1 C# 的访问修饰符	159
6.1.2 定义类	106	8.1.2 公有访问和私有 访问的应用	161
6.2 如何创建对象	110	8.1.3 访问控制: 案例分析	161
6.3 引用类型的变量和赋值	111	8.2 给方法传递引用	166
6.4 方法	111	8.3 使用 ref 和 out 参数	170
6.4.1 给 Building 类添加方法	112	8.3.1 使用 ref 关键字	170
6.4.2 从方法返回	114	8.3.2 使用 out 关键字	172
6.4.3 返回值	115		
6.4.4 使用参数	117		

8.3.3 对引用参数使用 ref 和 out	174	10.2.3 属性限制	255
8.4 使用数量可变的参数	176	10.3 对存取器使用访问修饰符	255
8.5 返回对象	178	10.4 使用索引器和属性	258
8.6 方法重载	182	第 11 章 继承	265
8.7 构造函数重载	187	11.1 继承基础	265
8.8 对象初始化器	193	11.2 成员访问和继承	268
8.9 Main()方法	194	11.3 构造函数和继承	272
8.9.1 从 Main()返回值	194	11.4 继承和名称隐藏	279
8.9.2 给 Main()传递参数	194	11.5 创建多级层次结构	282
8.10 递归	196	11.6 构造函数的调用	285
8.11 理解 static 关键字	199	11.7 基类引用和派生对象	286
8.12 静态类	204	11.8 虚方法和重写	290
第 9 章 运算符重载	207	11.8.1 重写方法的原因	294
9.1 运算符重载基础	207	11.8.2 应用虚方法	294
9.1.1 重载二元运算符	208	11.9 使用抽象类	298
9.1.2 重载一元运算符	210	11.10 使用 sealed 来阻止继承	302
9.2 处理针对 C#内置 类型的运算	214	11.11 object 类	302
9.3 重载关系运算符	219	11.11.1 装箱和拆箱	304
9.4 重载 true 和 false	221	11.11.2 object 是否是通用 数据类型	306
9.5 重载逻辑运算符	223	第 12 章 接口、结构和枚举	309
9.5.1 一种重载逻辑运算符的 简单方法	223	12.1 接口	309
9.5.2 使用短路运算符	226	12.2 使用接口类型的引用	314
9.6 转换运算符	230	12.3 接口属性	317
9.7 运算符重载的注意事项	234	12.4 接口索引器	318
9.8 运算符重载的另一个示例	235	12.5 接口的继承	320
第 10 章 索引器和属性	239	12.6 接口继承引起的名称隐藏	321
10.1 索引器	239	12.7 显式实现	321
10.1.1 创建一维索引器	239	12.8 在接口和抽象类之间选择	324
10.1.2 索引器重载	243	12.9 .NET 标准接口	324
10.1.3 索引器不需要一个 潜在的数组	245	12.10 结构	325
10.1.4 多维索引器	246	12.11 枚举	330
10.2 属性	249	12.11.1 初始化一个枚举	332
10.2.1 自动实现属性	254	12.11.2 指定枚举的基本类型	332
10.2.2 对属性使用对象 初始化器	254	12.11.3 使用枚举	332
		第 13 章 异常处理	335
		13.1 System.Exception 类	335

