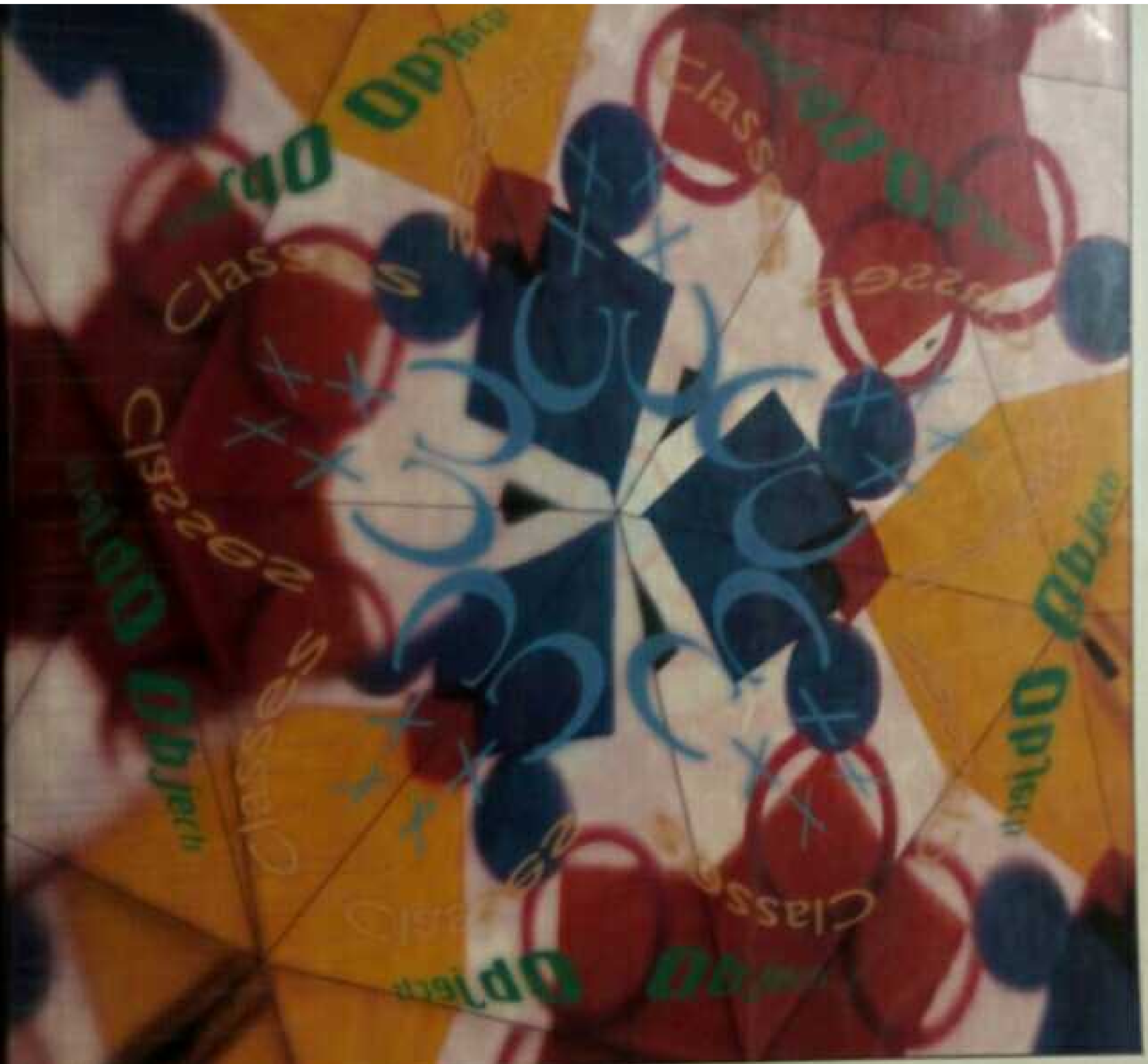
C++ PROGRAM DESIGN



An Introduction to
Programming and
Object-Oriented Design

James P. Cohoon ■ Jack W. Davidson

Contents

Chapter 1

Computing and the object-oriented design methodology, 1

1.1 Basic computing terminology, 2
- 1.1.1 Computing units of measure, 2
- 1.1.2 Computer organization, 3
- 1.1.3 Programming, 13

