

Library of Congress Cataloging-in-Publication Data

Schildt, Herbert.
 [C++ programming cookbook]
 Herb Schildt's C++ programming cookbook / Herb Schildt.
 p. cm.
 ISBN 978-0-07-148860-0 (alk. paper)
 1. C++ (Computer program language) I. Title. II. Title: C++
 programming cookbook.
 QA76.73.C153S325 2008
 005.13'3—dc22

2008014989

McGraw-Hill books are available at special quantity discounts to use as premiums and sales promotions, or for use in corporate training programs. To contact a representative, please visit the Contact Us pages at www.mhprofessional.com.

Herb Schildt's C++ Programming Cookbook

Copyright © 2008 by The McGraw-Hill Companies. All rights reserved. Printed in the United States of America. Except as permitted under the Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written permission of publisher, with the exception that the program listings may be entered, stored, and executed in a computer system, but they may not be reproduced for publication.

1234567890 DOC DOC 0198

ISBN: 978-0-07-148860-0
 MHID: 0-07-148860-X

Sponsoring Editor Wendy Rinaldi	Technical Editor Jim Keogh	Composition International Typesetting and Composition
Editorial Supervisor Patty Mon	Copy Editor Lisa McCoy	Illustration International Typesetting and Composition
Project Manager Vasundhara Sawhney, International Typesetting and Composition	Proofreader Andrea Fox	Art Director, Cover Jeff Weeks
Acquisitions Coordinator Mandy Canales	Indexer Sheryl Schildt	Cover Designer 12E Design
	Production Supervisor Jean Bodeaux	

Information has been obtained by McGraw-Hill from sources believed to be reliable. However, because of the possibility of human or mechanical error by our sources, McGraw-Hill, or others, McGraw-Hill does not guarantee the accuracy, adequacy, or completeness of any information and is not responsible for any errors or omissions or the results obtained from the use of such information.

Contents

Preface	xvii
1 Overview	1
What's Inside	1
How the Recipes Are Organized	2
A Few Words of Caution	3
C++ Experience Required	3
What Version of C++?	4
Two Coding Conventions	4
Returning a Value from main()	4
Using Namespace std?	4
2 String Handling	7
Overview of Null-Terminated Strings	8
Overview of the string Class	11
String Exceptions	16
Perform Basic Operations on Null-Terminated Strings	16
Step-by-Step	17
Discussion	17
Example	18
Options and Alternatives	19
Search a Null-Terminated String	20
Step-by-Step	21
Discussion	21
Example	21
Options and Alternatives	22
Reverse a Null-Terminated String	23
Step-by-Step	23
Discussion	24
Example	24
Options and Alternatives	25
Ignore Case Differences When Comparing Null-Terminated Strings	27
Step-by-Step	27
Discussion	28
Example	29
Options and Alternatives	31
Create a Search-and-Replace Function for Null-Terminated Strings	31
Step-by-Step	32
Discussion	32

Example	33
Options and Alternatives	36
Categorize Characters Within a Null-Terminated String	39
Step-by-Step	39
Discussion	40
Example	40
Bonus Example: Word Count	41
Options and Alternatives	43
Tokenize a Null-Terminated String	44
Step-by-Step	45
Discussion	45
Example	45
Options and Alternatives	47
Perform Basic Operations on string Objects	51
Step-by-Step	52
Discussion	52
Example	55
Options and Alternatives	58
Search a string Object	59
Step-by-Step	60
Discussion	60
Example	61
Bonus Example: A Tokenizer Class for string Objects	63
Options and Alternatives	65
Create a Search-and-Replace Function for string Objects	66
Step-by-Step	67
Discussion	67
Example	67
Options and Alternatives	69
Operate on string Objects Through Iterators	70
Step-by-Step	71
Discussion	71
Example	73
Options and Alternatives	75
Create Case-Insensitive Search and Search-and-Replace	
Functions for string Objects	76
Step-by-Step	77
Discussion	77
Example	78
Options and Alternatives	81
Convert a string Object into a Null-Terminated String	83
Step-by-Step	83
Discussion	83
Example	83
Options and Alternatives	85

