



Thinking in C++ 2nd edition

Volume 2: Standard Libraries & Advanced Topics

To be informed of future releases of this document and other information about object-oriented books, documents, seminars and CDs, subscribe to my free newsletter. Just send any email to: join-eckel-oo-programming@earth.lyris.net

Thinking in C++ 2nd edition Volume 2: Standard Libraries & Advanced Topics

To be informed of future releases of this document and other information about object- oriented books, documents, seminars and CDs, subscribe to my free newsletter. Just send any email to:
join-eckel-oo-programming@earth.lyris.net

— | | —

“This book is a tremendous achievement. You owe it to yourself to have a copy on your shelf. The chapter on iostreams is the most comprehensive and understandable treatment of that subject I’ve seen to date.”

Al Stevens
Contributing Editor, Doctor Dobbs Journal

“Eckel’s book is the only one to so clearly explain how to rethink program construction for object orientation. That the book is also an excellent tutorial on the ins and outs of C++ is an added bonus.”

Andrew Binstock
Editor, Unix Review

“Bruce continues to amaze me with his insight into C++, and *Thinking in C++* is his best collection of ideas yet. If you want clear answers to difficult questions about C++, buy this outstanding book.”

Gary Entsminger
Author, *The Tao of Objects*

“*Thinking in C++* patiently and methodically explores the issues of when and how to use inlines, references, operator overloading, inheritance and dynamic objects, as well as advanced topics such as the proper use of templates, exceptions and multiple inheritance. The entire effort is woven in a fabric that includes Eckel’s own philosophy of object and program design. A must for every C++ developer’s bookshelf, *Thinking in C++* is the one C++ book you must have if you’re doing serious development with C++.”

Richard Hale Shaw
Contributing Editor, PC Magazine

— | | —

“This book is a tremendous achievement. You owe it to yourself to have a copy on your shelf.”

The chapter on iostreams is the most comprehensive and understandable treatment of that subject I've seen to date."

Al Stevens Contributing Editor, Doctor

Dobbs Journal

"Eckel's book is the only one to so clearly explain how to rethink program construction for object orientation. That the book is also an excellent tutorial on the ins and outs of C++ is an added bonus."

Andrew Binstock Editor, Unix Review

"Bruce continues to amaze me with his insight into C++, and Thinking in C++ is his best collection of ideas yet. If you want clear answers to difficult questions about C++, buy this outstanding book."

Gary Entsminger Author, The Tao of Objects

"Thinking in C++ patiently and methodically explores the issues of when and how to use inlines, references, operator overloading, inheritance and dynamic objects, as well as advanced topics such as the proper use of templates, exceptions and multiple inheritance. The entire effort is woven in a fabric that includes Eckel's own philosophy of object and program design. A must for every C++ developer's bookshelf, Thinking in C++ is the one C++ book you must have if you're doing serious development with C++."

Richard Hale Shaw Contributing Editor, PC Magazine

Thinking In C++

2nd Edition, Volume 2

Bruce Eckel
President, MindView Inc.



Thinking In C++

2

nd

Edition, Volume 2

Bruce Eckel President, MindView Inc.



© 1999 by Bruce Eckel, MindView, Inc.

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 book, neither the author nor the publisher shall have any liability to any person or entitle with respect to any liability, loss or damage caused or alleged to be caused directly or indirectly by instructions contained in this book or by the computer software or hardware products described herein.

All rights reserved. No part of this book may be reproduced in any form or by any electronic or mechanical means including information storage and retrieval systems without permission in writing from the publisher or author, except by a reviewer who may quote brief passages in a review. Any of the names used in the examples and text of this book are fictional; any relationship to persons living or dead or to fictional characters in other works is purely coincidental.

© 1999 by Bruce Eckel, MindView, Inc.

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 book, neither the author nor the publisher shall have any liability to any person or entitle with respect to any liability, loss or damage caused or alleged to be caused directly or indirectly by instructions contained in this book or by the computer software or hardware products described herein.