1.2 Software, 17
- 1.2.1 System software, 18
- 1.2.2 Application software, 21

1.3 Engineering software, 25
- 1.3.1 Software engineering principles, 27

1.4 Object-oriented design, 31
- 1.4.1 Object-oriented programming, 34

1.5 Points to remember, 39

1.6 To delve further, 42

1.7 Exercises, 43

Chapter 2

C++: the fundamentals, 47

2.1 Program organization, 48

2.2 A first program, 48

2.3 A second program, 50

2.4 Comments, 51

2.5 Assigning a value, 54

2.6 Fundamental C++ objects, 55
- 2.6.1 Integer object types, 56
- 2.6.2 Character object types, 57
- 2.6.3 Floating-point object types, 58

2.7 Constants, 59
- 2.7.1 String and character constants, 59
- 2.7.2 Integer constants, 62
- 2.7.3 Floating-point constants, 64

2.8 Names, 67
2.8.1 Keywords, 68
2.8.2 Identifiers, 69
2.9 Definitions, 71
2.10 Expressions, 74
2.10.1 Simple expressions, 75
2.10.2 Binary arithmetic operations, 75
2.10.3 Unary arithmetic operations, 80
2.10.4 Area of a circle, 80
2.10.5 Mixed-mode expressions, 81
2.10.6 Precedence, 82
2.10.7 Associativity, 84
2.11 Output statements, 85
2.12 Computing average velocity, 88
2.13 Points to remember, 91
2.14 Exercises, 94

Chapter 3
Modifying objects, 99
3.1 Assignment, 100
3.1.1 Assignment conversions, 102
3.1.2 Assignment precedence and associativity, 103
3.2 Const definitions, 104
3.3 Input statements, 105
3.4 Computing the number of molecules in a hydrocarbon, 107
3.5 Compound assignment, 110
3.6 Increment and decrement, 112
3.7 Estimating yearly savings of change, 114
3.8 Using programmer-defined objects, 116
3.8.1 Class SimpleWindow, 117
3.8.2 Class RectangleShape, 120
3.9 Mowing lawns, 121
3.10 Points to remember, 130
3.11 Exercises, 132

Chapter 4
Control constructs, 137
4.1 Boolean algebra, 138
4.1.1 Truth tables, 138
4.1.2 Logical expressions, 139
4.2 A Boolean type, 140
4.2.1 Boolean operators, 141
4.2.2 Relational operators, 142
4.2.3 Operator precedence revisited, 144
4.2.4 Short-circuit evaluation, 145
4.3 Conditional execution using the if statement, 146
4.3.1 The if-else statement, 149
4.3.2 Sorting three numbers, 153

4.4 Conditional execution using the switch statement, 155
4.5 Computing a requested expression, 158
4.6 Validating a date, 160
4.7 Iteration using the while statement, 166
4.7.1 Simple text processing, 173
4.8 Iteration using the for construct, 177
4.9 Solving the lazy hobo riddle, 184
4.10 Iteration using the do construct, 185
4.11 Points to remember, 186
4.12 Exercises, 190

Chapter 5
Function usage basics and libraries, 197

5.1 Function basics, 198
5.1.1 Interface specification, 199
5.1.2 Function prototyping, 201
5.1.3 Invocation and flow of control, 202
5.2 The preprocessor, 207
5.2.1 File inclusion directives, 207
5.2.2 Macro definitions, 208
5.2.3 Conditional compilation, 210
5.3 Using software libraries, 211
5.4 The iostream library, 212
5.4.1 Standard streams, 213
5.4.2 Standard error stream objects, 214
5.4.3 The iostream manipulators, 214
5.5 The iomanip library, 216
5.6 The fstream library, 221
5.7 A first look at the string library, 224
5.8 The math library, 225
5.8.1 Computing compound interest, 227
5.9 Library ctype, 228
5.10 The assert macros, 229
5.11 Points to remember, 232
5.12 To delve further, 233
5.13 Exercises, 234

