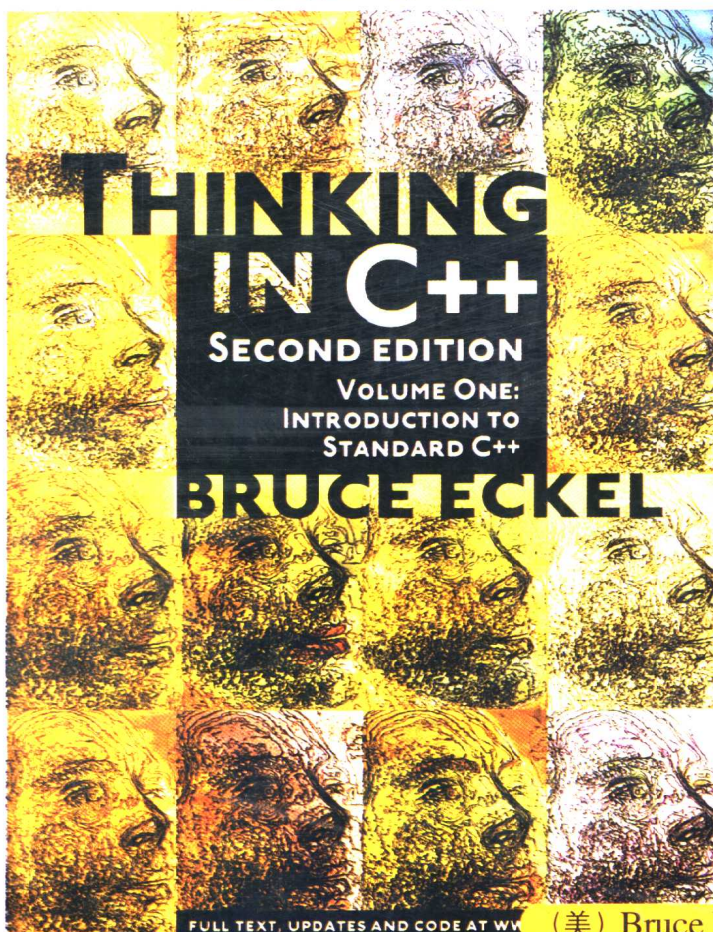


经 典 原 版 书 库

C++ 编程思想

(英文版·第2版)



FULL TEXT, UPDATES AND CODE AT [WWW.PRENTICEHALL.COM](http://www.prenticehall.com)

(美) Bruce Eckel 著



机械工业出版社
China Machine Press

Prentice Hall

经 典 原

(英文版·第2版)

C++编程思想

Thinking in C++
(Second Edition)

(美) Bruce Eckel 著



机械工业出版社
China Machine Press

English reprint edition copyright © 2002 by PEARSON
EDUCATION ASIA LIMITED and CHINA MACHINE PRESS.

Thinking In C++, Second Edition by Bruce Eckel, Copyright ©
2000. All rights reserved. Published by arrangement with Pearson
Education, Inc.

本书英文影印版由美国Prentice Hall公司授权机械工业出版社在
中国大陆境内独家出版发行, 未经出版者许可, 不得以任何方式抄袭、
复制或节录本书中的任何部分。

版权所有, 侵权必究。

本书版权登记号: 图字: 01-2001-3758

图书在版编目 (CIP) 数据

C++编程思想 (英文版·第2版) / (美) 埃克尔 (Eckel, B.) 著
—北京: 机械工业出版社, 2002.1

(经典原版书库)

ISBN 7-111-09162-0

I. C… II. 埃… III. C语言—程序设计—英文 IV. TP312

中国版本图书馆CIP数据核字 (2001) 第051439号

机械工业出版社 (北京市西城区百万庄大街22号 邮政编码 100037)

责任编辑: 华 章

北京昌平奔腾印刷厂印刷·新华书店北京发行所发行

2002年1月第1版·2002年3月第3次印刷

880mm × 1230mm 1/32 · 25.875印张

印数: 5 001-8 000册

定价: 58.00元(附光盘)

