

# CONTENTS IN DETAIL

**FOREWORD by Peter Bindels** **xxv**

**ACKNOWLEDGMENTS** **xxix**

**INTRODUCTION** **xxxix**

About This Book . . . . . xxxix  
Who Should Read This Book? . . . . . xxxix  
What's in This Book? . . . . . xxxix  
    Part I: The C++ Core Language . . . . . xxxix  
    Part II: C++ Libraries and Frameworks . . . . . xxxix

**AN OVERTURE TO C PROGRAMMERS** **xxxvii**

Upgrading to Super C . . . . . xxxix  
    Function Overloading . . . . . xxxix  
    References . . . . . xl  
    auto Initialization . . . . . xlii  
    Namespaces and Implicit typedef of struct, union, and enum . . . . . xliii  
    Intermingling C and C++ Object Files . . . . . xlv  
C++ Themes . . . . . xlvi  
    Expressing Ideas Concisely and Reusing Code . . . . . xlvi  
    The C++ Standard Library . . . . . xlviii  
    Lambdas . . . . . xlix  
    Generic Programming with Templates . . . . . l  
    Class Invariants and Resource Management . . . . . li  
    Move Semantics . . . . . lv  
Relax and Enjoy Your Shoes . . . . . lvi

**PART I: THE C++ CORE LANGUAGE** **1**

**1**  
**UP AND RUNNING** **3**

The Structure of a Basic C++ Program . . . . . 4  
    Creating Your First C++ Source File . . . . . 4  
    Main: A C++ Program's Starting Point . . . . . 4  
    Libraries: Pulling in External Code . . . . . 5  
The Compiler Tool Chain . . . . . 5  
    Setting Up Your Development Environment . . . . . 6  
    Windows 10 and Later: Visual Studio . . . . . 6

macOS: Xcode . . . . .	8
Linux and GCC . . . . .	9
Text Editors . . . . .	13
Bootstrapping C++ . . . . .	13
The C++ Type System . . . . .	13
Declaring Variables. . . . .	14
Initializing a Variable’s State . . . . .	14
Conditional Statements . . . . .	15
Functions . . . . .	16
printf Format Specifiers . . . . .	18
Revisiting step_function . . . . .	20
Comments . . . . .	21
Debugging . . . . .	21
Visual Studio . . . . .	21
Xcode . . . . .	23
GCC and Clang Debugging with GDB and LLDB . . . . .	25
Summary . . . . .	28

## **2 TYPES 31**

Fundamental Types. . . . .	31
Integer Types . . . . .	32
Floating-Point Types. . . . .	35
Character Types . . . . .	36
Boolean Types . . . . .	38
The std::byte Type. . . . .	40
The size_t Type. . . . .	41
void . . . . .	42
Arrays . . . . .	42
Array Initialization . . . . .	42
Accessing Array Elements . . . . .	43
A Nickel Tour of for Loops . . . . .	43
C-Style Strings . . . . .	45
User-Defined Types. . . . .	49
Enumeration Types . . . . .	49
Plain-Old-Data Classes. . . . .	52
Unions . . . . .	53
Fully Featured C++ Classes . . . . .	54
Methods . . . . .	55
Access Controls . . . . .	56
Constructors . . . . .	58
Initialization . . . . .	59
The Destructor. . . . .	64
Summary . . . . .	65

## **3 REFERENCE TYPES 67**

Pointers. . . . .	67
Addressing Variables . . . . .	68
Dereferencing Pointers. . . . .	70

The Member-of-Pointer Operator . . . . .	71
Pointers and Arrays . . . . .	72
Pointers Are Dangerous . . . . .	74
void Pointers and std::byte Pointers . . . . .	76
nullptr and Boolean Expressions . . . . .	76
References . . . . .	77
Usage of Pointers and References . . . . .	77
Forward-Linked Lists: The Canonical Pointer-Based Data Structure . . . . .	78
Employing References . . . . .	79
this Pointers . . . . .	80
const Correctness . . . . .	81
const Member Variables . . . . .	83
Member Initializer Lists . . . . .	83
auto Type Deduction . . . . .	84
Initialization with auto . . . . .	84
auto and Reference Types . . . . .	85
auto and Code Refactorings . . . . .	85
Summary . . . . .	86