Chapter 6
Programmer-defined functions, 237

6.1 Basics, 238
6.1.1 Function definition syntax, 239
6.1.2 Invocation and flow of control, 240
6.2 A tasty problem, 241
6.3 Some useful functions, 247
6.4 The local scope, 250
6.4.1 Local scope rules, 250
6.4.2 Name reuse with objects, 251

6.5 The global scope, 253
  6.5.1 Scope rules and name reuse with global objects, 254
  6.5.2 Initialization of global objects, 255
6.6 Displaying a price-interval chart, 256
6.7 Recursive functions, 264
6.8 Points to remember, 269
6.9 Exercises, 272

Chapter 7
Advanced parameter passing, 279
7.1 Reference parameters, 280
7.2 Passing objects by reference, 288
7.3 Validating telephone access codes, 290
7.4 Constant parameters, 293
7.5 Default parameters, 295
7.6 Casting of function parameters, 297
7.7 Function overloading, 298
7.8 Random numbers, 300
7.9 A factory automation trainer, 305
7.10 To delve further, 312
7.11 Points to remember, 315
7.12 Exercises, 318

Chapter 8
The class construct, 327
8.1 Introducing a programmer-defined data type, 328
8.2 The RectangleShape class, 330
  8.2.1 Public and private access, 336
8.3 Using the RectangleShape class, 338
8.4 Constructors, 341
8.5 Building a kaleidoscope, 342
8.6 Object-oriented analysis and design, 347
8.7 Points to remember, 356
8.8 To delve further, 359
8.9 Exercises, 359

Chapter 9
Abstract data types, 363
9.1 Introducing abstract data types, 364
9.2 Rational ADT basics, 364
  9.2.1 A client program using the rational library, 366
9.3 Rational class interface description, 370
  9.3.1 Access restrictions, 370
  9.3.2 Constructor member functions, 372
  9.3.3 Facilitator member functions, 372
  9.3.4 Inspect

9.3.5 Mutator member functions, 375
9.3.6 Data members, 375
9.3.7 Overloaded operators, 376
9.4 Implementing the Rational class, 378
9.4.1 Constructor definitions, 379
9.4.2 Inspector definitions, 379
9.4.3 Mutator definitions, 380
9.4.4 Arithmetic facilitator definitions, 380
9.4.5 Stream insertion and extraction facilitator definitions, 381
9.4.6 Auxiliary arithmetic operator definitions, 382
9.4.7 Auxiliary stream operator definitions, 383
9.5 Developing a random ADT library, 383
9.5.1 Random number implementation, 387
9.6 Red-yellow-green game, 391
9.6.1 Abstraction and interface, 391
9.6.2 Implementation of the class Element, 397
9.6.3 Implementation of the class Guess, 397
9.6.4 Implementation of the class Response, 400
9.6.5 Implementation of the class RYG, 402
9.7 Points to remember, 407
9.8 Exercises, 410

Chapter 10

Arrays, 415

10.1 Named collections, 416
10.2 One-dimensional arrays, 416
10.2.1 Array initialization, 418
10.2.2 Constant arrays, 419
10.2.3 Simple array processing, 420
10.3 Representing strings as character strings, 424
10.4 Arrays as parameters, 427
10.5 Sorting, 430
10.5.1 The InsertionSort() method, 432
10.5.2 Quality of InsertionSort(), 435
10.5.3 The QuickSort() method, 436
10.6 Binary searching, 440
10.7 Multidimensional arrays, 443
10.7.1 Matrices, 446
10.7.2 Find that word, 446
10.8 Representing a poker hand, 451
10.8.1 An ADT for playing cards, 451
10.8.2 An ADT for a deck of playing cards, 456
10.8.3 An ADT for a poker hand, 462
10.9 Points to remember, 468
10.10 Exercises, 470

[illegible], 475
Application programmer interfaces, 476
11.2 A simple window class, 477
11.2.1 Event-based programming, 477
11.2.2 SimpleWindow coordinate system, 478
11.2.3 Hello EzWindows, 481
11.2.4 EzWindows API mechanics, 484
11.3 The BitMap class, 486
11.4 Mouse events, 490
11.5 Bitmaps and mouse events, 494
11.6 Timer events, 496
11.7 Alert messages, 499
11.8 Simon says, 500
11.9 Points to remember, 516
11.10 Exercises, 517

