

LEARNING MADE EASY



4th Edition

# C++

ALL-IN-ONE

for  
**dummies**<sup>®</sup>  
A Wiley Brand



**John Paul Mueller**

Author of *Functional Programming  
For Dummies*





# C++

## ALL-IN-ONE

4th Edition

**by John Paul Mueller**

for  
**dummies**<sup>®</sup>  
A Wiley Brand

## C++ All-in-One For Dummies®, 4th Edition

Published by: **John Wiley & Sons, Inc.**, 111 River Street, Hoboken, NJ 07030-5774, [www.wiley.com](http://www.wiley.com)

Copyright © 2021 by John Wiley & Sons, Inc., Hoboken, New Jersey

Published simultaneously in Canada

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without the prior written permission of the Publisher. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at <http://www.wiley.com/go/permissions>.

**Trademarks:** Wiley, For Dummies, the Dummies Man logo, Dummies.com, Making Everything Easier, and related trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and may not be used without written permission. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc. is not associated with any product or vendor mentioned in this book.

LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: THE PUBLISHER AND THE AUTHOR MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS WORK AND SPECIFICALLY DISCLAIM ALL WARRANTIES, INCLUDING WITHOUT LIMITATION WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES OR PROMOTIONAL MATERIALS. THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR EVERY SITUATION. THIS WORK IS SOLD WITH THE UNDERSTANDING THAT THE PUBLISHER IS NOT ENGAGED IN RENDERING LEGAL, ACCOUNTING, OR OTHER PROFESSIONAL SERVICES. IF PROFESSIONAL ASSISTANCE IS REQUIRED, THE SERVICES OF A COMPETENT PROFESSIONAL PERSON SHOULD BE SOUGHT. NEITHER THE PUBLISHER NOR THE AUTHOR SHALL BE LIABLE FOR DAMAGES ARISING HEREFROM. THE FACT THAT AN ORGANIZATION OR WEBSITE IS REFERRED TO IN THIS WORK AS A CITATION AND/OR A POTENTIAL SOURCE OF FURTHER INFORMATION DOES NOT MEAN THAT THE AUTHOR OR THE PUBLISHER ENDORSES THE INFORMATION THE ORGANIZATION OR WEBSITE MAY PROVIDE OR RECOMMENDATIONS IT MAY MAKE. FURTHER, READERS SHOULD BE AWARE THAT INTERNET WEBSITES LISTED IN THIS WORK MAY HAVE CHANGED OR DISAPPEARED BETWEEN WHEN THIS WORK WAS WRITTEN AND WHEN IT IS READ.

For general information on our other products and services, please contact our Customer Care Department within the U.S. at 877-762-2974, outside the U.S. at 317-572-3993, or fax 317-572-4002. For technical support, please visit <https://hub.wiley.com/community/support/dummies>.

Wiley publishes in a variety of print and electronic formats and by print-on-demand. Some material included with standard print versions of this book may not be included in e-books or in print-on-demand. If this book refers to media such as a CD or DVD that is not included in the version you purchased, you may download this material at <http://booksupport.wiley.com>. For more information about Wiley products, visit [www.wiley.com](http://www.wiley.com).

Library of Congress Control Number: 2020949804

ISBN: 978-1-119-60174-6

ISBN 978-1-119-60175-3 (ebk); ISBN 978-1-119-60173-9 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

# Contents at a Glance

<b>Introduction</b>	1
<b>Book 1: Getting Started with C++</b>	7
CHAPTER 1: Configuring Your Desktop System	9
CHAPTER 2: Configuring Your Mobile System	27
CHAPTER 3: Creating Your First C++ Application	45
CHAPTER 4: Storing Data in C++	69
CHAPTER 5: Directing the Application Flow	105
CHAPTER 6: Dividing Your Work with Functions	139
CHAPTER 7: Splitting Up Source Code Files	169
CHAPTER 8: Referring to Your Data Through Pointers	187
<b>Book 2: Understanding Objects and Classes</b>	225
CHAPTER 1: Working with Classes	227
CHAPTER 2: Using Advanced C++ Features	269
CHAPTER 3: Planning and Building Objects	309
CHAPTER 4: Building with Design Patterns	335
<b>Book 3: Understanding Functional Programming</b>	367
CHAPTER 1: Considering Functional Programming	369
CHAPTER 2: Working with Lambda Expressions	397
CHAPTER 3: Advanced Lambda Expressions	415
<b>Book 4: Fixing Problems</b>	427
CHAPTER 1: Dealing with Bugs	429
CHAPTER 2: Debugging an Application	443
CHAPTER 3: Stopping and Inspecting Your Code	457
CHAPTER 4: Traveling About the Stack	469
<b>Book 5: Advanced Programming</b>	479
CHAPTER 1: Working with Arrays, Pointers, and References	481
CHAPTER 2: Creating Data Structures	515
CHAPTER 3: Constructors, Destructors, and Exceptions	541
CHAPTER 4: Advanced Class Usage	571
CHAPTER 5: Creating Classes with Templates	601
CHAPTER 6: Programming with the Standard Library	637

