

Table of Contents

About the Author xiii

About the Technical Reviewer xv

Acknowledgments xvii

Chapter 1: Introduction..... 1

 Software Entropy 3

 Why C++?..... 4

 Clean Code 6

 C++11: The Beginning of a New Era..... 6

 Who This Book Is For..... 7

 Conventions Used in This Book 8

 Sidebars 9

 Notes, Tips, and Warnings 9

 Code Samples..... 9

 Companion Website and Source Code Repository 11

 UML Diagrams..... 12

Chapter 2: Build a Safety Net..... 13

 The Need for Testing 13

 Introduction to Testing 16

 Unit Tests..... 19

 What About QA? 21

 Rules for Good Unit Tests 22

 Test Code Quality 22

 Unit Test Naming 22

TABLE OF CONTENTS

Unit Test Independence 24

One Assertion per Test..... 25

Independent Initialization of Unit Test Environments..... 26

Exclude Getters and Setters 27

Exclude Third-Party Code 27

Exclude External Systems 28

What Do We Do with the Database? 28

Don't Mix Test Code with Production Code..... 29

Tests Must Run Fast 32

How Do You Find a Test's Input Data? 33

Test Doubles (Fake Objects) 36

Chapter 3: Be Principled..... 41

What Is a Principle? 41

KISS..... 42

YAGNI 43

DRY 44

 It's About Knowledge!..... 44

 Building Abstractions Is Sometimes Hard 45

Information Hiding 48

Strong Cohesion..... 53

Loose Coupling..... 56

Be Careful with Optimizations..... 60

Principle of Least Astonishment (PLA) 61

The Boy Scout Rule 62

 Collective Code Ownership..... 63

Chapter 4: Basics of Clean C++ 65

Good Names..... 66

 Names Should Be Self-Explanatory..... 68

 Use Names from the Domain..... 70

 Choose Names at an Appropriate Level of Abstraction..... 72

Avoid Redundancy When Choosing a Name	73
Avoid Cryptic Abbreviations.....	74
Avoid Hungarian Notation and Prefixes	75
Avoid Using the Same Name for Different Purposes	76
Comments.....	77
Let the Code Tell the Story.....	77
Do Not Comment Obvious Things	78
Don't Disable Code with Comments	79
Don't Write Block Comments.....	80
The Rare Cases Where Comments Are Useful	84
Functions	89
One Thing, No More!	93
Let Them Be Small.....	94
Function Naming	96
Use Intention-Revealing Names	97
Parameters and Return Values	98
About Old C-Style in C++ Projects.....	114
Choose C++ Strings and Streams over Old C-Style char*	115
Avoid Using printf(), sprintf(), gets(), etc.	117
Choose Standard Library Containers over Simple C-Style Arrays	122
Use C++ Casts Instead of Old C-Style Casts	125
Avoid Macros	128
Chapter 5: Advanced Concepts of Modern C++	131
Managing Resources	132
Resource Acquisition Is Initialization (RAII).....	134
Smart Pointers.....	135
Avoid Explicit New and Delete	144
Managing Proprietary Resources	144
We Like to Move It.....	146
What Are Move Semantics?	146
The Matter with Those lvalues and rvalues	148

TABLE OF CONTENTS

rvalue References.....	150
Don't Enforce Move Everywhere	152
The Rule of Zero	153
The Compiler Is Your Colleague.....	159
Automatic Type Deduction	159
Computations During Compile Time	164
Variable Templates	167
Don't Allow Undefined Behavior.....	169
Type-Rich Programming	171
Know Your Libraries	182
Take Advantage of <algorithm>	183
Take Advantage of Boost	194
More Libraries That You Should Know About.....	194
Proper Exception and Error Handling.....	196
Prevention Is Better Than Aftercare.....	197
An Exception Is an Exception, Literally!.....	202
If You Can't Recover, Get Out Quickly.....	204
Define User-Specific Exception Types	205
Throw by Value, Catch by const Reference.....	207
Pay Attention to the Correct Order of Catch Clauses	208
Interface Design.....	209
Attributes	210
Concepts: Requirements for Template Arguments	215
Chapter 6: Modularization	221
The Basics of Modularization	222
Criteria for Finding Modules	222
The Whole Enchilada	227
Object-Orientation.....	227
Object-Oriented Thinking.....	228
Principles for Good Class Design.....	230

Modules	281
The Drawbacks of #include	281
Modules to the Rescue	283
Under the Hood.....	284
Three Options for Using Modules	286
The Impact of Modules	290
Chapter 7: Functional Programming	293
What Is Functional Programming?	295
What Is a Function?	296
Pure vs Impure Functions.....	297
Functional Programming in Modern C++	299
Functional Programming with C++ Templates.....	299
Function-Like Objects (Functors)	302
Binders and Function Wrappers	312
Lambda Expressions	315
Generic Lambda Expressions (C++14).....	318
Lambda Templates (C++20)	319
Higher-Order Functions.....	322
Map, Filter, and Reduce	324
Pipelining with Range Adaptors (C++20).....	329
Clean Code in Functional Programming.....	333
Chapter 8: Test-Driven Development	335
The Drawbacks of Plain Old Unit Testing (POUT).....	336
Test-Driven Development as a Game Changer.....	338
The Workflow of TDD	339
TDD by Example: The Roman Numerals Code Kata.....	342
Preparations	343
The First Test	346
The Second Test	349
The Third Test and the Tidying Afterward	349

TABLE OF CONTENTS

More Sophisticated Tests with a Custom Assertion..... 354

It's Time to Clean Up Again 359

Approaching the Finish Line 362

Done! 364

The Advantages of TDD 367

When We Should Not Use TDD 369

TDD Is Not a Replacement for Code Reviews 371

Chapter 9: Design Patterns and Idioms 375

Design Principles vs Design Patterns 376

Some Patterns and When to Use Them 377

 Dependency Injection (DI)..... 378

 Adapter 394

 Strategy 396

 Command 402

 Command Processor 407

 Composite..... 412

 Observer 416

 Factories..... 422

 Facade 425

 The Money Class 427

 Special Case Object (Null Object) 431

What Is an Idiom? 435

 Some Useful C++ Idioms..... 436

Appendix A: Small UML Guide 451

Structural Modeling 452

 Component 452

 Class and Object..... 453

 Interface 457

 Association 459

TABLE OF CONTENTS

Generalization.....	462
Dependency.....	463
Template and Template Binding	465
Behavioral Modeling	466
Activity Diagram	466
Sequence Diagram	469
State Diagram.....	471
Stereotypes	474
 Bibliography	 477
Index.....	481