凡购本书, 如有倒页、脱页、缺页, 由本社发行部调换

出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域中取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭开了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短、从业人员较少的现状下，美国等发达国家在其计算机科学发展的几十年间积淀的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章图文信息有限公司较早意识到“出版要为教育服务”。自1998年始，华章公司就将工作重点放在了遴选、移译国外优秀教材上。经过几年的不懈努力，我们与Prentice Hall, Addison-Wesley, McGraw-Hill, Morgan Kaufmann等世界著名出版公司建立了良好的合作关系，从它们现有的数百种教材中甄选出Tanenbaum, Stroustrup, Kernighan, Jim Gray等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及收藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专诚为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍，为进一步推广与发展打下了坚实的基础。

11/4/38/05

随着学科建设的初步完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都步入一个新的阶段。为此，华章公司将加大引进教材的力度，在“华章教育”的总规划之下出版三个系列的计算机教材：针对本科生的核心课程，剔抉外版菁华而成“国外经典教材”系列；对影印版的教材，则单独开辟出“经典原版书库”；定位在高级教程和专业参考的“计算机科学丛书”还将保持原来的风格，继续出版新的品种。为了保证这三套丛书的权威性，同时也为了更好地为学校和老师服务，华章公司聘请了中国科学院、北京大学、清华大学、国防科技大学、复旦大学、上海交通大学、南京大学、浙江大学、中国科技大学、哈尔滨工业大学、西安交通大学、中国人民大学、北京航空航天大学、北京邮电大学、中山大学、解放军理工大学、郑州大学、湖北工学院、中国国家信息安全测评认证中心等国内重点大学和科研机构在计算机的各个领域的著名学者组成“专家指导委员会”，为我们提供选题意见和出版监督。

“经典原版书库”是响应教育部提出的使用原版国外教材的号召，为国内高校的计算机教学度身订造的。在广泛地征求并听取丛书的“专家指导委员会”的意见后，我们最终选定了这30多种篇幅内容适度、讲解鞭辟入里的教材，其中的大部分已经被M.I.T.、Stanford、U.C. Berkley、C.M.U.等世界名牌大学采用。丛书不仅涵盖了程序设计、数据结构、操作系统、计算机体系结构、数据库、编译原理、软件工程、图形学、通信与网络、离散数学等国内大学计算机专业普遍开设的核心课程，而且各具特色——有的出自语言设计者之手、有的历三十年而不衰、有的已被全世界的几百所高校采用。在这些圆熟通博的名师大作的指引之下，读者必将在计算机科学的宫殿中由登堂而入室。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证，但我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。教材的出版只是我们的后续服务的起点。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方式如下：

电子邮件：hzedu@hzbook.com

联系电话：(010) 68995265

联系地址：北京市西城区百万庄南街1号

邮政编码：100037

专家指导委员会

(按姓氏笔画顺序)

尤晋元	王 珊	冯博琴	史忠植	史美林
石教英	吕 建	孙玉芳	吴世忠	吴时霖
张立昂	李伟琴	李师贤	李建中	杨冬青
邵维忠	陆丽娜	陆鑫达	陈向群	周伯生
周克定	周傲英	孟小峰	岳丽华	范 明
郑国梁	施伯乐	钟玉琢	唐世渭	袁崇义
高传善	梅 宏	程 旭	程时端	谢希仁
裘宗燕	戴 葵			

Winner, Software Development Magazine's 1996 Jolt Award for Best Book of the Year

"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

Comments from Readers:

Wonderful book ... Great stuff! Andrew Schulman, Doctor Dobbs Journal

An absolute, unqualified must. One of the most-used, most trusted books on my shelf." TUG Lines

This is stuff a programmer can really use. IEEE Computer

A refreshing departure. PJ Plauger, Embedded Systems Programming magazine

...Eckel succeeds ... it's so readable. Unix World

Should definitely be your first buy. C Gazette

A fantastic reference for C++! Michael Brandt, Senior Analyst/Programmer, Sydney, Australia