<b>Book 6: Reading and Writing Files</b> .....	681
CHAPTER 1: Filing Information with the Streams Library .....	683
CHAPTER 2: Writing with Output Streams .....	697
CHAPTER 3: Reading with Input Streams .....	711
CHAPTER 4: Building Directories and Contents. ....	727
CHAPTER 5: Streaming Your Own Classes .....	737
 <b>Book 7: Advanced Standard Library Usage</b> .....	751
CHAPTER 1: Exploring the Standard Library Further .....	753
CHAPTER 2: Working with User-Defined Literals (UDLs) .....	779
CHAPTER 3: Building Original Templates .....	795
CHAPTER 4: Investigating Boost. ....	817
CHAPTER 5: Boosting up a Step .....	849
 <b>Index</b> .....	869

# Table of Contents

<b>INTRODUCTION</b>	1
About This Book	1
Foolish Assumptions	4
Icons Used in This Book	4
Beyond the Book	5
Where to Go from Here	6
<b>BOOK 1: GETTING STARTED WITH C++</b>	7
<b>CHAPTER 1: Configuring Your Desktop System</b>	9
Obtaining a Copy of C++ 20	10
Obtaining Code::Blocks	11
Installing Code::Blocks	12
Working with Windows	12
Working with Mac OS X	13
Using the standard Linux installation	14
Using the graphical Linux installation	15
Touring the Essential Code::Blocks Features	17
Starting Code::Blocks for the first time	18
Opening the sample projects	19
Viewing the essential windows	20
Using Other IDEs	25
<b>CHAPTER 2: Configuring Your Mobile System</b>	27
Obtaining CppDroid	28
Understanding why CppDroid is such a great choice	29
Getting your copy of CppDroid	31
Ensuring you get a good install	32
Considering Other Alternatives	32
Working with C4Droid	33
Getting multiple language support with AIDE	33
Using web-based IDEs	34
Touring the Essential CppDroid Features	35
Getting started with CppDroid	35
Accessing an example	37
Working with a simple online project	37
Accessing your source code	38
Considering differences with the desktop environment	39

Obtaining CppDroid Help.....	40
Working with the Help documentation .....	40
Getting community support.....	41
Using the free examples.....	42
Accessing the tutorials .....	43
<b>CHAPTER 3: Creating Your First C++ Application .....</b>	<b>45</b>
Code::Blocks Creating a Project.....	46
Understanding projects .....	46
Defining your first project .....	47
Building and executing your first application .....	52
Typing the Code.....	53
Starting with Main .....	55
Showing Information .....	55
Doing some math .....	60
Tabbing your output .....	66
Let Your Application Run Away .....	67
<b>CHAPTER 4: Storing Data in C++ .....</b>	<b>69</b>
Putting Your Data Places: Variables .....	70
Creating an integer variable.....	70
Declaring multiple variables .....	73
Changing values.....	74
Setting one variable equal to another .....	74
Initializing a variable .....	75
Creating a great name for yourself.....	76
Manipulating Integer Variables .....	78
Adding integer variables.....	78
Subtracting integer variables.....	82
Multiplying integer variables .....	84
Dividing integer variables.....	86
Characters .....	88
Null character.....	89
Nonprintable and other cool characters .....	89
Strings .....	93
Getting a part of a string .....	94
Changing part of a string .....	95
Adding onto a string .....	96
Adding two strings.....	97
Making Decisions Using Conditional Operators .....	98
Telling the Truth with Boolean Variables.....	100
Reading from the Console .....	102



<b>CHAPTER 5: Directing the Application Flow</b>	105
Doing This or Doing That	106
Evaluating Conditions in C++	107
Finding the right C++ operators	108
Combining multiple evaluations	110
Including Evaluations in C++ Conditional Statements	111
Deciding what if and also what else	112
Going further with the else and if	113
Repeating Actions with Statements That Loop	115
Understanding how computers use loops	116
Looping situations	116
Looping for	117
Performing a simple for loop	118
Using multiple initialization variables	123
Working with ranges	126
Placing a condition within the declaration	128
Letting C++ determine the type	129
Looping while	130
Doing while	132
Breaking and continuing	133
Breaking	134
Continuing	135
Nesting loops	136
<b>CHAPTER 6: Dividing Your Work with Functions</b>	139
Dividing Your Work	139
Calling a Function	144
Passing a variable	146
Passing multiple variables	147
Writing Your Own Functions	148
Defining the AddOne() function	149
Seeing how AddOne() is called	150
Taking the AddOne() Function apart	150
Considering the AddOne() parameter	151
Understanding the AddOne() name and type	152
Improving On the Basic Function	153
Using multiple parameters or no parameters	153
Returning nothing	156
Keeping your variables local	157
Forward references and function prototypes	159
Writing two versions of the same function	161

