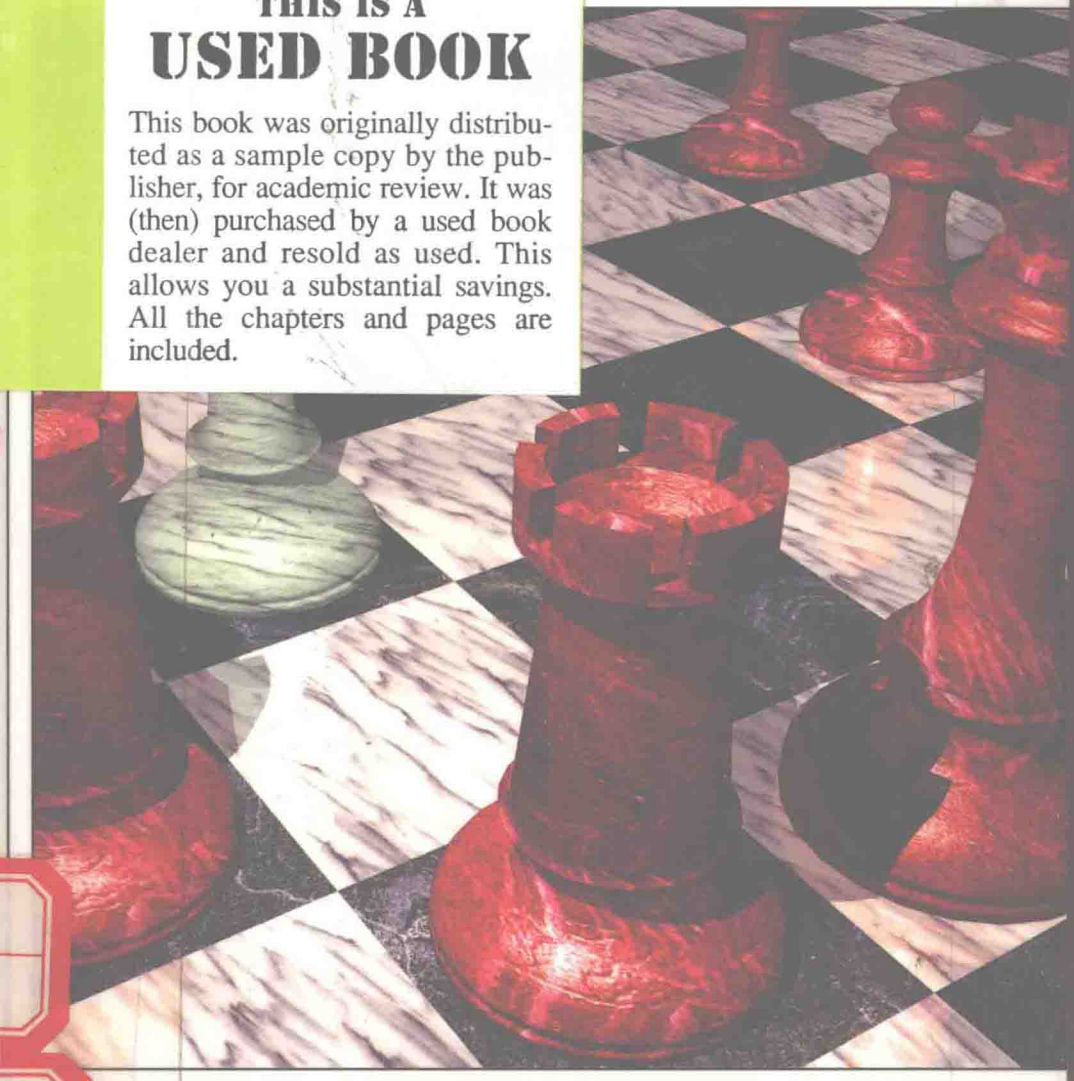
C++ FOR PASCAL PROGRAMMERS

SECOND EDITION



IRA POHL

C++ for Pascal Programmers

Second Edition

Ira Pohl

University of California, Santa Cruz

The Benjamin/Cummings Publishing Company, Inc.
Redwood City, California · Menlo Park, California
Reading, Massachusetts · New York · Don Mills, Ontario
Wokingham, U.K. · Amsterdam · Bonn · Sydney
Singapore · Tokyo · Madrid · San Juan

Acquisitions Editor: J. Carter Shanklin
Executive Editor: Dan Joraanstad
Editorial Assistant: Melissa Standen
Copy Editor: Elizabeth Gehrman
Text Design Consultant: Lisa Jahred
Cover Illustration: Joseph Maas, Paragon[3]
Production Editor: Ray Kanarr
Composition: Debra Dolsberry
Proofreader: Joe Ruddick
Cover Design: Yvo Riezebos

FrameMaker® is a trademark of Frame Technology Corporation. The camera-ready copy was prepared on a PC with FrameMaker®.