13.2	异常处理的基础.....	335	14.4.1	使用 StreamWriter 类.....	376
13.2.1	使用 try 和 catch.....	336	14.4.2	StreamReader 类.....	378
13.2.2	一个简单的异常示例.....	336	14.5	重定向标准数据流.....	380
13.2.3	另一个异常示例.....	338	14.6	读取和写入二进制数据.....	382
13.3	未捕获异常的后果.....	339	14.6.1	BinaryWriter.....	382
13.4	使用多个 catch 子句.....	341	14.6.2	BinaryReader.....	382
13.5	捕获所有的异常.....	342	14.6.3	二进制 I/O 操作的 程序示例.....	384
13.6	嵌套 try 块.....	344	14.7	随机访问文件.....	388
13.7	抛出异常.....	345	14.8	使用 MemoryStream.....	391
13.8	finally 语句.....	347	14.9	StringReader 和 StringWriter.....	393
13.9	进一步分析异常类.....	349	14.10	把数值型字符串转换为 内部表示格式.....	394
13.10	派生异常类.....	352			
13.11	捕获派生类异常.....	356	第 15 章	委托、事件和拉姆达 表达式.....	399
13.12	checked 语句和 unchecked 语句.....	357	15.1	委托.....	399
第 14 章	I/O 系统.....	361	15.1.1	委托的方法组转换.....	402
14.1	C# 的 I/O 依赖于数据流.....	361	15.1.2	使用实例方法作为委托.....	402
14.1.1	字节数据流和字符 数据流.....	361	15.1.3	多播委托.....	404
14.1.2	预定义数据流.....	361	15.1.4	协变和逆变.....	406
14.1.3	数据流类.....	362	15.1.5	System.Delegate.....	408
14.1.4	Stream 类.....	362	15.1.6	使用委托的原因.....	408
14.1.5	字节数据流类.....	363	15.2	匿名函数.....	408
14.1.6	字符数据流封装类.....	363	15.3	匿名方法.....	408
14.1.7	二进制数据流.....	365	15.3.1	给匿名方法传递参数.....	409
14.2	控制台 I/O.....	365	15.3.2	从匿名方法中返回 一个值.....	410
14.2.1	读取控制台输入.....	365	15.3.3	在匿名方法中使用 外部变量.....	411
14.2.2	使用 ReadKey() 方法.....	367	15.4	拉姆达表达式.....	413
14.2.3	写入控制台输出.....	369	15.4.1	拉姆达运算符.....	413
14.3	文件数据流和面向字节的 文件 I/O 操作.....	370	15.4.2	表达式拉姆达.....	413
14.3.1	打开和关闭文件.....	370	15.4.3	语句拉姆达.....	416
14.3.2	从 FileStream 中 读取字节.....	372	15.5	事件.....	418
14.3.3	写入文件.....	373	15.5.1	多播委托事件的示例.....	420
14.3.4	使用 FileStream 复制文件.....	374	15.5.2	作为事件处理程序的 实例方法和静态 方法的区别.....	422
14.4	基于字符的文件 I/O 操作.....	376			

15.5.3	使用事件存取器	424	17.2	反射	465
15.5.4	事件的其他特性	429	17.3	使用反射	467
15.6	对事件使用匿名方法和 拉姆达表达式	429	17.3.1	获取方法的相关信息	467
15.7	.NET 事件的规则	430	17.3.2	GetMethods()的 另一种形式	470
15.8	事件的应用: 案例分析	433	17.3.3	使用反射调用方法	471
第 16 章	名称空间、预处理器 和程序集	437	17.3.4	获取 Type 对象的 构造函数	474
16.1	名称空间	437	17.3.5	从程序集获得类型	478
16.1.1	名称空间的声明	437	17.3.6	全自动类型查询	483
16.1.2	名称空间可以避免 名称冲突	440	17.4	特性	486
16.1.3	using 命令	441	17.4.1	特性基础	486
16.1.4	using 命令的另 一种形式	443	17.4.2	创建特性	486
16.1.5	名称空间的合成	445	17.4.3	连接特性	487
16.1.6	嵌套名称空间	446	17.4.4	获取对象的特性	487
16.1.7	全局名称空间	448	17.4.5	位置参数和命名参数	489
16.1.8	使用名称空间别名 限定符(::)	448	17.5	三个内置特性	493
16.2	预处理器	452	17.5.1	AttributeUsage 特性	493
16.2.1	#define	452	17.5.2	Conditional 特性	494
16.2.2	#if 和 #endif	453	17.5.3	Obsolete 特性	495
16.2.3	#else 和 #elif	454	第 18 章	泛型	497
16.2.4	#undef	456	18.1	泛型概念	497
16.2.5	#error	456	18.2	一个简单的泛型示例	498
16.2.6	#warning	457	18.2.1	泛型类型因类型参数 的不同而不同	501
16.2.7	#line	457	18.2.2	泛型如何实现类型安全	501
16.2.8	#region 和 #endregion	457	18.3	使用两个类型参数的 泛型类	504
16.2.9	#pragma	457	18.4	泛型类的通用形式	505
16.3	程序集和 internal 访问 修饰符	458	18.5	类型约束	505
第 17 章	运行时类型标识、反射 和特性	461	18.5.1	基类约束	506
17.1	运行时类型标识	461	18.5.2	接口约束	514
17.1.1	使用 is 运算符测试类型	461	18.5.3	new()构造函数约束	518
17.1.2	使用 as 运算符	462	18.5.4	引用类型和值类型约束	519
17.1.3	使用 typeof 运算符	464	18.5.5	使用约束建立两个 类型参数之间的关系	522
			18.5.6	使用多重约束	523
			18.6	创建类型参数的默认值	524