Calling All String Functions. . . . .	163
Inserting a string into a string . . . . .	163
Removing parts of a string. . . . .	164
Replacing parts of a string . . . . .	164
Using the string functions together . . . . .	164
Understanding main(). . . . .	165
<b>CHAPTER 7: Splitting Up Source Code Files. . . . .</b>	<b>169</b>
Creating Multiple Source Files. . . . .	170
Adding a new source code file. . . . .	170
Removing an existing source code file. . . . .	173
Creating a project with multiple existing files. . . . .	173
Getting multiple files to interact . . . . .	177
Sharing with Header Files. . . . .	179
Adding the header only once. . . . .	182
Using angle brackets or quotes. . . . .	182
Sharing Variables among Source Files . . . . .	183
Using the Mysterious Header Wrappers . . . . .	185
<b>CHAPTER 8: Referring to Your Data Through Pointers . . . . .</b>	<b>187</b>
Understanding the Changes in Pointers for C++ 20. . . . .	188
Avoiding broken code . . . . .	188
Considering the issues . . . . .	189
Writing cleaner and less bug-prone code . . . . .	191
Heaping and Stacking the Variables . . . . .	192
Getting a variable's address. . . . .	196
Changing a variable by using a pointer . . . . .	198
Pointing at a string . . . . .	200
Pointing to something else. . . . .	203
Tips on pointer variables . . . . .	204
Creating New Raw Pointers . . . . .	205
Using new . . . . .	206
Using an initializer . . . . .	208
Freeing Raw Pointers . . . . .	209
Working with Smart Pointers. . . . .	211
Creating smart pointers using <code>std::unique_ptr</code> and <code>std::shared_ptr</code> . . . . .	212
Defining nullable values using <code>std::optional</code> and <code>std::nullopt</code> . . . . .	216
Passing Pointer Variables to Functions . . . . .	218
Returning Pointer Variables from Functions. . . . .	221

<b>PART 2: UNDERSTANDING OBJECTS AND CLASSES</b>	225
<b>CHAPTER 1: Working with Classes</b>	227
Understanding Objects and Classes	227
Classifying classes and objects	230
Describing methods and data	231
Implementing a class	232
Separating method code	237
The parts of a class	240
Working with a Class	241
Accessing members	241
Using classes and raw pointers	244
Using classes and smart pointers	248
Passing objects to functions	249
Using const parameters in functions	251
Using the this pointer	252
Overloading methods	256
Starting and Ending with Constructors and Destructors	259
Starting with constructors	259
Ending with destructors	260
Sampling constructors and destructors	260
Adding parameters to constructors	263
Building Hierarchies of Classes	264
Creating a hierarchy in C++	265
Understanding types of inheritance	266
Creating and Using Object Aliases	267
<b>CHAPTER 2: Using Advanced C++ Features</b>	269
Filling Your Code with Comments	270
Converting Types	272
Understanding how int and string conversions work	272
Seeing int and string conversions in action	273
Considering other conversion issues	276
Reading from the Console	277
Understanding Preprocessor Directives	282
Understanding the basics of preprocessing	282
Creating constants and macros with #define	283
Performing conditional compilation	286
Exercising the basic preprocessor directives	288
Using Constants	292
Using Switch Statements	295

Supercharging enums with Classes .....	298
Working with Random Numbers. ....	300
Storing Data in Arrays. ....	302
Declaring and accessing an array .....	303
Arrays of pointers .....	304
Passing arrays to functions .....	306
Adding and subtracting pointers. ....	307
<b>CHAPTER 3: Planning and Building Objects .....</b>	<b>309</b>
Recognizing Objects .....	310
Observing the Mailboxes class .....	312
Observing the Mailbox class .....	314
Finding other objects .....	315
Encapsulating Objects. ....	316
Considering the Application Programming Interface. ....	316
Understanding properties .....	316
Choosing between private and protected .....	318
Defining a process. ....	318
Implementing properties .....	319
Building Hierarchies .....	322
Establishing a hierarchy .....	322
Protecting members when inheriting. ....	324
Overriding methods .....	330
Specializing with polymorphism .....	332
Getting abstract about things .....	333
<b>CHAPTER 4: Building with Design Patterns. ....</b>	<b>335</b>
Delving Into Pattern History. ....	336
Introducing a Simple Pattern: the Singleton .....	337
Using an existing pattern .....	337
Creating a singleton pattern class. ....	338
Watching an Instance with an Observer .....	341
Understanding the observer pattern .....	341
Defining an observer pattern class. ....	343
Observers and the Standard C++ Library. ....	346
Automatically adding an observer .....	347
Mediating with a Pattern .....	349
Defining the mediator pattern scenario. ....	350
Outlining the car example .....	351
Creating the car example .....	354

<b>BOOK 3: UNDERSTANDING FUNCTIONAL PROGRAMMING</b>	<b>367</b>
<b>CHAPTER 1: Considering Functional Programming</b>	<b>369</b>
Understanding How Functional Programming Differs	370
Defining an Impure Language	373
Considering the requirements	373
Understanding the C++ functional limitations	374
Seeing Data as Immutable	375
Working with immutable variables	376
Working with immutability in classes and structures	377
Creating constant expressions	378
Considering the Effects of State	381
Eliminating Side Effects	382
Contrasting declarations and functions	383
Associating functions with side effects	384
Removing side effects	385
Creating a declarative C++ example	387
Understanding the Role of auto	388
Passing Functions to Functions	390
Seeing a simple example of function input	391
Using transforms	393
Using Lambda Expressions for Implementation	394
<b>CHAPTER 2: Working with Lambda Expressions</b>	<b>397</b>
Creating More Readable and Concise C++ Code	398
Defining the Essential Lambda Expression	399
Defining the parts of a lambda expression	399
Relying on computer detection of return type	401
Using the auto keyword with lambda expressions	404
Using lambda expressions as macros	405
Developing with Lambda Expressions	406
Using lambda expressions with classes and structures	407
Working with the capture clause	408
Sorting data using a lambda expression	411
Specifying that the lambda expression throws exceptions	413
<b>CHAPTER 3: Advanced Lambda Expressions</b>	<b>415</b>
Considering the C++ 20 Lambda Extensions	416
Defining an immediate function	416
Using = and this in captures	417
Finding other changes	418