Implement Subtraction for string Objects	85
Step-by-Step	86
Discussion	87
Example	88
Options and Alternatives	90
3 Working with STL Containers	93
STL Overview	94
Containers	94
Algorithms	94
Iterators	94
Allocators	95
Function Objects	95
Adaptors	96
Predicates	96
Binders and Negators	96
The Container Classes	96
Common Functionality	98
Performance Issues	101
Basic Sequence Container Techniques	102
Step-by-Step	103
Discussion	103
Example	105
Options and Alternatives	109
Use vector	111
Step-by-Step	111
Discussion	112
Example	115
Options and Alternatives	118
Use deque	118
Step-by-Step	119
Discussion	119
Example	120
Options and Alternatives	124
Use list	124
Step-by-Step	125
Discussion	125
Example	127
Options and Alternatives	130
Use the Sequence Container Adaptors: stack, queue, and priority_queue	132
Step-by-Step	132
Discussion	133
Example	135

Bonus Example: Use stack to Create a Four-Function Calculator . . .	137	Example	194
Options and Alternatives	140	Bonus Example: Extract Sentences from a Vector of Characters	195
Store User-Defined Objects in a Container	140	Options and Alternatives	197
Step-by-Step	140	Use search() to Find a Matching Sequence	199
Discussion	141	Step-by-Step	200
Example	141	Discussion	200
Options and Alternatives	144	Example	200
Basic Associative Container Techniques	145	Options and Alternatives	202
Step-by-Step	146	Reverse, Rotate, and Shuffle a Sequence	203
Discussion	147	Step-by-Step	204
Example	150	Discussion	204
Options and Alternatives	155	Example	204
Use map	156	Bonus Example: Use Reverse Iterators to	
Step-by-Step	157	Perform a Right-Rotate	206
Discussion	157	Options and Alternatives	207
Example	159	Cycle Through a Container with for_each()	208
Options and Alternatives	162	Step-by-Step	208
Use multimap	163	Discussion	208
Step-by-Step	163	Example	209
Discussion	163	Options and Alternatives	210
Example	165	Use transform() to Change a Sequence	211
Options and Alternatives	167	Step-by-Step	211
Use set and multiset	169	Discussion	212
Step-by-Step	170	Example	212
Discussion	170	Options and Alternatives	214
Example	172	Perform Set Operations	217
Bonus Example: Use multiset to Store Objects		Step-by-Step	217
with Duplicate Keys	174	Discussion	218
Options and Alternatives	178	Example	219
4 Algorithms, Function Objects, and Other STL Components	181	Options and Alternatives	221
Algorithm Overview	182	Permute a Sequence	222
Why Algorithms?	182	Step-by-Step	222
Algorithms Are Template Functions	182	Discussion	222
The Algorithm Categories	183	Example	223
Function Object Overview	184	Options and Alternatives	224
Binders and Negators Overview	188	Copy a Sequence from One Container to Another	225
Sort a Container	189	Step-by-Step	225
Step-by-Step	189	Discussion	225
Discussion	189	Example	226
Example	190	Options and Alternatives	227
Options and Alternatives	191	Replace and Remove Elements in a Container	227
Find an Element in a Container	192	Step-by-Step	228
Step-by-Step	193	Discussion	228
Discussion	193	Example	228
		Options and Alternatives	230

Merge Two Sorted Sequences	231
Step-by-Step	231
Discussion	231
Example	232
Options and Alternatives	234
Create and Manage a Heap	235
Step-by-Step	235
Discussion	235
Example	236
Options and Alternatives	238
Create an Algorithm	238
Step-by-Step	238
Discussion	239
Example	240
Bonus Example: Use a Predicate with a Custom Algorithm	242
Options and Alternatives	244
Use a Built-In Function Object	245
Step-by-Step	245
Discussion	246
Example	246
Options and Alternatives	248
Create a Custom Function Object	248
Step-by-Step	249
Discussion	249
Example	250
Bonus Example: Use a Function Object to Maintain State Information	253
Options and Alternatives	255
Use a Binder	255
Step-by-Step	256
Discussion	256
Example	257
Options and Alternatives	258
Use a Negator	259
Step-by-Step	259
Discussion	260
Example	260
Options and Alternatives	261
Use the Pointer-to-Function Adaptor	262
Step-by-Step	262
Discussion	262
Example	263
Options and Alternatives	265