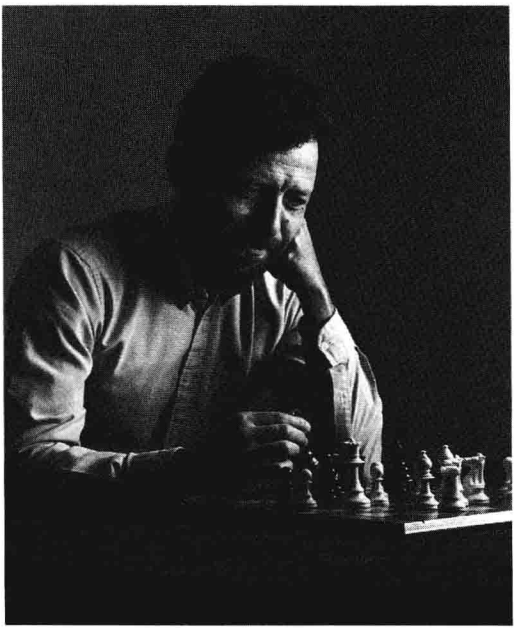
Library of Congress Cataloging-in-Publication Data
Pohl, Ira
C++ for Pascal Programmers / Ira Pohl -- 2nd ed.
p. cm.
Includes index
ISBN 0-8053-3158-1
1. C++ (Computer language program) I. Title.
QA76.73.C153P65 1994
005.13'3—dc20 94-24584
CIP

ISBN 0-8053-3158-1

1 2 3 4 5 6 7 8 9 10-MA-99 98 97 96 95 94

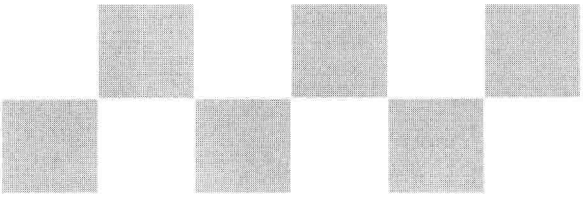
The Benjamin/Cummings Publishing Company, Inc.
390 Bridge Parkway
Redwood City, CA 94065

C++ for Pascal Programmers
Second Edition



About the Author

Ira Pohl, Ph.D., is a professor of Computer and Information Sciences at the University of California, Santa Cruz. He has two decades of experience as a software methodologist and is an international authority on C and C++ programming. His teaching and research interests include artificial intelligence and programming languages. Professor Pohl has lectured extensively at UC Berkeley, the Courant Institute, Edinburgh University, Stanford, the Vrije University in Amsterdam, and Auckland University in New Zealand. He is the author of the best-selling *Object-Oriented Programming Using C++*, and *C++ for C Programmers,* Second Edition, and coauthor, with Al Kelley, of *A Book on C: Programming in C,* Second Edition; and *C By Dissection,* Second Edition. When not programming, he enjoys riding bicycles in Aptos, California, with his wife Debra and daughter Laura.



Preface

This book is intended as an introduction to programming in C++ for the programmer or student already familiar with Pascal. It uses an evolutionary teaching process with Pascal as a starting point and C++ as a destination. The book is written to allow the reader to stop and use the language facilities up to that point in the text.

Pascal is the major teaching language for beginning computer science students. Designed by Niklaus Wirth in 1970, it is a small, powerful language, popular with both the academic community and the personal computer community. Many efficient and fast compilers exist for it, which indeed was one of its design goals. Pascal lacks some key features that limit its use in the professional community, where C is the dominant language.

C++, invented at Bell Labs by Bjarne Stroustrup in the mid-1980s, is a powerful modern successor language to C. C++ adds to C the concept of **class**, a mechanism for providing user-defined types also called **abstract data types**. It supports **object-oriented** programming by these means and by providing inheritance and run-time type binding. C is the present; C++ is the future.

By carefully developing working C++ programs, using the method of **dissection**, this book presents a simple and thorough introduction to the programming process in C++. Dissection is a technique for explaining new elements in a program that the student is seeing for the first time. It highlights key points in the many examples of working code that are used to teach by example.