Working in Unevaluated Contexts . . . . .	418
Using Assignable Stateless Lambda Expressions . . . . .	420
Dealing with Pack Expansions . . . . .	422
Considering the template. . . . .	422
Processing the variables using recursion. . . . .	423
Processing the variables using a lambda expression . . . . .	424
<b>BOOK 4: FIXING PROBLEMS . . . . .</b>	<b>427</b>
<b>CHAPTER 1: Dealing with Bugs . . . . .</b>	<b>429</b>
It's Not a Bug. It's a Feature!. . . . .	430
Make Your Application Features Look Like Features . . . . .	431
Anticipating (Almost) Everything . . . . .	432
Considering menus . . . . .	432
Dealing with textual input . . . . .	435
Performing string processing . . . . .	437
Avoiding Mistakes, Plain and Simple . . . . .	441
<b>CHAPTER 2: Debugging an Application . . . . .</b>	<b>443</b>
Programming with Debuggers . . . . .	444
Running the debugger . . . . .	446
Recognizing the parts of the Code::Blocks debugger. . . . .	453
Debugging with Different Tools. . . . .	455
Debugging a Code::Blocks Application with Command-Line Arguments . . . . .	456
<b>CHAPTER 3: Stopping and Inspecting Your Code . . . . .</b>	<b>457</b>
Setting and Disabling Breakpoints . . . . .	458
Setting a breakpoint in Code::Blocks . . . . .	459
Enabling and disabling breakpoints . . . . .	460
Watching, Inspecting, and Changing Variables . . . . .	463
Watching the variables . . . . .	465
Changing values. . . . .	466
<b>CHAPTER 4: Traveling About the Stack . . . . .</b>	<b>469</b>
Stacking Your Data . . . . .	470
Moving about the stack . . . . .	471
Storing local variables. . . . .	473
Debugging with Advanced Features . . . . .	475
Viewing threads . . . . .	475
Tracing through assembly code . . . . .	475

## BOOK 5: ADVANCED PROGRAMMING.....479

<b>CHAPTER 1:</b>	<b>Working with Arrays, Pointers, and References.....</b>	<b>481</b>
	Building Up Arrays.....	482
	Declaring arrays.....	482
	Arrays and pointers.....	484
	Using multidimensional arrays.....	488
	Arrays and command-line parameters.....	492
	Allocating an array on the heap.....	494
	Deleting an array from the heap.....	494
	Storing arrays of pointers and arrays of arrays.....	495
	Building constant arrays.....	498
	Pointing with Pointers.....	498
	Becoming horribly complex.....	499
	Pointers to functions.....	505
	Pointing a variable to a method.....	506
	Pointing to static methods.....	509
	Referring to References.....	510
	Reference variables.....	510
	Returning a reference from a function.....	511
<b>CHAPTER 2:</b>	<b>Creating Data Structures.....</b>	<b>515</b>
	Working with Data.....	515
	The great variable roundup.....	516
	Expressing variables from either side.....	518
	Casting a spell on your data.....	520
	Comparing casting and converting.....	521
	Casting safely with C++.....	523
	Structuring Your Data.....	529
	Structures as component data types.....	531
	Equating structures.....	531
	Returning compound data types.....	532
	Naming Your Space.....	534
	Creating a namespace.....	534
	Employing using namespace.....	535
	Using variables.....	537
	Using part of a namespace.....	538
<b>CHAPTER 3:</b>	<b>Constructors, Destructors, and Exceptions.....</b>	<b>541</b>
	Constructing and Destructing Objects.....	542
	Overloading constructors.....	542
	Initializing members.....	543
	Adding a default constructor.....	548
	Functional constructors.....	550

Calling one constructor from another .....	553
Copying instances with copy constructors .....	555
When constructors go bad.....	557
Destroying your instances .....	558
Virtually inheriting destructors .....	560
Programming the Exceptions to the Rule .....	563
Creating a basic try...catch block.....	563
Using multiple catch blocks .....	565
Throwing direct instances .....	566
Catching any exception.....	567
Rethrowing an exception .....	568
Using a standard category .....	570
<b>CHAPTER 4: Advanced Class Usage .....</b>	<b>571</b>
Inherently Inheriting Correctly .....	572
Morphing your inheritance .....	572
Avoiding polymorphism .....	573
Adjusting access.....	574
Avoiding variable naming conflicts .....	575
Using class-based access adjustment .....	576
Returning something different, virtually speaking .....	577
Multiple inheritance .....	581
Virtual inheritance.....	584
Friend classes and functions .....	588
Using Classes and Types within Classes.....	591
Nesting a class .....	591
Types within classes .....	597
<b>CHAPTER 5: Creating Classes with Templates .....</b>	<b>601</b>
Templatizing a Class .....	602
Considering types .....	602
Defining the need for templates .....	602
Creating and using a template .....	605
Understanding the template keyword .....	607
Going Beyond the Basics .....	609
Separating a template from the function code.....	609
Including static members in a template.....	611
Parameterizing a Template .....	612
Putting different types in the parameter.....	613
Including multiple parameters .....	616
Working with non-type parameters .....	619