Use the Stream Iterators	265
Step-by-Step	266
Discussion	266
Example	269
Bonus Example: Create an STL-Based File Filter	272
Options and Alternatives	273
Use the Insert Iterator Adaptors	274
Step-by-Step	274
Discussion	275
Example	275
Options and Alternatives	277
5 Working with I/O	279
I/O Overview	280
C++ Streams	280
The C++ Stream Classes	281
The Stream Class Specializations	285
C++'s Predefined Streams	287
The Format Flags	287
The I/O Manipulators	287
Checking for Errors	288
Opening and Closing a File	289
Write Formatted Data to a Text File	293
Step-by-Step	293
Discussion	294
Example	295
Options and Alternatives	296
Read Formatted Data from a Text File	296
Step-by-Step	297
Discussion	297
Example	298
Options and Alternatives	300
Write Unformatted Binary Data to a File	300
Step-by-Step	301
Discussion	301
Example	302
Options and Alternatives	304
Read Unformatted Binary Data from a File	305
Step-by-Step	305
Discussion	306
Example	307
Options and Alternatives	309
Use get() and getline() to Read from a File	310
Step-by-Step	310
Discussion	310

- Example 311
- Options and Alternatives 313
- Read from and Write to a File 314
 - Step-by-Step 314
 - Discussion 315
 - Example 316
 - Options and Alternatives 317
- Detecting EOF 317
 - Step-by-Step 318
 - Discussion 318
 - Example 318
 - Bonus Example: A Simple File-Comparison Utility 320
 - Options and Alternatives 322
- Use Exceptions to Detect and Handle I/O Errors 322
 - Step-by-Step 323
 - Discussion 323
 - Example 324
 - Options and Alternatives 326
- Use Random-Access File I/O 326
 - Step-by-Step 327
 - Discussion 327
 - Example 328
 - Bonus Example: Use Random-Access I/O to
 - Access Fixed-Size Records 329
 - Options and Alternatives 332
- Look Ahead in a File 332
 - Step-by-Step 333
 - Discussion 333
 - Example 334
 - Options and Alternatives 336
- Use the String Streams 337
 - Step-by-Step 337
 - Discussion 338
 - Example 338
 - Options and Alternatives 340
- Create Custom Inserters and Extractors 341
 - Step-by-Step 341
 - Discussion 342
 - Example 343
 - Options and Alternatives 344
- Create a Parameterless Manipulator 344
 - Step-by-Step 345
 - Discussion 345
 - Example 346
 - Options and Alternatives 347

- Create a Parameterized Manipulator 348
 - Step-by-Step 348
 - Discussion 349
 - Example 350
 - Options and Alternatives 352
- Obtain or Set a Stream's Locale 352
 - Step-by-Step 353
 - Discussion 353
 - Example 353
 - Options and Alternatives 355
- Use the C-Based File System 355
 - Step-by-Step 356
 - Discussion 356
 - Example 359
 - Options and Alternatives 361
- Rename and Remove a File 363
 - Step-by-Step 363
 - Discussion 363
 - Example 364
 - Options and Alternatives 365
- 6 Formatting Data 367**
 - Formatting Overview 368
 - The Format Flags 368
 - The Field Width, Precision, and Fill Character 369
 - Format-Related Stream Member Functions 370
 - The I/O Manipulators 370
 - Format Data Using the Localization Library 370
 - The printf() Family of Functions 371
 - The strftime() Function 371
 - Facet Overview 372
 - Access the Format Flags via Stream Member Functions 374
 - Step-by-Step 374
 - Discussion 374
 - Example 375
 - Bonus Example: Display the Format Flag Settings 376
 - Options and Alternatives 378
 - Display Numeric Values in Various Formats 379
 - Step-by-Step 379
 - Discussion 380
 - Example 380
 - Options and Alternatives 382
 - Set the Precision 383
 - Step-by-Step 383
 - Discussion 383