This book is intended for use in a first course in programming in C++. The audience is expected to know Pascal or have enough programming experience to follow this tutorial. It can be used as a supplementary text in an advanced programming course, data structures course, software methodology course, comparative language course, or other courses where the instructor wants C++ to be the language of choice. Each chapter presents a number of carefully explained programs. Many programs and functions are dissected.

All the major pieces of code were tested. A consistent and proper coding style is adopted from the beginning. The style standard used is one chosen by professionals in the C++ community.

Pascal is a language of roughly the same size and utility as C. For the Pascal programmer who wants C experience, this book could be used in conjunction with *A Book on C, Second Edition* by Al Kelley and Ira Pohl (Redwood City, California: Benjamin/Cummings, 1990). As a package, the two books offer an integrated treatment of the C and C++ programming languages and their use that is unavailable elsewhere.

Each chapter contains:

Dissections. A program particularly illustrative of the chapter's themes is analyzed by dissection. Dissection is similar to a structured walk-through of the code. Its intention is to explain to the reader newly encountered programming elements and idioms.

Summary. A succinct list of points covered in the chapter are reiterated as helpful review.

Exercises. The exercises test the student's knowledge of the language. Many exercises are intended to be done interactively while reading the text. This encourages self-paced instruction by the reader. The exercises also frequently extend the reader's knowledge to an advanced area of use.

The book incorporates:

An Evolutionary Approach. The Pascal programmer is introduced to equivalent concepts in the C++ programming language. By learning how individual elements of a Pascal program translate into C++, the Pascal programmer can immediately gain a facility with the C++ programming language. Chapter 1, "An Overview of C++ and Object-Oriented Programming," provides an introduction to C++'s use as an object-oriented programming language. Chapter 2, "Native Types and Statements," shows the parallels between programming in Pascal and C++ with regard to data types, expressions, and simple statements. Chapter 3, "Functions and Pointers," continues with similarities between functions and complex data types . The middle chapters show how classes work. Classes are the basis for abstract data types and object-oriented programming. Again, the student starts from the perspective of Pascal and moves to C++. The later chapters give advanced details of the use of inheritance, templates, and exceptions. At any point in the text the programmer can stop and use the new material.

Teaching by Example. The book is a tutorial that stresses examples of working code. Right from the start the student is introduced to full

working programs. An interactive environment is assumed. Exercises are integrated with the examples to encourage experimentation. Excessive detail is avoided in explaining the larger elements of writing working code. Each chapter has several important example programs. Major elements of these programs are explained by dissection.

Data Structures in C++. The text emphasizes many of the standard data structures from computer science. Stacks, safe arrays, dynamically allocated multidimensional arrays, lists, trees, and strings are all implemented. Exercises extend the student's understanding of how to implement and use these structures. Implementation is consistent with an abstract data type approach to software.

Object-Oriented Programming. The reader is led gradually to the object-oriented style. Chapter 1, "An Overview of C++ and Object-Oriented Programming," discusses how the Pascal programmer can benefit in important ways from a switch to C++ and object-oriented programming (OOP). Object-oriented concepts are defined, and the way in which these concepts are supported by C++ is introduced. Chapter 4, "Classes," introduces classes, which are the basic mechanism for producing modular programs and implementing abstract data types. Class variables are the objects being manipulated. Chapter 7, "Inheritance," develops inheritance and virtual functions, two key elements in this paradigm. Chapter 10, "OOP Using C++," discusses OOP and the **Platonic** programming philosophy. This book develops in the programmer an appreciation of this point of view.

Turbo Pascal 7.0 Equivalence. Where appropriate, C++ code is given with equivalent Pascal code. This gives the experienced Pascal programmer immediate access to idiomatic C++ code. Wirth's Pascal has largely been superceded by commercially developed, extended Pascals that have many additional features, such as modules and OO extensions. Borland International's Turbo Pascal 7.0 has many OO features, and is among the most widely used. Where Turbo Pascal is specifically mentioned, we show equivalent code for OO features of C++.

ANSI C++ language and *iostream.h*. For an existing, widely used language, C++ continues to change at a rapid pace. This book is based on the most recent standard: the ANSI C++ Committee language documents. A succinct informal language reference is provided in Appendix D, "Language Guide." Chief additions include templates and exception handling. The examples use the *iostream.h* I/O library. This has replaced the older *stream.h* used in the first edition and *stdio.h* used in the C community. Use of the *iostream.h* library is described in Appendix E, "Input/Output."