Typedefing a Template . . . . .	622
Deriving Templates . . . . .	623
Deriving classes from a class template . . . . .	623
Deriving a class template from a class . . . . .	626
Deriving a class template from a class template . . . . .	627
Templatizing a Function . . . . .	630
Overloading and function templates . . . . .	632
Templatizing a method . . . . .	635
<b>CHAPTER 6: Programming with the Standard Library . . . . .</b>	<b>637</b>
Architecting the Standard Library . . . . .	638
Containing Your Classes . . . . .	638
Storing in a vector . . . . .	639
Working with <code>std::array</code> . . . . .	642
Mapping your data . . . . .	643
Containing instances, pointers, or references . . . . .	644
Working with copies . . . . .	648
Comparing instances . . . . .	649
Iterating through a container . . . . .	655
A map of pairs in your hand . . . . .	658
The Great Container Showdown . . . . .	658
Associating and storing with a set . . . . .	658
Unionizing and intersecting sets . . . . .	662
Listing with list . . . . .	664
Stacking the deque . . . . .	669
Waiting in line with stacks and queues . . . . .	670
Copying Containers . . . . .	673
Creating and Using Dynamic Arrays . . . . .	675
Working with Unordered Data . . . . .	677
Using <code>std::unordered_set</code> to create an unordered set . . . . .	677
Manipulating unordered sets . . . . .	677
Working with Ranges . . . . .	679
<b>BOOK 6: READING AND WRITING FILES . . . . .</b>	<b>681</b>
<b>CHAPTER 1: Filing Information with the Streams Library . . . . .</b>	<b>683</b>
Seeing a Need for Streams . . . . .	684
Programming with the Streams Library . . . . .	686
Getting the right header file . . . . .	686
Opening a file . . . . .	687
Reading from a file . . . . .	690
Reading and writing a file . . . . .	691
Working with containers . . . . .	692
Handling Errors When Opening a File . . . . .	693
Flagging the ios Flags . . . . .	695

<b>CHAPTER 2:</b>	<b>Writing with Output Streams</b>	697
	Inserting with the << Operator	698
	Formatting Your Output	699
	Formatting with flags	700
	Specifying a precision	704
	Setting the width and creating fields	707
<b>CHAPTER 3:</b>	<b>Reading with Input Streams</b>	711
	Extracting with Operators	712
	Encountering the End of File	715
	Using the record count approach	715
	Using the EOF check approach	718
	Reading Various Types	720
	Understanding data reading issues	720
	Writing and reading string-type data	721
	Writing and reading structured data	724
<b>CHAPTER 4:</b>	<b>Building Directories and Contents</b>	727
	Manipulating Directories	728
	Creating a directory	728
	Deleting a directory	730
	Getting the Contents of a Directory	731
	Copying Files	733
	Copying with windows	734
	Using the quick-and-dirty method	734
	Moving and Renaming Files and Directories	735
<b>CHAPTER 5:</b>	<b>Streaming Your Own Classes</b>	737
	Streaming a Class for Text Formatting	738
	Understanding the process	739
	Considering the insertion implementation	739
	Considering the extraction implementation	741
	Manipulating a Stream	742
	What's a manipulator?	742
	Writing your own manipulator	744
	<b>BOOK 7: ADVANCED STANDARD LIBRARY USAGE</b>	751
<b>CHAPTER 1:</b>	<b>Exploring the Standard Library Further</b>	753
	Considering the Standard Library Categories	755
	Algorithms	755
	Atomic operations	757
	C Compatibility	759
	Concepts	759

Containers	760
Coroutines	760
Filesystem	761
Input/Output	761
Iterators	761
Localization	763
Numerics	763
Ranges	764
Regular Expressions	766
Strings	766
Thread Support	767
Utilities	767
Parsing Strings Using a Hash	768
Obtaining Information Using a Random Access Iterator	771
Locating Values Using the Find Algorithm	774
Using the Random Number Generator	776
Working with Temporary Buffers	777
<b>CHAPTER 2: Working with User-Defined Literals (UDLs)</b>	<b>779</b>
Understanding the Need for UDLs	780
Prefixes and suffixes	781
Differentiating between raw and cooked	784
Working with the UDLs in the Standard Library	785
std::basic_string	785
std::complex	788
std::chrono::duration	789
Creating Your Own UDLs	791
Developing a conversion UDL	792
Developing a custom type UDL	793
Using a custom UDL for side effects	794
<b>CHAPTER 3: Building Original Templates</b>	<b>795</b>
Deciding When to Create a Template	796
Defining the Elements of a Good Template	797
Creating a Basic Math Template	799
Building a Structure Template	801
Developing a Class Template	804
Considering Template Specialization	807
Creating a Template Library	809
Defining the library project	810
Configuring the library project	812
Coding the library	813
Using Your Template Library	815