Example	384
Options and Alternatives	384
Set the Field Width and Fill Character	385
Step-by-Step	385
Discussion	385
Example	386
Bonus Example: Line Up Columns of Numbers	387
Options and Alternatives	388
Justify Output	388
Step-by-Step	388
Discussion	389
Example	389
Options and Alternatives	391
Use I/O Manipulators to Format Data	391
Step-by-Step	392
Discussion	392
Example	394
Options and Alternatives	395
Format Numeric Values for a Locale	395
Step-by-Step	396
Discussion	396
Example	396
Options and Alternatives	397
Format Monetary Values Using the money_put Facet	398
Step-by-Step	399
Discussion	399
Example	400
Options and Alternatives	401
Use the moneypunct and numpunct Facets	402
Step-by-Step	402
Discussion	403
Example	404
Options and Alternatives	405
Format Time and Date Using the time_put Facet	407
Step-by-Step	408
Discussion	408
Example	410
Options and Alternatives	411
Format Data into a String	412
Step-by-Step	412
Discussion	412
Example	412
Options and Alternatives	414
Format Time and Date Using strftime()	414
Step-by-Step	414
Discussion	415

Example	415
Options and Alternatives	417
Use printf() to Format Data	418
Step-by-Step	419
Discussion	419
Example	422
Options and Alternatives	424
7 Potpourri	425
Operator Overloading Basic Techniques	426
Step-by-Step	426
Discussion	427
Example	432
Options and Alternatives	435
Overload the Function Call Operator ()	437
Step-by-Step	437
Discussion	437
Example	439
Options and Alternatives	440
Overload the Subscripting Operator []	441
Step-by-Step	441
Discussion	441
Example	442
Options and Alternatives	445
Overload the -> Operator	445
Step-by-Step	446
Discussion	446
Example	446
Bonus Example: A Simple Safe Pointer Class	447
Options and Alternatives	451
Overload new and delete	451
Step-by-Step	451
Discussion	452
Example	453
Options and Alternatives	456
Overload the Increment and Decrement Operators	457
Step-by-Step	457
Discussion	457
Example	459
Options and Alternatives	462
Create a Conversion Function	463
Step-by-Step	463
Discussion	463
Example	464
Options and Alternatives	466

Create a Copy Constructor	466
Step-by-Step	467
Discussion	467
Example	468
Bonus Example: A Safe Array that Uses Dynamic Allocation	471
Options and Alternatives	477
Determine an Object's Type at Runtime	478
Step-by-Step	479
Discussion	479
Example	480
Options and Alternatives	484
Use Complex Numbers	484
Step-by-Step	485
Discussion	485
Example	486
Options and Alternatives	487
Use auto_ptr	487
Step-by-Step	488
Discussion	488
Example	489
Options and Alternatives	490
Create an Explicit Constructor	491
Step-by-Step	491
Discussion	491
Example	492
Options and Alternatives	494
Index	495

Preface

Over the years, friends and readers have asked me to write a programming cookbook, sharing some of the techniques and approaches that I use when I program. From the start, I liked the idea, but was unable to make time for it in my very busy writing schedule. As many readers know, I write extensively about several facets of programming, with a special focus on C++, Java, and C#. Because of the rapid revision cycles of those languages, I spend nearly all of my available time updating my books to cover the latest versions. Fortunately, early in 2007, a window of opportunity opened and I was finally able to devote time to the cookbook. The two most requested cookbooks were ones for Java and C++. I began with Java, with the result being my Java programming cookbook. As soon as I finished the Java book, I moved on to C++. The result is, of course, this book. I must admit that both projects were among my most enjoyable.

Based on the format of a traditional food cookbook, this book distills the essence of many general-purpose C++ techniques into a collection of step-by-step *recipes*. Each recipe describes a set of key ingredients, such as classes, functions, and headers. It then shows the steps needed to assemble those ingredients into a code sequence that achieves the desired result. This organization makes it easy to find the technique in which you are interested and then put that technique *into action*.

Actually, "into action" is an important part of this book. I believe that good programming books contain two elements: solid theory and practical application. In the recipes, the step-by-step instructions and discussions supply the theory. To put that theory into practice, each recipe includes a complete code example. The examples demonstrate in a concrete, unambiguous way how the recipes can be applied. In other words, the examples eliminate the "guess work" and save you time.

Although no cookbook can include every recipe that one might desire (there is a nearly unbounded number of possible recipes), I tried to span a wide range of topics. My criteria for including a recipe are discussed in detail in Chapter 1, but briefly, I included recipes that would be useful to many programmers and that answered frequently asked questions. Even with these criteria, it was difficult to decide what to include and what to leave out. This was the most challenging part of writing this book. Ultimately, it came down to experience, judgment, and intuition. Hopefully, I have included something to satisfy every programmer's taste!

HS