On our project at HRB Systems we call your book "The Answer Book". It is our C++ Bible for the project. Curt Snyder, HRB Systems

Your book is really great, and I can't thank you enough for making it available for free on the web. It's one of the most thorough and useful references for C++ I've seen. Russell Davis

... the only book out there that even comes close to being actually readable when trying to learn the ropes of C++ (and the basics of good object oriented programming in general). Gunther Schulz, KawaiiSoft

I love the examples in your book. There's stuff there that I never would have thought of (and some things that I didn't know you could do)! Rich Herrick, Senior Associate Software Engineer, Lockheed-Martin Federal Systems, Owego, NY

It's an amazing book. Any questions I have I refer to this online book. Helped in every case. I'm simply happy to have access to a book of this-caliber. Wes Kells, Comp Eng. Student, SLC Kingston.

You are an invaluable resource and I greatly appreciate your books, email list etc... It seems every project I have worked on has been successful because of your insights. Justin Voshell

This is the book I have been looking for on C++. Thomas A. Fink, Managing Director, Trepp, LLC

Your books are authoritative yet easy to read. To my colleagues I call you the K&R of C++. Mark Orlassino, Senior Design Engineer, Harmon Industries, Inc., Hauppauge, NY

When I first started learning C++, your book "Thinking in C++" was my shining guide light in a dark tunnel. It has been my endeavor to improve my C++ skills

whenever possible, and to that effect, "Thinking in C++" has given me the strong foundation for my continuous improvement. Peter Tran, Senior Systems Analyst (IM), Compaq Computer Corporation

This book is the best general reference in my on-going quest to master C++. Most books explain some topics thoroughly but are deficient in others. "Thinking in C++" 2/E does not pass the buck to another book. When I have questions it has answers. Thomas Michel

I have a whole mountain of books and none of them make sense nor do they explain things properly. I have been dying for a good template and STL book. Then I decided to read your material and I was amazed. What you did was show how to write C++ with templates and STL without bogging down with details. What you did was what I expected of the C++ community, the next generation of C++ authors. As an author I AM IMPRESSED at your writing and explanation skills. You covered topics that nobody has properly covered before. Your approach is one from a person who has actually sat down and went through the material in detail. And then you questioned the sanity of the situation and what would be the problem areas. On my bookshelf, it will definitely be one of the necessary books, right beside Petzold. Christian Gross, consultant/mentor cgross@eusoft.com

I think your book is very, very, VERY good. I have compared it to others in the bookstore, and have found that your book actually teaches me basic C++ fundamentals while I learn the STL... a very nice experience to learn about both at once, hand-in-hand. I think your book is laid out very well, and explains things in an easy-to-understand fashion. Jeff Meininger, Software Developer, boxybutgood.com

Your book is the best by far of any I've seen. Please get it right so that we can all have an excellent and "reliable" reference work! And please hurry! We are desperate for a work of this quality! Steve Strickland, Live Minds (a Puzzle business)

(On Usenet) Unlike most other C++ authors, Eckel has made a career of teaching C++ and Java classes ONLY. He's had the benefit of a GREAT deal of novice feedback, and the books reflect that. His books are not just about writing in C++/Java, but understanding the intent of the languages and the mindset that goes with thinking in them. Eckel's also the best technical writer I've read since Jeff Duntemann. Very clear and easy to read. Don't be put off by the apparent large size of his books. Either can be read in *less* than 21 days. :-} Randy Crawford, MRJ Technology Solutions, Fairfax VA

Your work is greatly appreciated and I thank you for helping me understand both C++ and Java better. Barry Wallin, Math/Computer Science Teacher, Rosemount High School, Rosemount, MN

I would like to thank you for your book "Thinking in C++" which is, with no doubt, the best book I ever read about this subject. Riccardo Tarli - SW Engineer - R&D TXT Ingegneria Informatica - Italy

I have been reading both of your books, Thinking In Java and Thinking In C++. Each of these books is easily the best in its category. Ratnakarprasad H. Tiwari, Mumbai, India

... the "Debugging Hints" section is so valuable, I'm tempted to print it and keep it with me at all times. I think this section should be a mandatory part of any introductory class after the first one or two programming problems. Fred Ballard, Synectics Inc.