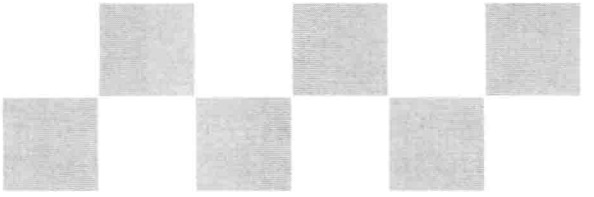
Industry- and Course-Tested. It is the basis of many on-site professional training courses given by the author, who has used its contents to train professionals and students in various forums since 1986. The various changes in the new edition are course-tested, and reflect considerable teaching and consulting experience by the author. In its first edition, the book won a UNIXWORLD commendation for the professional programmer migrating to C++.

Acknowledgments

My special thanks go to my wife, Debra Dolsberry, who encouraged me throughout this project. She acted as book designer and technical editor for this second edition. She developed appropriate formats and style sheets in FrameMaker 4.0 and guided the transition process from the first edition that was developed in troff. She also implemented and tested all major pieces of code. Her careful implementations of the code and exercises often led to important improvements. Stephen Clamage of TauMetric Corporation provided wonderfully insightful comments on language detail. Other reviewers who specifically commented on this edition were: Douglas Campbell, Brigham Young University; Blayne Mayfield, Oklahoma State University; and Henry Ruston, Polytechnic University. William Engles of the University of Wisconsin described an improved shuffling routine for the poker example.

The first edition had help, inspiration, and encouragement from, Nan Borreson, Borland International; Skona Brittain; Al Conrad; Steve Demurjian; Samuel Druker, Zortech Limited; Robert Durling; Bruce Eckel; Daniel Edelson; Gene Fisher; Robert Hansen, Lattice, Incorporated; John Hardin, Hewlett-Packard, Incorporated; Al Kelley; Jim Kempf, Sun Microsystems, Incorporated; Ellen Mickanin; Laura Pohl, Cottage Consultants; and Linda Werner. The first edition editors were Alan Apt and Mark McCormick. The second edition was developed with J. Carter Shanklin. Finally, I thank Bjarne Stroustrup for inventing such a powerful language and encouraging others to help teach it.

Ira Pohl
University of California, Santa Cruz



Contents

Chapter 1

An Overview of C++ and Object-Oriented Programming **1**

1.1 Object-Oriented Programming 3
1.2 Why C++ Is a Better C 3
1.3 Why Switch to C++? 5
1.4 Pascal as a Starting Point 6
1.5 Classes and Abstract Data Types 7
1.6 Overloading 10
1.7 Constructors and Destructors 13
1.8 Inheritance 15
1.9 Polymorphism 18
1.10 Templates 20
1.11 C++ Exceptions 22
1.12 Benefits of Object-Oriented Programming 23
1.13 References 24

Chapter 2

Native Types and Statements **27**

2.1 Program Elements 28
2.1.1 Comments 28
2.1.2 Keywords 29
2.1.3 Identifiers 29
2.1.4 Literals 30
2.1.5 Operators and Punctuators 32
2.2 Input/Output 32
2.3 Program Structure 34
2.4 Simple Types 37
2.4.1 Initialization 38
2.5 The Traditional Conversions 40
2.6 Enumeration Types 44
2.7 Expressions 45

2.8 Statements 50
2.8.1 Assignment and Expressions 51
2.8.2 The Compound Statement 52
2.8.3 The `if` and the `if-else` Statements 52
2.8.4 The `while` Statement 54
2.8.5 The `for` Statement 54
2.8.6 The `do` Statement 56
2.8.7 Transfer Statements 57
2.9 Summary 61
2.10 Exercises 62

Chapter 3
Functions and Pointers 71
3.1 Functions 72
3.1.1 Function Invocation 72
3.2 Function Definition 73
3.3 The `return` Statement 74
3.4 Function Prototypes 76
3.5 Default Arguments 79
3.6 Overloading Functions 80
3.7 Inlining 81
3.8 Scope and Storage Class 82
3.8.1 The Storage Class `auto` 84
3.8.2 The Storage Class `register` 85
3.8.3 The Storage Class `extern` 85
3.8.4 The Storage Class `static` 87
3.8.5 Linkage Mysteries 88
3.9 Pointer Types 89
3.9.1 Addressing and Dereferencing 90
3.9.2 Simulating Call-by-Reference 90
3.10 Reference Declarations and Call-by-Reference 92
3.11 The Uses of `void` 96
3.12 Arrays and Pointers 98
3.12.1 Subscripting 99
3.12.2 Initialization 100
3.13 The Relationship between Arrays and Pointers 100
3.14 Passing Arrays to Functions 102
3.15 Strings: A Kernel Language ADT 103
3.16 Multidimensional Arrays 105
3.17 Free Store Operators `new` and `delete` 105
3.18 Summary 108
3.19 Exercises 111

