

# **THINK LIKE A PROGRAMMER**

**An Introduction to  
Creative Problem Solving**

**by V. Anton Spraul**



San Francisco

**THINK LIKE A PROGRAMMER.** Copyright © 2012 by V. Anton Spraul.

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

First printing

15 14 13 12    1 2 3 4 5 6 7 8 9

ISBN-10: 1-59327-424-6

ISBN-13: 978-1-59327-424-5

Publisher: William Pollock

Production Editor: Alison Law

Cover Design: Charlie Wylie

Interior Design: Octopod Studios

Developmental Editor: Keith Fancher

Technical Reviewer: Dan Randall

Copyeditor: Julianne Jigour

Compositor: Susan Glinert Stevens

Proofreader: Ward Webber

For information on book distributors or translations, please contact No Starch Press, Inc. directly:

No Starch Press, Inc.

38 Ringold Street, San Francisco, CA 94103

phone: 415.863.9900; fax: 415.863.9950; [info@nostarch.com](mailto:info@nostarch.com); [www.nostarch.com](http://www.nostarch.com)

*Library of Congress Cataloging-in-Publication Data*

A catalog record of this book is available from the Library of Congress.

No Starch Press and the No Starch Press logo are registered trademarks of No Starch Press, Inc. Other product and company names mentioned herein may be the trademarks of their respective owners. Rather than use a trademark symbol with every occurrence of a trademarked name, we are using the names only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The information in this book is distributed on an “As Is” basis, without warranty. While every precaution has been taken in the preparation of this work, neither the author nor No Starch Press, Inc. shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in it.



## BRIEF CONTENTS

Acknowledgments .....	xi
Introduction .....	xiii
Chapter 1: Strategies for Problem Solving .....	1
Chapter 2: Pure Puzzles .....	25
Chapter 3: Solving Problems with Arrays .....	55
Chapter 4: Solving Problems with Pointers and Dynamic Memory .....	81
Chapter 5: Solving Problems with Classes .....	111
Chapter 6: Solving Problems with Recursion .....	143
Chapter 7: Solving Problems with Code Reuse.....	171
Chapter 8: Thinking Like a Programmer .....	195
Index .....	227

# CONTENTS IN DETAIL

<b>ACKNOWLEDGMENTS</b>	<b>xi</b>
<b>INTRODUCTION</b>	<b>xiii</b>
About This Book .....	xv
Prerequisites .....	xv
Chosen Topics .....	xv
Programming Style .....	xvi
Exercises .....	xvi
Why C++? .....	xvii
<b>1 STRATEGIES FOR PROBLEM SOLVING</b>	<b>1</b>
Classic Puzzles .....	2
The Fox, the Goose, and the Corn .....	3
<b>Problem: How to Cross the River?</b> .....	3
Sliding Tile Puzzles .....	7
<b>Problem: The Sliding Eight</b> .....	7
<b>Problem: The Sliding Five</b> .....	8
Sudoku .....	11
<b>Problem: Completing a Sudoku Square</b> .....	11
The Quarrasi Lock .....	13
<b>Problem: Opening the Alien Lock</b> .....	13
General Problem-Solving Techniques .....	15
Always Have a Plan .....	16
Restate the Problem .....	17
Divide the Problem .....	17
Start with What You Know .....	18
Reduce the Problem .....	19
Look for Analogies .....	20
Experiment .....	20
Don't Get Frustrated .....	21
Exercises .....	22
<b>2 PURE PUZZLES</b>	<b>25</b>
Review of C++ Used in This Chapter .....	26
Output Patterns .....	26
<b>Problem: Half of a Square</b> .....	26
<b>Problem: A Square (Half of a Square Reduction)</b> .....	27
<b>Problem: A Line (Half of a Square Further Reduction)</b> .....	27
<b>Problem: Count Down by Counting Up</b> .....	28
<b>Problem: A Sideways Triangle</b> .....	29
Input Processing .....	31
<b>Problem: Luhn Checksum Validation</b> .....	31
Breaking Down the Problem .....	33

<b>Problem: Convert Character Digit to Integer</b> .....	35
<b>Problem: Luhn Checksum Validation, Fixed Length</b> .....	36
<b>Problem: Simple Checksum Validation, Fixed Length</b> .....	36
<b>Problem: Positive or Negative</b> .....	39
Putting the Pieces Together .....	39
<b>Tracking State</b> .....	41
<b>Problem: Decode a Message</b> .....	41
<b>Problem: Reading a Number with Three or Four Digits</b> .....	45
<b>Problem: Reading a Number with Three or Four Digits, Further Simplified</b> .....	46
<b>Conclusion</b> .....	53
<b>Exercises</b> .....	53

## 3 SOLVING PROBLEMS WITH ARRAYS 55

<b>Review of Array Fundamentals</b> .....	56
Store .....	56
Copy .....	57
Retrieval and Search .....	57
Sort .....	59
Compute Statistics .....	61
<b>Solving Problems with Arrays</b> .....	62
<b>Problem: Finding the Mode</b> .....	62
Refactoring .....	65
<b>Arrays of Fixed Data</b> .....	67
<b>Non-scalar Arrays</b> .....	69
<b>Multidimensional Arrays</b> .....	71
<b>Deciding When to Use Arrays</b> .....	74
<b>Exercises</b> .....	78