Your book is really a treasure trove of C++ knowledge. I feel like you give a good overview and then explain the nuts and bolts. Raymond Pickles, Antenna Section, Radar Division, U.S. Naval Research Laboratory, Washington DC

As an Internal Medicine Specialist and Computer Scientist I spend a great deal of time trying to extract information from books and journals. My experience is that a good author is one who makes difficult concepts accessible, a great one makes it look almost easy. On this score you are certainly one of my top three technical writers. Keep up the good work. Dr. Declan O'Kane, Leicester, England

For my second-level C++ course, "Thinking in C++" is my constant reference and companion, and I urge my students to consult it regularly. I refer to the chapter on Operator Overloading constantly. The examples/code alone are worth the cost of the book many times over. So many books and development environments are predicated on the assumption that the only application for a programming language is for a Windows environment; it's great to find and use a book which concentrates on C++ so we can prepare our students for careers in fields like embedded systems, networking, etc., which require real depth of understanding. Robert Chase, Professor, Sweet Briar College

I think it's a fantastic intro to C++, especially for longtime dabblers like me - I often know "how," but rarely "why," and TIC2 is a godsend. Tony Likhite, System Administrator/DBA, Together Networks

After reading the first 80 pages of this book, I have a better understanding of oop than I've gotten out of the ton of books I've accumulated on the subject. Thanks... Rick Schneewind

What's inside...

Preface	1	1: Introduction to Objects	21
What's new in the second edition	2	The progress of abstraction.....	22
What's in Volume 2 of this book.....	3	An object has an interface	25
How to get Volume 2.....	3	The hidden implementation	28
Prerequisites	3	Reusing the implementation	29
Learning C++	4	Inheritance: reusing the interface	31
Goals	5	Is-a vs. is-like-a relationships.....	35
Chapters	7	Interchangeable objects with polymorphism	37
Exercises.....	12	Creating and destroying objects... ..	41
Exercise solutions	12	Exception handling: dealing with errors ..	43
Source code.....	12	Analysis and design	44
Language standards	14	Phase 0: Make a plan.....	47
Language support	15	Phase 1: What are we making?	48
The book's CD ROM	15	Phase 2: How will we build it?	52
CD ROMs, seminars, and consulting	16	Phase 3: Build the core	56
Errors	16	Phase 4: Iterate the use cases	57
About the cover.....	17		
Book design and production	18		
Acknowledgements .	19		

Phase 5: Evolution	58
Plans pay off	60

Extreme programming	61
Write tests first	61
Pair programming	63

Why C++ succeeds	64
A better C	65
You're already on the learning curve	65
Efficiency	66
Systems are easier to express and understand	66
Maximal leverage with libraries	67
Source-code reuse with templates	67
Error handling	67
Programming in the large	68

Strategies for transition	68
Guidelines	69
Management obstacles	71
Summary	73

2: Making & Using Objects 75

The process of language translation	76
Interpreters	77
Compilers	77
The compilation process	79

Tools for separate compilation	80
Declarations vs. definitions	81
Linking	87
Using libraries	88

Your first C++ program	90
Using the iostreams class	90
Namespaces	91
Fundamentals of program structure	93
"Hello, world!"	94
Running the compiler	95

More about iostreams	96
Character array concatenation	96
Reading input	97
Calling other programs	98

Introducing strings	98
Reading and writing files	100
Introducing vector	102
Summary	108
Exercises	109

3: The C in C++ 111

Creating functions	112
Function return values	115
Using the C function library	116
Creating your own libraries with the librarian	117