All rights reserved. No part of this book may be reproduced in any form or by any electronic or mechanical means including information storage and retrieval systems without permission in writing from the publisher or author, except by a reviewer who may quote brief passages in a review. Any of the names used in the examples and text of this book are fictional; any relationship to persons living or dead or to fictional characters in other works is purely coincidental.



dedication

To the scholar, the healer, and the muse



dedication

To the scholar, the healer, and the muse

What's inside...

Thinking in C++ 2nd edition Volume 2: Standard Libraries & Advanced Topics Revision 1, xx 1999
.....1

Preface	13	
What's new in the second edition	13	
What's in Volume 2 of this book	14	
How to get Volume 2	14	
Prerequisites	14	
Learning C++	14	
Goals	16	
Chapters	17	
Exercises	18	
Exercise solutions	18	
Source code	18	
Language standards	20	
Language support	20	
The book's CD ROM	20	
Seminars, CD Roms & consulting	20	
Errors	21	
Acknowledgements	21	
Part 1: The Standard C++ Library		23
Library overview	24	
1: Strings	27	
What's in a string	27	
Creating and initializing C++ strings	29	
Operating on strings	31	
Appending, inserting and concatenating strings	32	
Replacing string characters	34	
Concatenation using non-member overloaded operators	37	
Searching in strings	38	
Finding in reverse	43	
Finding first/last of a set	44	
Removing characters from strings	45	
Comparing strings	49	
Using iterators	53	

What's inside...

Thinking in C++ 2nd edition Volume 2: Standard Libraries & Advanced Topics Revision 1, xx 1999

.....1 Preface 13

What's new in the second edition	13	What's in Volume 2 of this book	14	How to get Volume 2	14
Prerequisites.....	14	Learning C++.....	14	Goals	16
Chapters	17	Exercises	18	Exercise solutions	18
Source code.....	18	Language standards.....	20	Language support.....	20
book's CD ROM	20	Seminars, CD Roms & consulting	20	Errors	21
Acknowledgements.....	21	Part 1: The Standard C++ Library	23		
Library overview.....	24	1: Strings	27		
What's in a string	27	Creating and initializing C++ strings	29	Operating on strings.....	31
Appending, inserting and concatenating strings	32	Replacing string characters	34	Concatenation using non-member	
overloaded operators	37	Searching in strings.....	38	Finding in reverse.....	43
set.....	44	Removing characters from strings.....	45	Comparing strings.....	49
Using		iterators.....	53		

Strings and character traits	55
A string application	58
Summary	61
Exercises	62

2: Iostreams 63

Why iostreams?	63
True wrapping	65
Iostreams to the rescue	67
Sneak preview of operator overloading	68
Inserters and extractors	69
Common usage	70
Line-oriented input	72
File iostreams	74
Open modes	76
Iostream buffering	76
Using get() with a streambuf	78
Seeking in iostreams	78
Creating read/write files	80
stringstreams	81
strstreams	81
User-allocated storage	81
Automatic storage allocation	84
Output stream formatting	87
Internal formatting data	88
An exhaustive example	92
Formatting manipulators	95
Manipulators with arguments	96
Creating manipulators	99
Effectors	100
Iostream examples	102
Code generation	102
A simple datalogger	110
Counting editor	117
Breaking up big files	118
Summary	120
Exercises	120

3: Templates in depth 121

Nontype template arguments ...	121
Default template arguments	122
The typename keyword	122
Typedefing a typename	124
Using typename instead of class	124
Function templates	124
A string conversion system	125
A memory allocation system	126
Type induction in function templates	129
Taking the address of a generated function template	130