## 4 SOLVING PROBLEMS WITH POINTERS AND DYNAMIC MEMORY 81

<b>Review of Pointer Fundamentals</b> .....	82
<b>Benefits of Pointers</b> .....	83
Runtime-Sized Data Structures .....	83
Resizable Data Structures .....	83
Memory Sharing .....	84
<b>When to Use Pointers</b> .....	84
<b>Memory Matters</b> .....	85
The Stack and the Heap .....	86
Memory Size .....	88
Lifetime .....	90
<b>Solving Pointer Problems</b> .....	91
Variable-Length Strings .....	91
<b>Problem: Variable-Length String Manipulation</b> .....	91
Linked Lists .....	101
<b>Problem: Tracking an Unknown Quantity of Student Records</b> .....	101
<b>Conclusion and Next Steps</b> .....	108
<b>Exercises</b> .....	109

## 5

# SOLVING PROBLEMS WITH CLASSES

111

Review of Class Fundamentals .....	112
Goals of Class Use .....	113
Encapsulation .....	114
Code Reuse .....	114
Dividing the Problem .....	115
Information Hiding .....	115
Readability .....	117
Expressiveness .....	117
Building a Simple Class .....	118
<b>Problem: Class Roster</b> .....	<b>118</b>
The Basic Class Framework .....	119
Support Methods .....	122
Classes with Dynamic Data .....	125
<b>Problem: Tracking an Unknown Quantity of Student Records</b> .....	<b>126</b>
Adding a Node .....	128
Rearranging the List .....	130
Destructor .....	133
Deep Copy .....	134
The Big Picture for Classes with Dynamic Memory .....	139
Mistakes to Avoid .....	140
The Fake Class .....	140
Single-Taskers .....	141
Exercises .....	141

## 6

# SOLVING PROBLEMS WITH RECURSION

143

Review of Recursion Fundamentals .....	144
Head and Tail Recursion .....	144
<b>Problem: How Many Parrots?</b> .....	<b>144</b>
Approach 1 .....	145
Approach 2 .....	146
<b>Problem: Who's Our Best Customer?</b> .....	<b>148</b>
Approach 1 .....	149
Approach 2 .....	151
The Big Recursive Idea .....	152
<b>Problem: Computing the Sum of an Array of Integers</b> .....	<b>153</b>
Common Mistakes .....	155
Too Many Parameters .....	155
Global Variables .....	156
Applying Recursion to Dynamic Data Structures .....	158
Recursion and Linked Lists .....	158
<b>Problem: Counting Negative Numbers in a Singly Linked List</b> .....	<b>159</b>
Recursion and Binary Trees .....	160
<b>Problem: Find the Largest Value in a Binary Tree</b> .....	<b>162</b>
Wrapper Functions .....	163
<b>Problem: Find the Number of Leaves in a Binary Tree</b> .....	<b>163</b>
When to Choose Recursion .....	165
Arguments Against Recursion .....	166

<b>Problem: Display a Linked List in Order .....</b>	<b>168</b>
<b>Problem: Display a Linked List in Reverse Order .....</b>	<b>168</b>
<b>Exercises .....</b>	<b>170</b>

## 7 SOLVING PROBLEMS WITH CODE REUSE 171

Good Reuse and Bad Reuse .....	172
Review of Component Fundamentals .....	173
Code Block .....	173
Algorithms .....	173
Patterns .....	174
Abstract Data Types .....	175
Libraries .....	175
Building Component Knowledge .....	176
Exploratory Learning .....	176
<b>Problem: The First Student .....</b>	<b>177</b>
As-Needed Learning .....	180
<b>Problem: Efficient Traversal .....</b>	<b>180</b>
Choosing a Component Type .....	188
Component Choice in Action .....	189
<b>Problem: Sorting Some, Leaving Others Alone .....</b>	<b>189</b>
Comparing the Results .....	193
<b>Exercises .....</b>	<b>193</b>

## 8 THINKING LIKE A PROGRAMMER 195

Creating Your Own Master Plan .....	196
Playing to Your Strengths and Weaknesses .....	196
Putting the Master Plan Together .....	202
Tackling Any Problem .....	203
<b>Problem: Cheating at Hangman .....</b>	<b>204</b>
Finding a Way to Cheat .....	205
Required Operations for Cheating at Hangman .....	206
Initial Design .....	208
Initial Coding .....	210
Analysis of Initial Results .....	217
The Art of Problem Solving .....	218
Learning New Programming Skills .....	219
New Languages .....	219
New Skills for a Language You Already Know .....	222
New Libraries .....	223
Take a Class .....	223
Conclusion .....	224
<b>Exercises .....</b>	<b>225</b>

## INDEX 227

## **ACKNOWLEDGMENTS**

No book is truly the work of one author, and I've received lots of help on *Think Like a Programmer*.

I'm grateful to everyone at No Starch Press, especially Keith Fancher and Alison Law, who edited, shaped, and shepherded the book throughout its production. I must also thank Bill Pollock for his decision to sign me up in the first place—I hope he is as pleased with the result as I am. The folks at No Starch have been unfailingly kind and helpful in their correspondence with me. I hope one day to meet them in person and see to what degree they resemble their cartoon avatars on the company website.

Dan Randall did a wonderful job as technical editor. His numerous suggestions beyond the technical review helped me strengthen the manuscript in many areas.

On the home front, the most important people in my life, Mary Beth and Madeline, provided love, support, and enthusiasm—and, crucially, time to write.

Finally, to all the students of programming I've had over the years: Thank you for letting me be your teacher. The techniques and strategies described in this book were developed through our joint efforts. I hope we've made the journey easier for the next generation of programmers.