Controlling execution	117
True and false	117
if-else	118
while	119
do-while	120
for	121
The break and continue keywords	122
switch	123
Using and misusing goto	125
Recursion	126

Introduction to operators	127
Precedence	127
Auto increment and decrement	128

Introduction to data types	129
Basic built-in types	129
bool, true, & false	131
Specifiers	132
Introduction to pointers	133

Modifying the outside object	137
Introduction to C++ references	140
Pointers and references as modifiers	141
Scoping	143
Defining variables on the fly	145
Specifying storage allocation	147
Global variables	147
Local variables	149
static	149
extern	151
Constants	153
volatile	155
Operators and their use	156
Assignment	156
Mathematical operators	156
Relational operators	158
Logical operators	158
Bitwise operators	159
Shift operators	160
Unary operators	163
The ternary operator	164
The comma operator	165
Common pitfalls when using operators	166
Casting operators	166
C++ explicit casts	167
sizeof - an operator by itself	172
The asm keyword	173
Explicit operators	173
Composite type creation	174
Aliasing names with typedef	174
Combining variables with struct	175
Clarifying programs with enum	179
Saving memory with union	181
Arrays	182
Debugging hints	193
Debugging flags	194
Turning variables and expressions into strings	196
The C assert() macro	197

Function addresses	198
Defining a function pointer	198
Complicated declarations & definitions	199
Using a function pointer	200
Arrays of pointers to functions	201
Make: managing separate compilation	202
Make activities	204
Makefiles in this book	207
An example makefile	208
Summary	210
Exercises	210

4: Data Abstraction 217

A tiny C-like library	219
Dynamic storage allocation	223
Bad guesses	227
What's wrong?	229
The basic object....	230
What's an object? .	238
Abstract data typing	239
Object details	240
Header file etiquette	241
Importance of header files	242
The multiple-declaration problem	244
The preprocessor directives #define, #ifdef, and #endif	245
A standard for header files	246
Namespaces in headers	247
Using headers in projects	248
Nested structures..	248
Global scope resolution	253
Summary	253

Exercises	254
5: Hiding the Implementation	259
Setting limits	260
C++ access control	261
protected	263
Friends	263
Nested friends	266
Is it pure?	269
Object layout	269
The class	270
Modifying Stash to use access control	273
Modifying Stack to use access control	274
Handle classes	275
Hiding the implementation	276
Reducing recompilation	276
Summary	279
Exercises	280
6: Initialization & Cleanup	283
Guaranteed initialization with the constructor	285
Guaranteed cleanup with the destructor	287
Elimination of the definition block	289
for loops	291
Storage allocation	292
Stash with constructors and destructors	294
Stack with constructors & destructors	298

Aggregate initialization	301
Default constructors	304
Summary	305
Exercises	306

7: Function Overloading & Default Arguments 309

More name decoration	311
Overloading on return values	312
Type-safe linkage	313
Overloading example	314
unions	318
Default arguments	321
Placeholder arguments	323
Choosing overloading vs. default arguments	324
Summary	329
Exercises	330

8: Constants 333

Value substitution	334
const in header files	335
Safety consts	336
Aggregates	337
Differences with C	338
Pointers	340
Pointer to const	340
const pointer	341
Assignment and type checking	343
Function arguments & return values	344
Passing by const value	344

Returning by const value	345
Passing and returning addresses	349
Classes	352
const in classes	353
Compile-time constants in classes.....	356
const objects & member functions.....	359
volatile.....	365
Summary	367
Exercises.....	367

9: Inline Functions 371

Preprocessor	
pitfalls	372
Macros and access	376
Inline functions	377
Inlines inside classes	378
Access functions	379
Stash & Stack	
with inlines	385
Inlines & the compiler	390
Limitations	390
Forward references	391
Hidden activities in constructors & destructors	392
Reducing clutter	393
More preprocessor features	395
Token pasting	396
Improved error checking	396
Summary	400
Exercises.....	400