Chapter 12
Pointer types, 521

12.1 Lvalues and rvalues, 522
12.2 Pointer basics, 522
12.2.1 Addressing and indirection, 525
12.2.2 Pointers to pointers, 528
12.2.3 Pointers as parameters, 529
12.3 Constant pointers and pointers to constants, 532
12.4 Arrays and pointers, 533
12.5 Character string processing, 537
12.6 Command-line parameters to a program, 540
12.7 Pointers to functions, 542
12.8 Pointers to remember, 546
12.9 Exercises, 547

Chapter 13
Dynamic data types, 551

13.1 Dynamic objects, 552
13.2 Constructing a string table, 558
13.3 An ADT for very big numbers, 563
13.3.1 BigNum class definition, 565
13.3.2 Member operators versus auxiliary operators, 565
13.3.3 Copy constructors, member assignment, and destructors, 57
13.3.4 Implementation of the BigNum constructors, 572
13.3.5 Inspectors and mutators, 574
13.3.6 Managing BigNum dynamic memory, 575
13.3.7 Member assignment and the this pointer, 576
13.3.8 Member addition with the BigNum class, 578
13.3.9 Multiplication in the BigNum class, 579
13.4 Points to remember, 580
13.5 Exercises, 584

Chapter 14
Inheritance, 587
14.1 Object-oriented design using inheritance, 588
14.2 Reuse via inheritance, 589
14.3 A hierarchy of shapes, 591
14.3.1 Declaring a derived class, 594
14.3.2 Implementing a derived class, 598
14.4 Protected members and inheritance, 605
14.5 Controlling inheritance, 608
14.5.1 Public inheritance, 608
14.5.2 Private inheritance, 609
14.5.3 Protected inheritance, 609
14.6 Multiple inheritance, 610
14.7 A prettier kaleidoscope, 617
14.8 Points to remember, 631
14.9 Exercises, 632

Chapter 15
Templates and polymorphism, 637
15.1 Generic actions and types, 638
15.2 Function templates, 638
15.3 Class templates, 641
15.4 A simple list class using a class template, 642
15.5 Sequential lists, 649
15.5.1 SeqItem class template, 650
15.5.2 SeqList member function basics, 652
15.5.3 Insert() member function, 652
15.5.4 Adding and removing elements, 655
15.5.5 A small program using SeqList template class, 657
15.5.6 A SeqList iterator class, 658
15.5.7 Implementation of the SeqElement iterator class, 660
15.6 Polymorphism, 662
15.7 Virtual function nuances, 664
15.8 Abstract base classes, 668
15.9 Virtual multiple inheritance, 672
15.10 Points to remember, 674
15.11 To delve further, 676
15.12 Exercises, 676

Chapter 16
Software project — bug hunt!, 681
16.1 Bug hunt, 682
16.2 Base class Bug, 683
16.2.1 Derived class SlowBug, 690
16.2.2 Derived class FastBug, 693
16.3 Class GameController, 697
16.4 Bug hunt, 701

CHAPTER 1

Computing and the object-oriented design methodology

Introduction

Computers are an integral part of life in the 90s. For example, most of us have used a word processing program. Computers are also being used in ways that are not as obvious. For example, every time you use your telephone, it likely connects to a computer system. Similarly, on your next plane trip, it may be that the aircraft was landed by a computer system and not the pilot! The term, *computer system*, is used to emphasize that there are two distinct components: hardware and software. The hardware is the computer itself. The software is the programs that tell the computer what to do. In the telephone system, it is the software that provides special features such as call waiting. Designing and building software is especially challenging today where a piece of software may consist of millions of lines of code. In recent years, the object-oriented programming design methodology has emerged and shown much promise for managing and coping with such complexity. In this chapter, we introduce basic computing terminology and the concepts behind object-oriented design. In successive chapters, we show how to design and write software using the object-oriented programming language C++.

Key Concepts

- CPU
- binary number system
- machine language
- system software
- application software
- operating system
- translation system
- compiler
- abstraction
- information hiding
- encapsulation
- modularization
- hierarchy
- reuse
- object-oriented design
- object-oriented language
- inheritance
- polymorphism

INTRODUCTION TO STATISTICS

Third Edition

RONALD E. WALPOLE