	Exercises	62	2: Iostreams	63
Why iostreams?.....	63	True wrapping.....	65	Iostreams to the rescue.....
	67			
Sneak preview of operator overloading.....	68	Inserters and extractors	69	Common usage.....
	70			
Line-oriented input.....	72	File iostreams.....	74	Open modes
	76			
buffering.....	76	Using get() with a streambuf.....	78	Seeking in iostreams
	78			
Creating read/write files	80	stringstreams	81	strstreams
	81			
User-allocated storage.....	81			
Automatic storage allocation.....	84	Output stream formatting.....	87	Internal formatting data.....
	88			
An exhaustive example	92	Formatting manipulators.....	95	Manipulators with arguments.....
	96			
Creating manipulators.....	99	Effectors.....	100	Iostream examples
	102			
Code generation	102	A simple datalogger	110	Counting editor
	117			
Breaking up big files	118	Summary.....	120	Exercises
	120			
3: Templates in depth	121			
Nontype template arguments ...	121	Default template arguments	122	The typename keyword.....
	122			
Typedefing a typename.....	124	Using typename instead of class	124	Function templates
	124			
A string conversion system	125	A memory allocation system.....	126	Type induction in function
templates	129	Taking the address of a generated function template	130	

Chapter 2: Hiding the Implementation 7

Local classes in templates	131
Applying a function to an STL sequence	131
Template-templates	134
Member function templates	135
Why virtual member template functions are disallowed	137
Nested template classes	137
Template specializations	137
Full specialization	137
Partial Specialization	137
A practical example	137
Design & efficiency	141
Preventing template bloat	141
Explicit instantiation	143
Explicit specification of template functions	144
Controlling template instantiation	144
The inclusion vs. separation models	145
The export keyword	145
Template programming idioms	145
The "curiously-recurring template"	145
Traits	145
Summary	145

4: STL Containers & Iterators 147

Containers and iterators	147
STL reference documentation	149
The Standard Template Library	149
The basic concepts	151
Containers of strings	155
Inheriting from STL containers	157
A plethora of iterators	159
Iterators in reversible containers	161
Iterator categories	162
Predefined iterators	163
Basic sequences: vector, list & deque	169
Basic sequence operations	169
vector	172
Cost of overflowing allocated storage	173
Inserting and erasing elements	177
deque	179
Converting between sequences	181
Cost of overflowing allocated storage	182
Checked random-access	184
list	185
Special list operations	187
Swapping all basic sequences	191
Robustness of lists	192
Performance comparison	193
set	198
Eliminating strtok()	199
StreamTokenizer: a more flexible solution	201

134 Member function templates 135

Why virtual member template functions are disallowed 137 Nested template classes.....137 Template specializations
..... 137 Full specialization137 Partial Specialization.....137 A practical example
.....137 Design & efficiency141 Preventing template bloat.....141 Explicit instantiation
..... 143

Explicit specification of template functions 144 Controlling template instantiation144

The inclusion vs. separation models145 The export keyword145 Template programming idioms 145 The
“curiously-recurring template”.145 Traits.....145 Summary..... 145 4: STL Containers
& Iterators147

Containers and iterators 147 STL reference documentation149 The Standard Template Library 149

The basic concepts 151 Containers of strings 155 Inheriting from STL containers 157 A
plethora of iterators 159 Iterators in reversible containers161 Iterator categories.....162
Predefined iterators163 Basic sequences: vector, list & deque 169

Basic sequence operations.....169 vector 172

Cost of overflowing allocated storage173 Inserting and erasing elements177 deque..... 179 Converting
between sequences181 Cost of overflowing allocated storage182 Checked random-access184 list
..... 185 Special list operations187 Swapping all basic sequences.....191 Robustness of
lists.....192 Performance comparison 193 set.....198 Eliminating strtok(
).....199 StreamTokenizer: a more flexible solution 201

Chapter 2: Hiding the Implementation 8

A completely reusable tokenizer	203
stack	208
queue	211
Priority queues	216
Holding bits	226
bitset<cn>	226
vector<bool>	230
Associative containers	232
Generators and fillers for associative containers	236
The magic of maps	239
Multimaps and duplicate keys	244
Multisets	247
Combining STL containers	250
Cleaning up containers of pointers	253
Creating your own containers ..	255
Freely-available STL extensions ..	257
Summary	259
Exercises	260