Chapter 4
Classes **119**
4.1 The Aggregate Type `struct` 120
4.2 Structure Pointer Operator 122
4.2.1 `typedef` and Casting 123
4.3 A Linked List 123
4.3.1 Dynamic Storage Allocation for a Linked List 125
4.3.2 Counting and Lookup 130
4.4 An Example: `Stack` 131
4.5 Unions 135
4.6 Bit Fields 138
4.7 Member Functions 140
4.8 Visibility `private` and `public` 143
4.9 Classes 144
4.10 `static` Member 146
4.11 Class Scope 147
4.11.1 Scope Resolution Operator `::` 147
4.11.2 Nested Classes 148
4.12 An Example: Flushing 149
4.13 Key Differences Between C++ and Pascal 154
4.14 Pragmatics 155
4.15 Turbo Pascal Equivalence 156
4.16 Summary 159
4.17 Exercises 160

Chapter 5
Constructors and Destructors **165**
5.1 Classes with Constructors 166
5.1.1 The Default Constructor 168
5.2 Constructing a Dynamically Sized Stack 169
5.2.1 The Copy Constructor 170
5.2.2 Constructor Initializer 171
5.3 Classes with Destructors 172
5.4 The `this` Pointer 173
5.5 `static` and `const` Member Functions 174
5.6 An Example: Dynamically Allocated Strings 178
5.7 A Class `vect` 182
5.8 Members That Are Class Types 185
5.9 An Example: A Singly Linked List 186
5.10 Two-Dimensional Arrays 191
5.11 Strings Using Reference Semantics 193

5.12 No Constructor, Copy Constructor, and Other Mysteries 195
5.12.1 Destructor Details 197
5.12.2 Constructors as Conversions 197
5.13 Pragmatics 199
5.14 Turbo Pascal Equivalence 199
5.15 Summary 202
5.16 Exercises 204

Chapter 6
Operator Overloading and Conversions 211
6.1 The Traditional Conversions 212
6.2 ADT Conversions 214
6.3 Overloading and Function Selection 216
6.4 Friend Functions 220
6.5 Overloading Operators 223
6.6 Unary Operator Overloading 224
6.7 Binary Operator Overloading 227
6.8 Overloading Assignment and Subscripting Operators 229
6.9 Signature Matching 233
6.10 Pragmatics 236
6.11 Summary 237
6.12 Exercises 239

Chapter 7
Inheritance 247
7.1 A Derived Class 248
7.2 Typing Conversions and Visibility 250
7.3 Code Reuse: Dynamic Array Bounds 253
7.4 Code Reuse: A Binary Tree Class 256
7.5 Virtual Functions 260
7.6 Abstract Base Classes 264
7.7 Multiple Inheritance 270
7.8 Detailed C++ Considerations 275
7.9 Pragmatics 276
7.10 Turbo Pascal Equivalence 278
7.11 Summary 280
7.12 Exercises 282

Chapter 8
Templates **289**
8.1 Template Class Stack 289
8.2 Function Templates 291
8.2.1 Signature Matching and Overloading 293
8.3 Class Templates 294
8.3.1 Friends 294
8.3.2 Static Members 295
8.3.3 Class Template Arguments 295
8.4 Parameterizing the Class `vect` 296
8.5 Parameterized Binary Search Tree 300
8.6 Inheritance 304
8.7 Pragmatics 306
8.8 Summary 307
8.9 Exercises 309

Chapter 9
Exceptions **313**
9.1 Using *assert.h* 314
9.2 C++ Exceptions 315
9.3 Throwing Exceptions 316
9.4 Try Blocks 319
9.5 Handlers 320
9.6 Exception Specification 320
9.7 `terminate()` and `unexpected()` 321
9.8 Example Exception Code 321
9.9 Pragmatics 324
9.10 Summary 325
9.11 Exercises 326