18.7	泛型结构	525	19.13.2	使用查询方法 创建查询	576
18.8	创建泛型方法	526	19.13.3	查询语法与查询 方法的对比	578
18.8.1	调用泛型方法时 显式地指定类型实参	529	19.13.4	更多与查询相关的 扩展方法	578
18.8.2	为泛型方法指定约束	529	19.14	延期执行查询和立即 执行查询	581
18.9	泛型委托	529	19.15	表达式树	582
18.10	泛型接口	532	19.16	扩展方法	583
18.11	比较同一类型参数的实例	536	第 20 章	不安全代码、指针、空类型 和其他主题	587
18.12	泛型类的层次结构	539	20.1	不安全代码	587
18.12.1	使用泛型基类	539	20.1.1	指针基础	588
18.12.2	泛型派生类	541	20.1.2	使用 <code>unsafe</code> 关键字	589
18.13	重写泛型类中的虚方法	542	20.1.3	使用 <code>fixed</code> 修饰符	590
18.14	重载带类型参数的方法	544	20.1.4	通过指针访问结构成员	591
18.15	泛型类型的实例化	545	20.1.5	指针运算	591
18.16	使用泛型时的一些局限	546	20.1.6	指针的比较	593
18.17	小结	546	20.1.7	指针和数组	593
第 19 章	LINQ	547	20.1.8	指针和字符串	595
19.1	LINQ 的定义	547	20.1.9	多重间接寻址	596
19.2	LINQ 的基础知识	548	20.1.10	指针数组	597
19.2.1	简单查询	548	20.1.11	<code>sizeof</code>	597
19.2.2	查询可以多次执行	550	20.1.12	<code>stackalloc</code>	597
19.2.3	查询中的数据类型的 关联方式	551	20.1.13	创建固定大小的缓冲区	598
19.2.4	查询的一般形式	552	20.2	空类型	599
19.3	使用 <code>where</code> 子句筛选值	552	20.2.1	空类型基础	600
19.4	使用 <code>orderby</code> 子句排序结果	554	20.2.2	表达式中的空对象	601
19.5	深入讨论 <code>select</code> 子句	558	20.2.3	<code>??</code> 运算符	602
19.6	使用嵌套的 <code>from</code> 子句	561	20.2.4	在空对象上使用关系和 逻辑运算符	603
19.7	使用 <code>group</code> 子句分组结果	562	20.3	部分类型	604
19.8	使用 <code>into</code> 子句创建继续	564	20.4	部分方法	605
19.9	在查询中使用 <code>let</code> 子句 创建变量	566	20.5	友元程序集	606
19.10	使用 <code>join</code> 子句连接 两个序列	567	20.6	其他关键字	607
19.11	匿名类型	570	20.6.1	<code>lock</code> 关键字	607
19.12	创建组连接	572	20.6.2	<code>readonly</code> 关键字	607
19.13	查询方法	575			
19.13.1	基本查询方法	575			

20.6.3	const 和 volatile 关键字	608
20.6.4	using 语句	608
20.7	extern 关键字	609
20.7.1	声明 extern 方法	610
20.7.2	声明 extern 程序集别名	611

第 II 部分 C#类库研究

第 21 章 研究 System 名称空间 615

21.1	System 的成员	615
21.2	Math 类	617
21.3	与内置值类型对应的 .NET 结构	622
21.3.1	整型结构	623
21.3.2	浮点类型结构	625
21.3.3	Decimal 结构	628
21.3.4	Char 结构	633
21.3.5	Boolean 结构	638
21.4	Array 类	639
21.4.1	排序和搜索数组	646
21.4.2	反转数组	649
21.4.3	复制数组	649
21.4.4	使用谓词	650
21.4.5	使用 Action 委托	652
21.5	BitConverter 类	653
21.6	用 Random 产生随机数	655
21.7	内存管理和 GC 类	656
21.8	Object 类	657
21.9	IComparable 和 IComparable<T>接口	657
21.10	IEquatable<T>接口	658
21.11	IConvertible 接口	658
21.12	ICloneable 接口	658
21.13	IFormatProvider 接口和 IFormattable 接口	660