5: STL Algorithms 263

Function objects	263
Classification of function objects	264
Automatic creation of function objects	265
SGI extensions	279
A catalog of STL algorithms	285
Support tools for example creation	287
Filling & generating	291
Counting	293
Manipulating sequences	294
Searching & replacing	299
Comparing ranges	305
Removing elements	308
Sorting and operations on sorted ranges	311
Heap operations	322
Applying an operation to each element in a range	323
Numeric algorithms	331
General utilities	334
Creating your own STL-style algorithms	336
Summary	337
Exercises	337

Part 2: Advanced Topics 341

6: Multiple inheritance 342

Perspective	342
Duplicate subobjects	344
Ambiguous upcasting	345
virtual base classes	346

Priority queues	216	Holding bits.....	226	bitset<n>	226
		vector<bool>.....	230	Associative containers	232
Generators and fillers for associative containers	236	The magic of maps.....	239	Multimaps and duplicate keys.....	244
Multisets	247	Combining STL containers	250	Cleaning up containers of pointers	253
Creating your own containers ..	255	Freely-available STL extensions	257	Summary.....	259
Exercises	260	5: STL Algorithms	263	Function objects.....	263
Classification of function objects	264	Automatic creation of function objects	265	SGI extensions	279
A catalog of STL algorithms....	285	Support tools for example creation..	287	Filling & generating.....	291
Counting	293	Manipulating sequences	294	Searching & replacing.....	299
Comparing ranges	305	Removing elements.....	308	Sorting and operations on sorted ranges	311
Heap operations	322	Applying an operation to each element in a range	323	Numeric algorithms.....	331
General utilities.....	334	Creating your own STL-style algorithms	336	Summary.....	337
Exercises	337	Part 2: Advanced Topics	341		

6: Multiple inheritance 342

Perspective	342	Duplicate subobjects	344	Ambiguous upcasting.....	345
virtual base classes.....	346				

Chapter 2: Hiding the Implementation 9

The "most derived" class and virtual base initialization	348
"Tying off" virtual bases with a default constructor	349
Overhead	351
Upcasting	352
Persistence	355
Avoiding MI	362
Repairing an interface	362
Summary	367
Exercises	368

7: Exception handling 369

Error handling in C	369
Throwing an exception	372
Catching an exception	373
The try block	373
Exception handlers	373
The exception specification	374
Better exception specifications?	377
Catching any exception	377
Rethrowing an exception	378
Uncaught exceptions	378
Function-level try blocks	380
Cleaning up	380
Constructors	384
Making everything an object	386
Exception matching	388
Standard exceptions	390
Programming with exceptions	391
When to avoid exceptions	391
Typical uses of exceptions	392
Overhead	396
Summary	397
Exercises	397

8: Run-time type identification 399

The "Shape" example	399
What is RTTI?	400
Two syntaxes for RTTI	400
Syntax specifics	404
typeid() with built-in types	404
Producing the proper type name	405
Nonpolymorphic types	405
Casting to intermediate levels	406
void pointers	408
Using RTTI with templates	408
References	409
Exceptions	410
Multiple inheritance	411

Overhead	351	Upcasting	352	Persistence	355
Avoiding MI.....	362	Repairing an interface	362	Summary.....	367
Exercises	368	7: Exception handling	369		
Error handling in C	369	Throwing an exception	372	Catching an exception.....	373
.....	373	Exception handlers.....	373	The exception specification.....	374
Better exception specifications?.....	377	Catching any exception	377	Rethrowing an exception.....	378
Uncaught exceptions	378	Function-level try blocks.....	380	Cleaning up	380
Constructors	380	Making everything an object.....	386	Exception matching	388
Standard exceptions	390	Programming with exceptions .	391	When to avoid exceptions	391
Typical uses of exceptions	392	Overhead	396	Summary.....	397
Exercises	397	8: Run-time type identification	399		
The “Shape” example	399	What is RTTI?.....	400	Two syntaxes for RTTI	400
Syntax specifics	404	typeid() with built-in types	404	Producing the proper type name.....	405
Nonpolymorphic types	405	Casting to intermediate levels	406	void pointers	408
Using RTTI with templates	408	References.....	409	Exceptions.....	410
Multiple inheritance	411				

Chapter 2: Hiding the Implementation 10