## **4 THE OBJECT LIFE CYCLE 89**

An Object's Storage Duration . . . . .	89
Allocation, Deallocation, and Lifetime . . . . .	90
Memory Management . . . . .	90
Automatic Storage Duration . . . . .	90
Static Storage Duration . . . . .	91
Thread-Local Storage Duration . . . . .	94
Dynamic Storage Duration . . . . .	95
Tracing the Object Life Cycle . . . . .	96
Exceptions . . . . .	98
The throw Keyword . . . . .	98
Using try-catch Blocks . . . . .	99
stdlib Exception Classes . . . . .	100
Handling Exceptions . . . . .	102
User-Defined Exceptions . . . . .	104
The noexcept Keyword . . . . .	104
Call Stacks and Exceptions . . . . .	105
A SimpleString Class . . . . .	107
Appending and Printing . . . . .	108
Using SimpleString . . . . .	109
Composing a SimpleString . . . . .	110
Call Stack Unwinding . . . . .	111
Exceptions and Performance . . . . .	113
Alternatives to Exceptions . . . . .	114
Copy Semantics . . . . .	115
Copy Constructors . . . . .	117
Copy Assignment . . . . .	119
Default Copy . . . . .	121
Copy Guidelines . . . . .	122

Move Semantics . . . . .	122
Copying Can Be Wasteful . . . . .	122
Value Categories . . . . .	124
lvalue and rvalue References . . . . .	124
The std::move Function. . . . .	125
Move Construction . . . . .	126
Move Assignment . . . . .	126
The Final Product . . . . .	128
Compiler-Generated Methods . . . . .	129
Summary . . . . .	130

## **5**

### **RUNTIME POLYMORPHISM**

## **133**

Polymorphism . . . . .	134
A Motivating Example . . . . .	134
Adding New Loggers . . . . .	136
Interfaces . . . . .	137
Object Composition and Implementation Inheritance . . . . .	137
Defining Interfaces . . . . .	138
Base Class Inheritance . . . . .	138
Member Inheritance . . . . .	139
virtual Methods. . . . .	140
Pure-Virtual Classes and Virtual Destructors. . . . .	142
Implementing Interfaces . . . . .	143
Using Interfaces . . . . .	144
Updating the Bank Logger. . . . .	144
Constructor Injection . . . . .	145
Property Injection . . . . .	146
Choosing Constructor or Property Injection . . . . .	146
Summary . . . . .	147

## **6**

### **COMPILE-TIME POLYMORPHISM**

## **149**

Templates . . . . .	149
Declaring Templates . . . . .	150
Template Class Definitions . . . . .	150
Template Function Definitions . . . . .	151
Instantiating Templates. . . . .	151
Named Conversion Functions . . . . .	151
const_cast . . . . .	152
static_cast . . . . .	152
reinterpret_cast . . . . .	153
narrow_cast . . . . .	154
mean: A Template Function Example . . . . .	155
Genericizing mean . . . . .	156
Template Type Deduction . . . . .	158
SimpleUniquePointer: A Template Class Example . . . . .	159
Type Checking in Templates . . . . .	161

Concepts . . . . .	163
Defining a Concept . . . . .	164
Type Traits . . . . .	164
Requirements . . . . .	166
Building Concepts from Requires Expressions . . . . .	167
Using Concepts. . . . .	168
Ad Hoc Requires Expressions . . . . .	172
static_assert: The Preconcepts Stopgap . . . . .	173
Non-Type Template Parameters . . . . .	174
Variadic Templates. . . . .	177
Advanced Template Topics . . . . .	177
Template Specialization. . . . .	178
Name Binding . . . . .	178
Type Function . . . . .	178
Template Metaprogramming. . . . .	178
Template Source Code Organization . . . . .	179
Polymorphism at Runtime vs. Compile Time . . . . .	179
Summary . . . . .	179

## **7 EXPRESSIONS 181**

Operators . . . . .	182
Logical Operators . . . . .	182
Arithmetic Operators . . . . .	182
Assignment Operators . . . . .	184
Increment and Decrement Operators . . . . .	185
Comparison Operators . . . . .	185
Member Access Operators. . . . .	185
Ternary Conditional Operator. . . . .	186
The Comma Operator . . . . .	186
Operator Overloading. . . . .	187
Overloading Operator new . . . . .	189
Operator Precedence and Associativity . . . . .	194
Evaluation Order . . . . .	196
User-Defined Literals . . . . .	197
Type Conversions. . . . .	198
Implicit Type Conversions. . . . .	198
Explicit Type Conversion . . . . .	201
C-Style Casts . . . . .	202
User-Defined Type Conversions. . . . .	203
Constant Expressions . . . . .	204
A Colorful Example . . . . .	205
The Case for constexpr . . . . .	207
Volatile Expressions . . . . .	207
Summary . . . . .	209