第 22 章 字符串和格式化 663

22.1	C#中的字符串	663
22.2	String 类	663

22.2.1	字符串构造函数	664
22.2.2	String 类的字段、 索引器和属性	664
22.2.3	字符串运算符	665
22.2.4	字符串方法	665
22.2.5	填充和剪裁字符串	679
22.2.6	插入、删除和替换	681
22.2.7	改变字母大小写	682
22.2.8	使用 Substring()方法	682
22.2.9	字符串扩展方法	683
22.3	格式化类型	683
22.3.1	格式化类型概述	683
22.3.2	数值型数据的格式 说明符	684
22.3.3	理解参数编号	686
22.4	使用 String.Format()和 ToString()格式化数据	686
22.4.1	使用 String.Format() 格式化值	686
22.4.2	使用 ToString() 格式化数据	689
22.5	自定义数字格式	690
22.6	格式化日期和时间	693
22.7	格式化枚举	697

第 23 章 多线程程序设计 701

23.1	多线程基础	701
23.2	Thread 类	702
23.2.1	创建和启动线程	702
23.2.2	一些简单的改进	705
23.2.3	创建多个线程	706
23.3	确定线程结束的时间	708
23.4	为线程传递参数	710
23.5	IsBackground 属性	712
23.6	线程优先级	713
23.7	同步	715
23.7.1	实现同步的另一种方式	719
23.7.2	Monitor 类和锁	720

23.8	使用 Wait()、Pulse()和 PulseAll()实现线程通信	721	24.7.2	为泛型集合实现 IComparable<T>接口	797
23.8.1	Wait()和 Pulse()的示例 ..	721	24.8	使用 IComparer 接口	799
23.8.2	死锁和竞争条件	725	24.8.1	使用非泛型的 IComparer	799
23.9	使用MethodImplAttribute 属性	725	24.8.2	使用泛型的 IComparer<T>	800
23.10	使用互斥锁和信号量	727	24.9	通过枚举器访问集合	802
23.10.1	互斥锁	727	24.9.1	使用枚举器	802
23.10.2	信号量	731	24.9.2	使用 IDictionaryEnumerator	803
23.11	使用事件	734	24.10	实现 IEnumerable 和 IEnumerator 接口	805
23.12	Interlocked 类	736	24.11	迭代器	806
23.13	终止线程	737	24.11.1	停用迭代器	808
23.13.1	Abort()的另一种形式 ..	739	24.11.2	使用多个 yield 指令 ..	809
23.13.2	取消 Abort()	740	24.11.3	创建命名迭代器	810
23.14	挂起和恢复线程	742	24.11.4	创建泛型迭代器	811
23.15	判断线程的状态	742	24.12	集合初始化器	812
23.16	使用主线程	742	第 25 章	通过 Internet 连网	813
23.17	多线程编程提示	744	25.1	System.Net 的成员	813
23.18	开启独立任务	744	25.2	统一资源标识符	815
第 24 章	集合、枚举器和迭代器	747	25.3	Internet 访问基础	815
24.1	集合概述	747	25.3.1	WebRequest 类	816
24.2	非泛型集合	748	25.3.2	WebResponse 类	818
24.2.1	非泛型接口	748	25.3.3	HttpWebRequest 类和 HttpWebResponse 类	819
24.2.2	DictionaryEntry 结构	752	25.3.4	第一个简单的示例	819
24.2.3	非泛型集合类	753	25.4	处理网络错误	821
24.3	使用 BitArray 类存储位	768	25.4.1	Create()产生的异常	822
24.4	专用集合	770	25.4.2	GetResponse() 产生的异常	822
24.5	泛型集合	771	25.4.3	GetResponseStream() 产生的异常	822
24.5.1	泛型接口	771	25.4.4	使用异常处理	822
24.5.2	KeyValuePair<TK,TV> 结构	775	25.5	Uri 类	824
24.5.3	泛型集合类	775	25.6	访问附加的 HTTP 响应信息	825
24.6	在集合中存储用户 自定义的类	793			
24.7	实现 IComparable 接口	795			
24.7.1	为非泛型集合实现 IComparable 接口	796			