<b>CHAPTER 4: Investigating Boost</b>	817
Considering the Standard Library Alternative	818
Understanding why the Standard Library contains Boost features	818
Defining the trade-offs of using the Standard Library	819
Understanding Boost	820
Boost features	821
Licensing	822
Paid support	823
Obtaining and Installing Boost for Code::Blocks	823
Unpacking Boost	823
Using the header-only libraries	825
Building the libraries	825
Testing the installation	827
Creating the Boost Tools	833
Using Boost.Build	836
Getting a successful build	836
Creating your own example	836
Using Inspect	837
Understanding BoostBook	840
Using QuickBook	841
Using bcp	843
Using Wave	845
Building Your First Boost Application Using Date Time	846
<b>CHAPTER 5: Boosting up a Step</b>	849
Parsing Strings Using RegEx	850
Adding the RegEx library	851
Creating the RegEx code	855
Breaking Strings into Tokens Using Tokenizer	857
Performing Numeric Conversion	858
Creating Improved Loops Using Foreach	862
Accessing the Operating System Using Filesystem	864
<b>INDEX</b>	869

# Introduction

**T**here are many general-purpose programming languages today, but few can claim to be the language of the millennium. C++ can make that claim, and for good reason:

- » It's powerful. You can write almost any program in it.
- » It's fast, and it's fully compiled. That's a good thing.
- » It's easy to use — if you have this book.
- » It's object oriented. If you're not sure what that is, don't worry. You can find out about it by reading this very book you're holding.
- » It supports functional programming techniques, which makes modeling math problems considerably easier and makes parallel processing easier. This book covers functional programming techniques, too.
- » It's portable. Versions are available for nearly every computer.
- » It's standardized. The American National Standards Institute (ANSI) and the International Standards Organization (ISO) both approve an official version.
- » It's continually updated to meet the changing challenges of the computer community.
- » It's popular. More people are using C++ because so many other people use it.

Sure, some people criticize C++. But most of these people don't truly understand C++ or are just having a bad day. Or both.

## About This Book

This book is a hands-on, roll-up-your-sleeves experience that gives you the opportunity to truly learn C++. This edition starts out by helping you get a great C++ installation in place. A lot of readers wrote to tell me that they simply couldn't get C++ to work for them, and I listened by adding configuration instructions in Book 1, Chapter 1. You can find instructions for working with the Mac, Linux, and Windows throughout the book. The examples are also tested to work on all three platforms.

*C++ All-in-One For Dummies*, 4th Edition, is devoted to working with C++ wherever you want to use it. Book 1, Chapter 2 even includes techniques for writing C++ code on your mobile device, although writing a complex application on your smart-phone would be understandably difficult because of the small device size.

At the very beginning, I start you out from square one. I don't assume any programming experience whatsoever. Everybody has to start somewhere. You can start here. Not to brag, but you are in the hands of a highly successful C++ developer who has shown thousands of people how to program, many of whom also started out from square one.

You already know C++? This book is great for you, too, because although I start discussing C++ from the beginning, I cover the important aspects of the language in depth. Even if you've used C++ in the past, this book gets you up to speed with the latest in C++ 14 and above innovations, including C++ 20 additions. Plus, this edition of the book focuses on all the latest programming strategies while removing some of the less used functionality of the past.

If you're interested in using the time-tested Object Oriented Programming (OOP) techniques that C++ developers have used for years, then Book 2 is where you want to look. You start with a view of classes, but eventually move into more advanced topics, including the use of programming patterns in Book 2 Chapter 4.

One of the most exciting additions to this edition is the use of functional programming techniques, which you can find in Book 3. Functional programming has become extremely popular because it makes modeling math problems significantly easier, and many people use functional programming techniques to solve modern data science problems. More important, functional programming can be a lot easier than earlier programming paradigms.

Every application out there has a bug or two. If you doubt this statement, just try to find one that is bug free—you won't. Book 4 includes all sorts of techniques you can use to make your application as bug free as possible before it leaves your machine and then help you find the bugs that others graciously point out later.

Book 5 is all about moving you from generalized programming strategies into the advanced strategies used by modern developers. It starts with a look at standardized structures for working with classes in a safe manner. The minibook takes you through

- » Simple structures, such as arrays
- » More advanced data management
- » The use of constructors, destructors, and exceptions

- » Templated programming
- » Use of the Standard Library (originally called the Standard Template Library or STL).

Everyone needs to work with files at some point. You use local, network, and Internet files today on a regular basis. Book 6 is all about working with files in various ways. This book includes topics on working with data streams as well.

The Standard Library is immense and there are entire books written about its use. *C++ All-in-One For Dummies*, 4th Edition, focuses on providing you with a really good overview that you can use to drill down into more detailed topics later. Besides looking at the Standard Library in more detail, you discover how to work with User Defined Literals (UDLs) and how to create your own templates. This book also delves into the Boost library, which is the library that has added more to Standard Library than just about any other source. Check out Book 7, Chapters 4 and 5 to learn about Boost. If you use C++ and don't use Boost, you're really missing out!

C++ is standardized, and you can use the information in this book on many different platforms. I wrote the samples using Mac OS X, SUSE Linux (some of the beta readers used other flavors of Linux), and Windows systems (with some testing on my ASUS tablet as well). In order to make this happen, I used a compiler called *Code::Blocks* that runs on almost every computer (Windows, Linux, and Macintosh) and CppDroid for my tablet. It doesn't matter which device you're using!