## **8 STATEMENTS 211**

Expression Statements. . . . .	211
Compound Statements . . . . .	212

Declaration Statements . . . . .	213
Functions . . . . .	213
Namespaces . . . . .	216
Type Aliasing . . . . .	220
Structured Bindings . . . . .	222
Attributes . . . . .	223
Selection Statements . . . . .	225
if Statements . . . . .	225
switch Statements . . . . .	229
Iteration Statements . . . . .	230
while Loops . . . . .	230
do-while Loops . . . . .	231
for Loops . . . . .	232
Ranged-Based for Loops . . . . .	234
Jump Statements . . . . .	238
break Statements . . . . .	238
continue Statements . . . . .	239
goto Statements . . . . .	239
Summary . . . . .	241

## 9

## FUNCTIONS

**243**

Function Declarations . . . . .	244
Prefix Modifiers . . . . .	244
Suffix Modifiers . . . . .	245
auto Return Types . . . . .	247
auto and Function Templates . . . . .	248
Overload Resolution . . . . .	249
Variadic Functions . . . . .	250
Variadic Templates . . . . .	251
Programming with Parameter Packs . . . . .	252
Revisiting the sum Function . . . . .	252
Fold Expressions . . . . .	253
Function Pointers . . . . .	254
Declaring a Function Pointer . . . . .	254
Type Aliases and Function Pointers . . . . .	255
The Function-Call Operator . . . . .	255
A Counting Example . . . . .	256
Lambda Expressions . . . . .	258
Usage . . . . .	258
Lambda Parameters and Bodies . . . . .	259
Default Arguments . . . . .	260
Generic Lambdas . . . . .	261
Lambda Return Types . . . . .	262
Lambda Captures . . . . .	262
constexpr Lambda Expressions . . . . .	268
std::function . . . . .	269
Declaring a Function . . . . .	269
An Extended Example . . . . .	270

The main Function and the Command Line . . . . .	272
The Three main Overloads . . . . .	272
Exploring Program Parameters . . . . .	273
A More Involved Example . . . . .	274
Exit Status . . . . .	276
Summary . . . . .	277

## **PART II: C++ LIBRARIES AND FRAMEWORKS 279**

### **10 TESTING 281**

Unit Tests . . . . .	282
Integration Tests . . . . .	282
Acceptance Tests . . . . .	282
Performance Tests . . . . .	282
An Extended Example: Taking a Brake . . . . .	283
Implementing AutoBrake . . . . .	285
Test-Driven Development . . . . .	286
Adding a Service-Bus Interface . . . . .	297
Unit-Testing and Mocking Frameworks . . . . .	304
The Catch Unit-Testing Framework . . . . .	304
Google Test . . . . .	310
Boost Test . . . . .	317
Summary: Testing Frameworks . . . . .	322
Mocking Frameworks . . . . .	323
Google Mock . . . . .	324
HippoMocks . . . . .	332
A Note on Other Mocking Options: Fakelt and Trompeloeil . . . . .	337
Summary . . . . .	337

### **11 SMART POINTERS 341**

Smart Pointers . . . . .	341
Smart Pointer Ownership . . . . .	342
Scoped Pointers . . . . .	342
Constructing . . . . .	342
Bring in the Oath Breakers . . . . .	343
Implicit bool Conversion Based on Ownership . . . . .	344
RAII Wrapper . . . . .	344
Pointer Semantics . . . . .	345
Comparison with nullptr . . . . .	346
Swapping . . . . .	346
Resetting and Replacing a scoped_ptr . . . . .	347
Non-transferability . . . . .	348
boost::scoped_array . . . . .	348
A Partial List of Supported Operations . . . . .	349

