

<<代码大全>>

图书基本信息

书名：<<代码大全>>

13位ISBN编号：9787121033629

10位ISBN编号：7121033623

出版时间：2006-12

出版时间：电子工业出版社

作者：迈克康奈尔

页数：914

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

## &lt;&lt;代码大全&gt;&gt;

## 内容概要

《代码大全（第2版）》是著名IT畅销书作者、IEEE Software杂志前主编、具有20年编程与项目管理经验的Steve McConnell十余年前的经典著作的全新演绎：第2版做了全面的更新，增加了很多与时俱进的内容，包括对新语言、新的开发过程与方法论的讨论，等等。

这是一本百科全书式的软件构建手册，涵盖了软件构建活动的方方面面，尤其强调提高软件质量的种种实践方法。

作者特别注重源代码的可读性，详细讨论了类和函数命名、变量命名、数据类型和控制结构、代码布局等编程的最基本要素，也讨论了防御式编程、表驱动法、协同构建、开发者测试、性能优化等有效开发实践，这些都服务于软件的首要技术使命：管理复杂度。

为了培养程序员编写高质量代码的习惯，书中展示了大量高质量代码示例（以及用作对比的低质量代码），提高软件质量是降低开发成本的重要途径。

除此之外，《代码大全（第2版）（英文版）》归纳总结了来自专家的经验、业界研究以及学术成果，列举了大量软件开发领域的真实案例与统计数据，提高《代码大全（第2版）（英文版）》的说服力。

《代码大全（第2版）（英文版）》中所论述的技术不仅填补了初级与高级编程实践之间的空白，而且也为程序员们提供了一个有关软件开发技术的信息来源。

《代码大全（第2版）（英文版）》对经验丰富的程序员、技术带头人、自学的程序员及没有太多编程经验的学生都是大有裨益的。

## 作者简介

Steve McConnell 是Construx公司首席软件工程师。他是软件工程知识体系（SWEBOK）项目构建知识领域的先驱。Steve曾就职于微软、波音以及西雅图地区的一些公司，从事软件工程的研究。Steve McConnell是以下著作的作者：《快速开发Rapid Development》（1996）、《软件项目长存之道Software Project Survival Guide》（1998）、和《专业软件开发Professional Software Development》（2004）的作者。他的书作为杰出软件开发书籍，曾两次获得Software Development杂志的优震撼大奖。1998年，Steve被Software Development杂志的读者评为软件业最具影响力的三大人物之一，与比尔·盖茨（Bill Gates）和李纳斯·托瓦兹（Linus Torvalds）齐名。而且，Steve还是SPC（Software Productivity Center，加拿大软件进程改进公司）的ESTIMATE Professional（的一款计划和估算工具）主要开发者，Software Development Productivity award（软件开发生产力大奖）的获得者。Steve从1984年就开始从事桌面软件产业，现在在快速开发方法论、工程估算、软件架构实施、性能调整、系统整合、和第三方合同管理方面已经具有专业的技术。