To make absorbing the concepts easy, this book uses the following conventions:

- » Text that you're meant to type just as it appears in the book is in **bold**. The exception is when you're working through a step list: Because each step is bold, the text to type is not bold.
- » Web addresses and programming code appear in monofont. If you're reading a digital version of this book on a device connected to the Internet, you can click or tap the web address to visit that website, like this: `https://www.dummies.com`.
- » When you need to type command sequences, you see them separated by a special arrow, like this: File→New File. In this example, you go to the File menu first and then select the New File entry on that menu.
- » When you see words in *italics* as part of a typing sequence, you need to replace that value with something that works for you. For example, if you see "Type **Your Name** and press Enter," you need to replace *Your Name* with your actual name.

# Foolish Assumptions

This book is designed for novice and professional alike. You can either read this book from cover to cover, or you can look up topics and treat the book as a reference guide — whichever works best for you. Keep it on your shelf, and have it ready to grab when you need to look something up. However, I've made some assumptions about your level of knowledge when I put the book together. The most important of these assumptions is that you already know how to use your device and work with the operating system that supports it. You also need to know how to perform tasks like downloading files and installing applications. A familiarity with the Internet is also required, and you need to know how to interact with it moderately well to locate the resources you need to work with the book. Finally, you must know how to work with archives, such as the ZIP file format.

## Icons Used in This Book

As you read this book, you see icons in the margins that indicate material of interest (or not, as the case may be). This section briefly describes each icon in this book.



TIP

Tips are nice because they help you save time or perform some task without a lot of extra work. The tips in this book are time-saving techniques or pointers to resources that you should try so that you can get the maximum benefit from C++. Most important, many of these tips will help you make sense of the overwhelming quantity of libraries and tools that C++ developers have created over the years.



WARNING

I don't want to sound like an angry parent or some kind of maniac, but you should avoid doing anything that's marked with a Warning icon. Otherwise, you might find that your application fails to work as expected, you get incorrect answers from seemingly bulletproof code, or (in the worst-case scenario) you lose data. Given where C++ appears, you might also send the next rocket off to Mars prematurely, make someone's thermostat misbehave, or cause nationwide power outages. Really, warnings are for everyone!



TECHNICAL  
STUFF

Whenever you see this icon, think advanced tip or technique. You might find these tidbits of useful information just too boring for words, or they could contain the solution you need to get a program running. Skip these bits of information whenever you like.





REMEMBER

If you don't get anything else out of a particular chapter or section, remember the material marked by this icon. This text usually contains an essential process or a bit of information that you must know to work with C++, or to perform development tasks successfully.

## Beyond the Book

If you want to email me, please do! Make sure you send your book-specific requests to:

`John@JohnMuellerBooks.com`

I get a lot of email from readers, so sometimes it takes me a while to answer. I try very hard to answer every book-specific question I receive, though, so I highly recommend contacting me with your questions. I want to ensure that your book experience is the best one possible. The blog category at <http://blog.johnmuellerbooks.com/categories/263/c-all-in-one-for-dummies.aspx> contains a wealth of additional information about this book. You can check out the website at <http://www.johnmuellerbooks.com/>.

This book isn't the end of your C++ programming experience — it's really just the beginning. I provide online content to make this book more flexible and better able to meet your needs. That way, as I receive email from you, I can address questions and tell you how updates to either Code::Blocks or the C++ language affect book content. You can also access other cool materials:

- » **Cheat Sheet:** You remember using crib notes in school to make a better mark on a test, don't you? You do? Well, a cheat sheet is sort of like that. It provides you with some special notes on things you can do with C++ that not every other developer knows. You can find the cheat sheet for this book at [www.dummies.com](http://www.dummies.com) and typing **C++ All-in-One For Dummies, 4th Edition** in the search field. It contains really neat information like the top ten mistakes developers make when working with C++, a list of header files that you use in most applications, and some of the C++ syntax that gives most developers problems.
- » **Updates:** Sometimes changes happen. For example, I might not have seen an upcoming change when I looked into my crystal ball during the writing of this book. In the past, such a situation simply meant that the book would become outdated and less useful, but you can now find updates to the book at [www.dummies.com](http://www.dummies.com). In addition to these updates, check out the blog posts with

answers to reader questions and demonstrations of useful book-related techniques at <http://blog.johnmuelเลอร์books.com/>.

» **Companion files:** Hey! Who really wants to type all the code in the book? Most readers would prefer to spend their time actually working through coding examples rather than typing. Fortunately for you, the source code is available for download, so all you need to do is read the book to learn C++ coding techniques. Each of the book examples even tells you precisely which example project to use. You can find these files by visiting [www.dummies.com/go/caiofd4e](http://www.dummies.com/go/caiofd4e).

Just in case you're worried about Code::Blocks, you can find complete download and installation instructions for it in Book 1, Chapter 1. Don't worry about which platform you use. This chapter includes instructions for Mac OS X, Linux, and Windows.

## Where to Go from Here

If you're just starting your C++ adventure, I highly recommend starting at either Book 1, Chapter 1 (for desktop developers) or Book 1, Chapter 2 (for mobile developers). You really do need to create a solid foundation before you can tackle the code in this book. If you're in a hurry and already have a C++ installation, you can always try starting with Book 1, Chapter 3.

Readers with a little more experience, who already know some C++ basics, can skip some of these introductory chapters, but you definitely don't want to skip Book 1, Chapter 8 because it contains a lot of pointer-related changes in current versions of C++. If you skip this chapter, you may find later that you have a hard time following the example code in the book because the newer examples use these pointer features.