Unique Pointers . . . . .	349
Constructing . . . . .	350
Supported Operations . . . . .	350
Transferable, Exclusive Ownership . . . . .	350
Unique Arrays . . . . .	351
Deleters . . . . .	352
Custom Deleters and System Programming . . . . .	352
A Partial List of Supported Operations . . . . .	354
Shared Pointers . . . . .	355
Constructing . . . . .	356
Specifying an Allocator . . . . .	356
Supported Operations . . . . .	357
Transferable, Non-Exclusive Ownership . . . . .	358
Shared Arrays . . . . .	358
Deleters . . . . .	359
A Partial List of Supported Operations . . . . .	359
Weak Pointers . . . . .	360
Constructing . . . . .	361
Obtaining Temporary Ownership . . . . .	361
Advanced Patterns . . . . .	362
Supported Operations . . . . .	362
Intrusive Pointers . . . . .	363
Summary of Smart Pointer Options . . . . .	364
Allocators . . . . .	365
Summary . . . . .	367

## **12 UTILITIES 369**

Data Structures . . . . .	370
tribool . . . . .	370
optional . . . . .	372
pair . . . . .	374
tuple . . . . .	376
any . . . . .	378
variant . . . . .	379
Date and Time . . . . .	382
Boost DateTime . . . . .	383
Chrono . . . . .	387
Numerics . . . . .	392
Numeric Functions . . . . .	392
Complex Numbers . . . . .	393
Mathematical Constants . . . . .	394
Random Numbers . . . . .	396
Numeric Limits . . . . .	400
Boost Numeric Conversion . . . . .	401
Compile-Time Rational Arithmetic . . . . .	403
Summary . . . . .	405



<b>13</b>		
<b>CONTAINERS</b>		<b>407</b>
Sequence Containers . . . . .		408
Arrays . . . . .		408
Vectors . . . . .		415
Niche Sequential Containers . . . . .		423
Associative Containers . . . . .		434
Sets . . . . .		435
Unordered Sets . . . . .		442
Maps . . . . .		446
Niche Associative Containers . . . . .		453
Graphs and Property Trees . . . . .		454
The Boost Graph Library . . . . .		455
Boost Property Trees . . . . .		456
Initializer Lists . . . . .		457
Summary . . . . .		459

<b>14</b>		
<b>ITERATORS</b>		<b>463</b>
Iterator Categories . . . . .		464
Output Iterators . . . . .		464
Input Iterators . . . . .		466
Forward Iterators . . . . .		467
Bidirectional Iterators . . . . .		468
Random-Access Iterators . . . . .		469
Contiguous Iterators . . . . .		471
Mutable Iterators . . . . .		471
Auxiliary Iterator Functions . . . . .		472
std::advance . . . . .		472
std::next and std::prev . . . . .		473
std::distance . . . . .		475
std::iter_swap . . . . .		475
Additional Iterator Adapters . . . . .		476
Move Iterator Adapters . . . . .		476
Reverse Iterator Adapters . . . . .		477
Summary . . . . .		479

<b>15</b>		
<b>STRINGS</b>		<b>481</b>
std::string . . . . .		482
Constructing . . . . .		482
String Storage and Small String Optimizations . . . . .		485
Element and Iterator Access . . . . .		486
String Comparisons . . . . .		488
Manipulating Elements . . . . .		489
Search . . . . .		494
Numeric Conversions . . . . .		498

String View . . . . .	500
Constructing . . . . .	501
Supported string_view Operations . . . . .	502
Ownership, Usage, and Efficiency . . . . .	502
Regular Expressions . . . . .	503
Patterns . . . . .	504
basic_regex . . . . .	506
Algorithms . . . . .	506
Boost String Algorithms . . . . .	510
Boost Range . . . . .	510
Predicates . . . . .	511
Classifiers . . . . .	512
Finders . . . . .	514
Modifying Algorithms . . . . .	515
Splitting and Joining . . . . .	517
Searching . . . . .	519
Boost Tokenizer . . . . .	520
Localizations . . . . .	521
Summary . . . . .	521

## 16

### STREAMS

**523**

Streams. . . . .	523
Stream Classes . . . . .	524
Stream State . . . . .	530
Buffering and Flushing . . . . .	532
Manipulators . . . . .	533
User-Defined Types . . . . .	535
String Streams . . . . .	538
File Streams . . . . .	541
Stream Buffers . . . . .	546
Random Access . . . . .	548
Summary . . . . .	549

## 17

### FILESYSTEMS

**551**

Filesystem Concepts . . . . .	552
std::filesystem::path . . . . .	552
Constructing Paths . . . . .	552
Decomposing Paths . . . . .	553
Modifying Paths . . . . .	554
Summary of Filesystem Path Methods . . . . .	555
Files and Directories . . . . .	557
Error Handling . . . . .	557
Path-Composing Functions . . . . .	558
Inspecting File Types . . . . .	559
Inspecting Files and Directories . . . . .	561
Manipulating Files and Directories . . . . .	562