## 书籍目录

Preface Acknowledgments List of Checklists List of Tables List of Figures Part I Laying the Foundation 1  
 Welcome to Software Construction 1.1 What Is Software Construction? 1.2  
 Why Is Software Construction Important? 1.3 How to Read This Book 2  
 Metaphors for a Richer Understanding of Software Development 2.1 The Importance of Metaphors 2.2  
 How to Use Software Metaphors 2.3 Common Software Metaphors 3  
 Measure Twice, Cut Once: Upstream Prerequisites 3.1 Importance of Prerequisites 3.2  
 Determine the Kind of Software You're Working On 3.3 Problem-Definition Prerequisite 3.4  
 Requirements Prerequisite 3.5 Architecture Prerequisite 3.6 Amount of Time to Spend on Upstream Prerequisites 4  
 Key Construction Decisions 4.1 Choice of Programming Language 4.2 Programming Conventions 4.3  
 Your Location on the Technology Wave 4.4 Selection of Major Construction Practices Part II  
 Creating High-Quality Code 5 Design in Construction 5.1 Design Challenges 5.2 Key Design Concepts 5.3  
 Design Building Blocks: Heuristics 5.4 Design Practices 5.5 Comments on Popular Methodologies 6 Working Classes 6.1  
 Class Foundations: Abstract Data Types (ADTs) 6.2 Good Class Interfaces 6.3 Design and Implementation Issues 6.4  
 Reasons to Create a Class 6.5 Language-Specific Issues 6.6 Beyond Classes: Packages 7 High-Quality Routines 7.1  
 Valid Reasons to Create a Routine 7.2 Design at the Routine Level 7.3 Good Routine Names 7.4  
 How Long Can a Routine Be? 7.5 How to Use Routine Parameters 7.6 Special Considerations in the Use of Functions 7.7  
 Macro Routines and Inline Routines 8 Defensive Programming 8.1 Protecting Your Program from Invalid Inputs 8.2  
 Assertions 8.3 Error-Handling Techniques 8.4 Exceptions 8.5  
 Barricade Your Program to Contain the Damage Caused by Errors 8.6 Debugging Aids 8.7  
 Determining How Much Defensive Programming to Leave in Production Code 8.8  
 Being Defensive About Defensive Programming 9 The Pseudocode Programming Process 9.1  
 Summary of Steps in Building Classes and Routines 9.2 Pseudocode for Pros 9.3  
 Constructing Routines by Using the PPP 9.4 Alternatives to the PPP Part III Variables 10  
 General Issues in Using Variables 10.1 Data Literacy 10.2 Making Variable Declarations Easy 10.3  
 Guidelines for Initializing Variables 10.4 Scope 10.5 Persistence 10.6 Binding Time 10.7  
 Relationship Between Data Types and Control Structures 10.8 Using Each Variable for Exactly One Purpose 11  
 The Power of Variable Names 11.1 Considerations in Choosing Good Names 11.2 Naming Specific Types of Data 11.3  
 The Power of Naming Conventions 11.4 Informal Naming Conventions 11.5 Standardized Prefixes 11.6  
 Creating Short Names That Are Readable 11.7 Kind of Names to Avoid 12 Fundamental Data Types 12.1  
 Numbers in General 12.2 Integers 12.3 Floating-Point Numbers 12.4 Characters and Strings 12.5 Boolean Variables 12.6  
 Enumerated Types 12.7 Named Constants 12.8 Arrays 12.9 Creating Your Own Types (Type Aliasing) 13  
 Unusual Data Types 13.1 Structures 13.2 Pointers 13.3 Global Data Part IV Statements 14  
 Organizing Straight-Line Code 14.1 Statements That Must Be in a Specific Order 14.2  
 Statements Whose Order Doesn't Matter 15 Using Conditionals 15.1 if Statements 15.2 case Statements 16  
 Controlling Loops 16.1 Selecting the Kind of Loop 16.2 Controlling the Loop 16.3  
 Creating Loops Easily -- From the Inside Out 16.4 Correspondence Between Loops and Arrays 17  
 Unusual Control Structures 17.1 Multiple Returns from a Routine 17.2 Recursion 17.3 goto 17.4  
 Perspective on Unusual Control Structures 18 Table-Driven Methods 18.1  
 General Considerations in Using Table-Driven Methods 18.2 Direct Access Tables 18.3 Indexed Access Tables 18.4  
 Stair-Step Access Tables 18.5 Other Examples of Table Lookups 19 General Control Issues 19.1 Boolean Expressions 19.2  
 Compound Statements (Blocks) 19.3 Null Statements 19.4 Taming Dangerously Deep Nesting 19.5  
 A Programming Foundation: Structured Programming 19.6 Control Structures and Complexity Part V  
 Code Improvements 20 The Software-Quality Landscape 20.1 Characteristics of Software Quality 20.2  
 Techniques for Improving Software Quality 20.3 Relative Effectiveness of Quality Techniques 20.4  
 When to Do Quality Assurance 20.5 The General Principle of Software Quality 21 Collaborative Construction 21.1