An advanced reader with some idea of the current changes in C++ 20 could possibly skip Book 1, but scanning Book 2 is a good idea because there are some OOP changes you definitely want to know about. However, even for advanced readers, skipping Book 3 is a bad idea because modern development really is moving toward functional programming techniques.

# 1

## Getting Started with C++

# Contents at a Glance

<b>CHAPTER 1:</b>	<b>Configuring Your Desktop System</b>	9
	Obtaining a Copy of C++ 20.	10
	Obtaining Code::Blocks.	11
	Installing Code::Blocks	12
	Touring the Essential Code::Blocks Features.	17
	Using Other IDEs	25
<b>CHAPTER 2:</b>	<b>Configuring Your Mobile System</b>	27
	Obtaining CppDroid	28
	Considering Other Alternatives.	32
	Touring the Essential CppDroid Features	35
	Obtaining CppDroid Help.	40
<b>CHAPTER 3:</b>	<b>Creating Your First C++ Application</b>	45
	Code::Blocks Creating a Project.	46
	Typing the Code.	53
	Starting with Main	55
	Showing Information	55
	Let Your Application Run Away	67
<b>CHAPTER 4:</b>	<b>Storing Data in C++</b>	69
	Putting Your Data Places: Variables	70
	Manipulating Integer Variables	78
	Characters	88
	Strings	93
	Making Decisions Using Conditional Operators	98
	Telling the Truth with Boolean Variables	100
	Reading from the Console	102
<b>CHAPTER 5:</b>	<b>Directing the Application Flow</b>	105
	Doing This or Doing That	106
	Evaluating Conditions in C++	107
	Including Evaluations in C++ Conditional Statements	111
	Repeating Actions with Statements That Loop	115
	Looping for	117
	Looping while	130
	Doing while	132
	Breaking and continuing	133
	Nesting loops	136

*and more . . .*

- » Getting your own copy of C++ 20
- » Getting a copy of Code::Blocks
- » Creating a Code::Blocks work environment on your system
- » Seeing how Code::Blocks helps you perform tasks
- » Working with other IDEs

## Chapter **1**

# Configuring Your Desktop System

**T**his chapter is for those of you who have a desktop system and want to use it to create your application code. Chapter 2 discusses how to perform the same task using a mobile device (and provides you with some trade-offs between the two environments). Whether you use the desktop or the mobile solution, you need a copy of a compiler that supports C++ 20 features or some book examples won't work at all. This book relies on the GNU Compiler Collection (GCC) version 8.3 compiler because it provides great C++ 20 support (see [https://en.cppreference.com/w/cpp/compiler\\_support](https://en.cppreference.com/w/cpp/compiler_support)). The best way to obtain the version 8.3 compiler for your desktop system is to follow the steps in this chapter.

Before you can do anything interesting at all with C++, you need a copy of it installed on your system. Of course, this means going online, finding the location of the software that's appropriate for your platform, and then downloading it as necessary. If you use an Integrated Development Environment (IDE) such as Code::Blocks (the IDE used throughout this book), you get a copy of C++ with your installation, so you don't need to worry about reading the first section of this chapter. This book relies on your having a compiler capable of compiling C++ 20 code, which is the latest version of the language available at the time of this writing.



REMEMBER

Even though this book focuses on working with C++ on the Mac, Windows, and Linux platforms, you can actually use the techniques it provides on a great many other desktop systems. With this in mind, you'll find an overview of using C++ with other IDEs. As your platform becomes more esoteric, you'll find that fewer of the book examples work because your platform may require special programming techniques. The best option for working with this book is using a copy of Code::Blocks 17.12 with C++ 20 support installed on the Linux, Mac, or Windows platform.

## Obtaining a Copy of C++ 20

There is no product available named C++ 20. The C++ 20 standard simply says what the language contains and how someone should implement it. In other words, you can't just go online and get a copy of C++ 20; what you need to do instead is get a compiler vendor's implementation of the C++ 20 standard. For example, you can download the GNU Compiler Collection (GCC) version of C++ 20 from <https://gcc.gnu.org/releases.html>.



TIP

Every vendor will have a slightly different interpretation of this standard and could provide additions to the standard. In short, every compiler provides a unique version of C++. However, you also have the choice of not using the special features that the vendor provides, which means your source code is less susceptible to problems that occur when you use multiple compilers. The examples in this book are strictly written to the C++ 20 standard, so you shouldn't have a problem using them anywhere you want.

It's important that you also understand that a compiler is not the same as an Integrated Development Environment (IDE). The compiler is separate from the IDE in many cases and maintained by two separate parties. For example, the Code::Blocks IDE supports multiple compilers, and the GCC compiler works within multiple IDEs. The compiler is the important piece of software that turns your source code into an executable file that the operating system can run to produce the output you want.

The compiler you choose has to support the platforms you want to work with. For example, GCC supports Mac, Windows, and Linux development as well as some Acorn or (later) Advanced RISC Machine (ARM) processors (ARM doesn't officially stand for anything today). In fact, it may support other platforms by the time you read this chapter. Because it works in so many places, this book focuses on GCC, even though the examples will work with other compilers with some modification to overcome compiler differences.