

《代码大全（第2版）英文版》

图书基本信息

书名：《代码大全（第2版）英文版》

13位ISBN编号：9787121273152

出版时间：2016-4

作者：【美】Steve McConnell（史蒂夫·迈克康奈尔）

页数：952

版权说明：本站所提供下载的PDF图书仅提供预览和简介以及在线试读，请支持正版图书。

更多资源请访问：www.tushu000.com

《代码大全（第2版）英文版》

内容概要

《代码大全（第2版）英文版》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。《代码大全（第2版）英文版》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，《代码大全（第2版）英文版》中展示了大量高质量代码示例。此外，《代码大全（第2版）英文版》还归纳总结了来自专家的经验、业界研究及学术成果，列举了大量软件开发领域的真实案例与统计数据。《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。

《代码大全（第2版）英文版》

作者简介

Steve McConnell被公认为软件开发社区中的首要作者和发言人之一。他是Construx Software 公司的首席软件工程师。他所编著的图书包括曾被《Software Development》杂志授予优异产品震撼大奖（Jolt Award for product excellence）的《代码大全》（《Code Complete》）和《快速软件开发》（《Rapid Development》），以及《软件项目生存指南》（《Software Project Survival Guide》）和《专业软件开发》（《Professional Software Development》）等。

书籍目录

- 前言
- 致谢
- 核对清单
- 表目录
- 图目录
- 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 Kinds 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 Overview of Collaborative Development Practices
 - 21.2 Pair Programming
 - 21.3 Formal Inspections
 - 21.4 Other Kinds of Collaborative Development Practices
- 22 Developer Testing
 - 22.1 Role of Developer Testing in Software Quality
 - 22.2 Recommended Approach to Developer Testing
 - 22.3 Bag of Testing Tricks
 - 22.4 Typical Errors
 - 22.5 Test-Support Tools
 - 22.6 Improving Your Testing
 - 22.7 Keeping Test Records
- 23 Debugging

- 23.1 Overview of Debugging Issues
- 23.2 Finding a Defect
- 23.3 Fixing a Defect
- 23.4 Psychological Considerations in Debugging
- 23.5 Debugging Tools—Obvious and Not-So-Obvious
- 24 Refactoring
 - 24.1 Kinds of Software Evolution
 - 24.2 Introduction to Refactoring
 - 24.3 Specific Refactorings
 - 24.4 Refactoring Safely
 - 24.5 Refactoring Strategies
- 25 Code-Tuning Strategies
 - 25.1 Performance Overview
 - 25.2 Introduction to Code Tuning
 - 25.3 Kinds of Fat and Molasses
 - 25.4 Measurement
 - 25.5 Iteration
 - 25.6 Summary of the Approach to Code Tuning
- 26 Code-Tuning Techniques
 - 26.1 Logic
 - 26.2 Loops
 - 26.3 Data Transformations
 - 26.4 Expressions
 - 26.5 Routines
 - 26.6 Recoding in a Low-Level Language
 - 26.7 The More Things Change, the More They Stay the Same
- Part VI System Considerations
- 27 How Program Size Affects Construction
 - 27.1 Communication and Size
 - 27.2 Range of Project Sizes
 - 27.3 Effect of Project Size on Errors
 - 27.4 Effect of Project Size on Productivity
 - 27.5 Effect of Project Size on Development Activities
- 28 Managing Construction
 - 28.1 Encouraging Good Coding
 - 28.2 Configuration Management
 - 28.3 Estimating a Construction Schedule
 - 28.4 Measurement
 - 28.5 Treating Programmers as People
 - 28.6 Managing Your Manager
- 29 Integration
 - 29.1 Importance of the Integration Approach
 - 29.2 Integration Frequency—Phased or Incremental?
 - 29.3 Incremental Integration Strategies
 - 29.4 Daily Build and Smoke Test
- 30 Programming Tools
 - 30.1 Design Tools
 - 30.2 Source-Code Tools
 - 30.3 Executable-Code Tools

- 30.4 Tool-Oriented Environments
- 30.5 Building Your Own Programming Tools
- 30.6 Tool Fantasyland
- Part VII Software Craftsmanship
- 31 Layout and Style
 - 31.1 Layout Fundamentals
 - 31.2 Layout Techniques
 - 31.3 Layout Styles
 - 31.4 Laying Out Control Structures
 - 31.5 Laying Out Individual Statements
 - 31.6 Laying Out Comments
 - 31.7 Laying Out Routines
 - 31.8 Laying Out Classes
- 32 Self-Documenting Code
 - 32.1 External Documentation
 - 32.2 Programming Style as Documentation
 - 32.3 To Comment or Not to Comment
 - 32.4 Keys to Effective Comments
 - 32.5 Commenting Techniques
 - 32.6 IEEE Standards
- 33 Personal Character
 - 33.1 Isn't Personal Character Off the Topic?
 - 33.2 Intelligence and Humility
 - 33.3 Curiosity
 - 33.4 Intellectual Honesty
 - 33.5 Communication and Cooperation
 - 33.6 Creativity and Discipline
 - 33.7 Laziness
 - 33.8 Characteristics That Don't Matter As Much As You Might Think
 - 33.9 Habits
- 34 Themes in Software Craftsmanship
 - 34.1 Conquer Complexity
 - 34.2 Pick Your Process
 - 34.3 Write Programs for People First, Computers Second
 - 34.4 Program into Your Language, Not in It
 - 34.5 Focus Your Attention with the Help of Conventions
 - 34.6 Program in Terms of the Problem Domain
 - 34.7 Watch for Falling Rocks
 - 34.8 Iterate, Repeatedly, Again and Again
 - 34.9 Thou Shalt Rend Software and Religion Asunder
- 35 Where to Find More Information
 - 35.1 Information About Software Construction
 - 35.2 Topics Beyond Construction
 - 35.3 Periodicals
 - 35.4 A Software Developer's Reading Plan
 - 35.5 Joining a Professional Organization
- Bibliography
- Index

《代码大全（第2版）英文版》

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:www.tushu000.com