OverviewofCollaborativeDevelopmentPractices21.2 PairProgramming21.3 FormalInspections21.4  
 OtherKindsofCollaborativeDevelopmentPractices22 DeveloperTesting22.1  
 RoleofDeveloperTestinginSoftwareQuality22.2 RecommendedApproachtoDeveloperTesting22.3  
 BagofTestingTricks22.4 TypicalErrors22.5 Test-SupportTools22.6 ImprovingYourTesting22.7  
 KeepingTestRecords23 Debugging23.1 OverviewofDebuggingIssues23.2 FindingaDefect23.3 FixingaDefect23.4  
 PsychologicalConsiderationsinDebugging23.5 DebuggingTools--ObviousandNot-So-Obvious24 Refactoring24.1  
 KindsofSoftwareEvolution24.2 IntroductiontoRefactoring24.3 SpecificRefactorings24.4 RefactoringSafely24.5  
 RefactoringStrategies25 Code-TuningStrategies25.1 PerformanceOverview25.2 IntroductiontoCodeTuning25.3  
 KindsofFatandMolasses25.4 Measurement25.5 Iteration25.6 SummaryoftheApproachtoCodeTuning26  
 Code-TuningTechniques26.1 Logic26.2 Loops26.3 DataTransformations26.4 Expressions26.5 Routines26.6  
 RecodinginaLow-LevelLanguage26.7 TheMoreThingsChange,theMoreTheyStaytheSamePartVI  
 SystemConsiderations27 HowProgramSizeAffectsConstruction27.1 CommunicationandSize27.2  
 RangeofProjectSizes27.3 EffectofProjectSizeonErrors27.4 EffectofProjectSizeonProductivity27.5  
 EffectofProjectSizeonDevelopmentActivities28 ManagingConstruction28.1 EncouragingGoodCoding28.2  
 ConfigurationManagement28.3 EstimatingaConstructionSchedule28.4 Measurement28.5  
 TreatingProgrammersasPeople28.6 ManagingYourManager29 Integration29.1  
 ImportanceoftheIntegrationApproach29.2 IntegrationFrequency--PhasedorIncremental?29.3  
 IncrementalIntegrationStrategies29.4 DailyBuildandSmokeTest30 ProgrammingTools30.1 DesignTools30.2  
 Source-CodeTools30.3 Executable-CodeTools30.4 Tool-OrientedEnvironments30.5  
 BuildingYourOwnProgrammingTools30.6 ToolFantasylandPart VII SoftwareCraftsmanship31 LayoutandStyle31.1  
 LayoutFundamentals31.2 LayoutTechniques31.3 LayoutStyles31.4 LayingOutControlStructures31.5  
 LayingOutIndividualStatements31.6 LayingOutComments31.7 LayingOutRoutines31.8 LayingOutClasses32  
 Self-DocumentingCode32.1 ExternalDocumentation32.2 ProgrammingStyleasDocumentation32.3  
 ToCommentorNottoComment32.4 KeystoEffectiveComments32.5 CommentingTechniques32.6  
 IEEEStandards33 PersonalCharacter33.1 IsntPersonalCharacterOfftheTopic?33.2 IntelligenceandHumility33.3  
 Curiosity33.4 IntellectualHonesty33.5 CommunicationandCooperation33.6 CreativityandDiscipline33.7  
 Laziness33.8 CharacteristicsThatDontMatterAsMuchAsYouMightThink33.9 Habits34  
 ThemesinSoftwareCraftsmanship34.1 ConquerComplexity34.2 PickYourProcess34.3  
 WriteProgramsforPeopleFirst,ComputersSecond34.4 ProgramintoYourLanguage,NotinIt34.5  
 FocusYourAttentionwiththeHelpofConventions34.6 PrograminTermsoftheProblemDomair34.7  
 WatchforFallingRocks34.8 Iterate,Repeatedly,AgainandAgain34.9 ThouShaltRendSoftwareandReligionAsunder35  
 WheretofindMoreInformation35.1 InformationAboutSoftwareConstruction35.2 TopicsBeyondConstruction35.3  
 Periodicals35.4 ASoftwareDevelopersReadingPlan35.5 JoiningaProfessionalOrganizationBibliographyIndex

## &lt;&lt;代码大全&gt;&gt;

## 编辑推荐

“《代码大全》第1版在我看来堪称软件工程领域的经典之作——而第2版则更棒！”

——Ralph Johnson, 伊利诺伊州立大学：《设计模式》(Design Patterns) 作者之一 “无论您是新手还是经验丰富的开发人员，《代码大全》(第2版)都能教会您思考编程的最佳方法。”

——Jeffrey Richter (WWW.wintellect.com), 《Microsoft NET框架程序设计》(AppfiedMicrosoft.NET Framework Programming) 作者 “这本书是讲述软件构建的权威指南——准备孤身前往荒岛的程序员只要带上这本书就足够了。”

——Diomidis Spinellis, 《代码阅读方法与实践》(Code Reading: The Open Soume Perspective) 作者 “Steve McConnell是一位既在一线实践,又能把其中奥妙讲个明白的少数人之一。”

——John Vlissides, IBM研究院; 《设计模式》(Design Patterns) 作者之一 “Steve McConnell比任何人都懂得如何构建软件;我们十分庆幸他能将其所有的深邃见解和实践经验写成这样一本重要而新颖的图书。”

——“Visual Basic之父” Alan Cooper, 《软件观念革命》(About Face 2.0) 作者 Steve McConnell的原作《代码大全》(第1版)是公认的关于编程的最佳实践指南之一,在过去的十多年间,本书一直在帮助开发人员编写更好的软件。

现在,作者将这本经典著作全新演绎,融入了最前沿的实践技术,加入了上百个崭新的代码示例,充分展示了软件构建的艺术性和科学性。

McConnell汇集了来自研究机构、学术界以及业界日常实践的主要知识,把最高效的技术和最重要的原理交织融会为这本既清晰又实用的指南。

无论您的经验水平如何,也不管您在怎样的开发环境中工作,也无论项目是大是小,本书都将激发您的思维——并帮助您构建高品质的代码。

从本书可以了解到如下这些经久不衰的技术与策略：  
 做出具有最小复杂度和最大创造性的设计  
 从协作式的开发中获益  
 应用防御式编程技术来减少并排查错误  
 发掘重构或改善代码的机会,并安全可靠地进行代码重构和改善  
 结合项目的规格合理选用恰当的构建技术  
 快速而有效地排除问题  
 尽早地正确解决关键构建问题  
 分别在项目的早期、中期以及后期加强代码的质量

<<代码大全>>

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>