10: Name Control 405

Static elements from C	406
static variables inside functions	406
Controlling linkage	412

Other storage class specifiers.....	414
Namespaces.....	414
Creating a namespace	415
Using a namespace	417
The use of namespaces.....	422

Static members	
in C++	423
Defining storage for static data members.....	424
Nested and local classes.....	428
static member functions.....	429

Static initialization	
dependency	432
What to do	434

Alternate linkage	
specifications.....	442
Summary.....	443
Exercises	443

11: References & the Copy-Constructor 449

Pointers in C++	450
References	
in C++	451
References in functions	452
Argument-passing guidelines	455
The copy- constructor	455
Passing & returning by value.....	455
Copy-construction	462
Default copy-constructor ...	468
Alternatives to copy- construction	471
Pointers	
to members	473
Functions	475
Summary.....	478
Exercises	479

12: Operator Overloading 485

Warning & reassurance	486
Syntax	487
Overloadable operators	488
Unary operators	489
Binary operators	493
Arguments & return values	505
Unusual operators	508
Operators you can't overload	517

Non-member operators	518
Basic guidelines	520

Overloading assignment	521
Behavior of operator=	522

Automatic type conversion	533
Constructor conversion	534
Operator conversion	535
Type conversion example	538
Pitfalls in automatic type conversion	539
Summary	542
Exercises	542

13: Dynamic Object Creation 547

Object creation	549
C's approach to the heap	550
operator new	552
operator delete	553
A simple example	553
Memory manager overhead	554
Early examples redesigned	555
delete void* is probably a bug	555
Cleanup responsibility with pointers	557

Stash for pointers	558
new & delete for arrays	563
Making a pointer more like an array	564
Running out of storage	565
Overloading new & delete	566
Overloading global new & delete	568
Overloading new & delete for a class	570
Overloading new & delete for arrays	573
Constructor calls	576
placement new & delete	577
Summary	580
Exercises	580

14: Inheritance & Composition 583

Composition syntax	584
Inheritance syntax	586
The constructor initializer list	588
Member object initialization	589
Built-in types in the initializer list	589
Combining composition & inheritance	591
Order of constructor & destructor calls	592
Name hiding	595
Functions that don't automatically inherit	600
Inheritance and static member functions	604

Choosing composition vs. inheritance	604
Subtyping	606
private inheritance	609
protected	610
protected inheritance	611
Operator overloading & inheritance	612
Multiple inheritance	613
Incremental development.....	614
Upcasting	615
Why "upcasting?"	617
Upcasting and the copy-constructor.....	617
Composition vs. inheritance (revisited)	620
Pointer & reference upcasting.....	622
A crisis	622
Summary	623
Exercises	623

15: Polymorphism & Virtual Functions 627

Evolution of C++ programmers	628
Upcasting	629
The problem	631
Function call binding	631
virtual functions ...	632
Extensibility	633
How C++ implements late binding	636
Storing type information	637
Picturing virtual functions...	639
Under the hood	642
Installing the vpointer	643
Objects are different	644
Why virtual functions?.....	645

Abstract base classes and pure virtual functions	646
Pure virtual definitions	651
Inheritance and the VTABLE	652
Object slicing.....	655
Overloading & overriding	658
Variant return type.....	660
virtual functions & constructors.....	662
Order of constructor calls ..	663
Behavior of virtual functions inside constructors	664
Destructors and virtual destructors.	665
Pure virtual destructors.....	668
Virtuals in destructors.....	670
Creating an object-based hierarchy	671
Operator overloading	675
Downcasting	678
Summary.....	681
Exercises	682

16: Introduction to Templates 689

Containers	690
The need for containers	692
Overview of templates.....	693
The template solution	696
Template syntax ...	697
Non-inline function definitions.....	699
IntStack as a template.....	701
Constants in templates	703
Stack and Stash as templates	705
Templatized pointer Stash .	707
Turning ownership	