Chapter 10
OOP Using C++ **329**
10.1 OOP Language Requirements 330
10.2 ADTs in Non-OOP Languages 331
10.3 Clients and Manufacturers 332
10.4 Reuse and Inheritance 334
10.5 Polymorphism 334
10.6 Language Complexity 336
10.7 C++ OOP Bandwagon 337
10.8 Platonism: Tabula Rasa Design 338
10.9 Design Principles 340
10.10 Last Words 341
10.11 References 342
10.12 Summary 343

10.13 Exercises 345

Appendix A
ASCII Character Codes 347

Appendix B
Operator Precedence and Associativity 349

Appendix C
Turbo Pascal and C++ 351
C.1 Program Structure 351
C.2 Identifiers 354
C.3 Simple Data Types 355
C.4 Statements 355
C.5 Expressions 357
C.6 Procedures and Functions 357
C.7 Structured Types 359
C.8 Units and Objects 360
C.8.1 Constructors and Destructors 362
C.8.2 Inheritance 364

Appendix D
Language Guide 369
D.1 Lexical Elements 369
D.1.1 Comments 370
D.1.2 Identifiers 370
D.1.3 Keywords 371
D.2 Constants 372
D.3 Declarations and Scope Rules 374
D.4 Linkage Rules 377
D.5 Types 379
D.6 Conversion Rules 381
D.7 Expressions and Operators 384
D.7.1 `sizeof` Expressions 384
D.7.2 Autoincrement and Autodecrement Expressions 384
D.7.3 Arithmetic Expressions 385
D.7.4 Relational, Equality, and Logical Expressions 385
D.7.5 Assignment Expressions 387
D.7.6 Comma Expressions 388
D.7.7 Conditional Expressions 388
D.7.8 Bit Manipulation Expressions 389

	D.7.9 Address and Indirection Expressions	389
	D.7.10 new and delete Expressions	390
	D.7.11 Placement Syntax and Overloading	391
	D.7.12 Error Conditions	393
	D.7.13 Other Expressions	394
D.8	Statements	394
	D.8.1 Expression Statements	395
	D.8.2 The Compound Statement	395
	D.8.3 The if and if-else Statements	395
	D.8.4 The while Statement	396
	D.8.5 The for Statement	397
	D.8.6 The do Statement	398
	D.8.7 Transfer Statements	398
	D.8.8 The Declaration Statement	402
D.9	Classes	403
	D.9.1 Constructors and Destructors	403
	D.9.2 Member Functions	405
	D.9.3 The this Pointer	405
	D.9.4 static and const Member Functions	406
	D.9.5 Inheritance	407
	D.9.6 Multiple Inheritance	409
	D.9.7 Constructor Invocation	410
	D.9.8 Abstract Base Classes	411
	D.9.9 Pointer to Class Member	411
D.10	Functions	412
	D.10.1 Prototypes	412
	D.10.2 Overloading	413
	D.10.3 Call-by-Reference	415
	D.10.4 Inline	416
	D.10.5 Default Arguments	416
	D.10.6 Friend Functions	417
	D.10.7 Operator Overloading	418
	D.10.8 Virtual Functions	419
	D.10.9 Type-Safe Linkage	420
D.11	Templates	421
	D.11.1 Function Template	422
	D.11.2 Friends	423
	D.11.3 Static Members	424
	D.11.4 New Rules: Arguments and Specialization	424
D.12	Exceptions	425
	D.12.1 Throwing Exceptions	425
	D.12.2 Try Blocks	427
	D.12.3 Handlers	428
	D.12.4 Exception Specification	428
	D.12.5 terminate() and unexpected()	429

D.13 Caution and Compatibility 429
D.13.1 Nested Class Declarations 429
D.13.2 Type Compatibilities 430
D.13.3 Miscellaneous 430
D.14 New and Proposed Features 431
D.14.1 Types bool and wchar_t 432
D.14.2 Namespaces 432
D.14.3 Run-Time Type Identification 434
D.14.4 Mutable 435
D.14.5 Casts 435
D.14.6 Miscellaneous 437
D.15 Style Examples and Pragmatics 437
D.15.1 Class Definition Style 439

Appendix E
Input/Output **441**
E.1 The Output Class ostream 442
E.2 Formatted Output and *iomanip.h* 443
E.3 User-Defined Types: Output 445
E.4 The Input Class istream 447
E.5 Files 449
E.6 The Functions and Macros In *ctype.h* 453
E.7 Using Stream States 454
E.8 Mixing I/O Libraries 457

Index **459**