25.6.1	访问报头.....	826	26.4	Windows 窗体.....	839
25.6.2	访问 Cookie.....	827	26.5	基于窗体的 Windows 框架程序.....	839
25.6.3	使用 LastModified 属性..	829	26.6	添加按钮.....	841
25.7	MiniCrawler: 案例分析.....	829	26.6.1	按钮概述.....	842
25.8	使用 WebClient.....	833	26.6.2	给窗体添加按钮.....	842
第 26 章	使用 System.Windows. Forms 创建基于窗体的 Windows 应用程序.....	837	26.6.3	简单的按钮示例.....	842
26.1	Windows 程序设计简史.....	837	26.7	消息处理.....	843
26.2	编写基于窗体的 Windows 应用程序的两种方式.....	838	26.8	使用消息框.....	845
26.3	Windows 与用户交互操作 的方法.....	838	26.9	添加菜单.....	848
			26.9.1	创建传统样式的主菜单..	848
			26.9.2	使用 MenuStrip 创建新式菜单.....	852
			附录 A	文档注释快速参考.....	857

C# 语言

第 I 部分主要讨论 C# 语言的元素，包括：关键字、语法和运算符。另外还描述了一些和 C# 语言紧密相关的 C# 基本技术，比如：I/O 操作和反射。

第 1 章：
C# 的起源

第 2 章：
C# 概述

第 3 章：
数据类型、直接量和变量

第 4 章：
运算符

第 5 章：
程序控制语句

第 6 章：
类和对象

第 7 章：
数组和字符串

第 8 章：
方法和类

第 9 章：
运算符重载

第 10 章：
索引器和属性

第 11 章：
继承

第 12 章：
接口、结构和枚举

第 13 章：
异常处理

第 14 章：
I/O 系统

第 15 章：
委托、事件和拉姆达表达式

第 16 章：
名称空间、预处理器和程序集

第 17 章：
运行时类型标识、反射和特性

第 18 章：
泛型

第 19 章：
LINQ

第 20 章：
不安全代码、指针、空类型和其他主题

C#的起源

C#是微软.NET 开发人员的首选语言，它具有的新功能经受住了时间的考验，且始终位于科技前沿，为现代企业计算环境提供了一种可用性强的编程方法。无论从哪个角度看，C#都是21 世纪最重要的编程语言之一。

本章的目的是回顾 C#发展的历程，包括：C#创建的原动力、设计理念以及它是如何受其他计算机语言影响的。本章也解释了 C#和.NET Framework 的相关性。正如您将看到的，C#语言和.NET Framework 协同工作，一起构建了一个高度优雅的编程环境。

1.1 C#的族谱

计算机语言并不是凭空存在的，相反，它们彼此相关，新语言都或多或少地受到它之前的语言的影响。类似于异花授粉的过程，一种语言的功能会被另一种语言沿用，但新的创新内容会被集成到现有的环境中，而陈旧的构造则会被消除。就这样，编程语言不断地进化，编程艺术也不断地完善。C#也不例外。

C#继承了多种程序设计语言的精髓，它直接继承了当今最成功的两种计算机语言——C 和 C++ 语言的功能，并且与 Java 有紧密联系。理解它们之间的关系对于理解 C#语言是很重要的，因此，我们将分析这三种语言的发展环境，以此来开始 C#语言的研究。

1.1.1 C 语言：现代程序设计的开端

C 语言的创建标志着现代程序设计时代的开始，它是 Dennis Ritchie 于 20 世纪 70 年代在一台使用 UNIX 操作系统的 DEC PDP-11 机上创建的。尽管一些早期的语言，最著名的如 Pascal 语言，已经取得了相当的成功，然而 C 语言首先建立了面向过程编程的规范，至今仍适用。

C 语言成长于 20 世纪 60 年代的“结构化程序设计”变革。在结构化程序设计兴起之前，大型程序很难编写，因为程序逻辑容易混乱会呈现所谓的“无头绪的代码”，比如掺杂大量纷乱而难以跟踪的跳转、调用和返回。而结构化语言通过增加定义明确的控制语句、带有局部变量的子程序以及其他方法改进了这个局面。通过使用结构化的技术，程序变得更加组织良好和可靠，并且