Directory Iterators . . . . .	564
Constructing . . . . .	564
Directory Entries . . . . .	565
Recursive Directory Iteration . . . . .	567
fstream Interoperation . . . . .	569
Summary . . . . .	570

## 18

### **ALGORITHMS 573**

Algorithmic Complexity . . . . .	574
Execution Policies . . . . .	575
Non-Modifying Sequence Operations . . . . .	576
all_of . . . . .	576
any_of . . . . .	577
none_of . . . . .	578
for_each . . . . .	579
for_each_n . . . . .	580
find, find_if, and find_if_not . . . . .	581
find_end . . . . .	582
find_first . . . . .	584
adjacent_find . . . . .	585
count . . . . .	586
mismatch . . . . .	587
equal . . . . .	588
is_permutation . . . . .	589
search . . . . .	590
search_n . . . . .	591
Mutating Sequence Operations . . . . .	592
copy . . . . .	592
copy_n . . . . .	593
copy_backward . . . . .	594
move . . . . .	595
move_backward . . . . .	596
swap_ranges . . . . .	597
transform . . . . .	598
replace . . . . .	600
fill . . . . .	601
generate . . . . .	602
remove . . . . .	603
unique . . . . .	605
reverse . . . . .	606
sample . . . . .	607
shuffle . . . . .	609
Sorting and Related Operations . . . . .	611
sort . . . . .	611
stable_sort . . . . .	612
partial_sort . . . . .	614
is_sorted . . . . .	615
nth_element . . . . .	616

Binary Search . . . . .	617
lower_bound . . . . .	617
upper_bound . . . . .	618
equal_range . . . . .	619
binary_search . . . . .	620
Partitioning Algorithms . . . . .	620
is_partitioned . . . . .	621
partition . . . . .	622
partition_copy . . . . .	622
stable_partition . . . . .	624
Merging Algorithms . . . . .	625
merge . . . . .	625
Extreme-Value Algorithms . . . . .	626
min and max . . . . .	626
min_element and max_element . . . . .	627
clamp . . . . .	628
Numeric Operations . . . . .	629
Useful Operators . . . . .	629
iota . . . . .	630
accumulate . . . . .	630
reduce . . . . .	631
inner_product . . . . .	632
adjacent_difference . . . . .	633
partial_sum . . . . .	634
Other Algorithms . . . . .	635
Boost Algorithm . . . . .	637

## **19 CONCURRENCY AND PARALLELISM 639**

Concurrent Programming . . . . .	640
Asynchronous Tasks . . . . .	640
Sharing and Coordinating . . . . .	647
Low-Level Concurrency Facilities . . . . .	658
Parallel Algorithms . . . . .	658
An Example: Parallel sort . . . . .	659
Parallel Algorithms Are Not Magic . . . . .	660
Summary . . . . .	661

## **20 NETWORK PROGRAMMING WITH BOOST ASIO 663**

The Boost Asio Programming Model . . . . .	664
Network Programming with Asio . . . . .	666
The Internet Protocol Suite . . . . .	666
Hostname Resolution . . . . .	667
Connecting . . . . .	669
Buffers . . . . .	671
Reading and Writing Data with Buffers . . . . .	674
The Hypertext Transfer Protocol (HTTP) . . . . .	676

Implementing a Simple Boost Asio HTTP Client . . . . .	677
Asynchronous Reading and Writing . . . . .	679
Serving . . . . .	683
Multithreading Boost Asio . . . . .	687
Summary . . . . .	689

**21**  
**WRITING APPLICATIONS** **691**

Program Support . . . . .	692
Handling Program Termination and Cleanup . . . . .	693
Communicating with the Environment . . . . .	697
Managing Operating System Signals . . . . .	699
Boost ProgramOptions . . . . .	700
The Options Description . . . . .	701
Parsing Options . . . . .	703
Storing and Accessing Options . . . . .	704
Putting It All Together . . . . .	705
Special Topics in Compilation . . . . .	708
Revisiting the Preprocessor . . . . .	708
Compiler Optimization . . . . .	710
Linking with C . . . . .	711
Summary . . . . .	712

**INDEX** **715**