

# Contents

## Preface

xxii

## Before You Begin

I

### **1** Introduction to Computers, the Internet and the World Wide Web

I

|      |  |    |
|------|--|----|
| 1.1  | Introduction   | 2  |
| 1.2  | What Is a Computer?  | 4  |
| 1.3  | Computer Organization  | 4  |
| 1.4  | Early Operating Systems  | 5  |
| 1.5  | Personal, Distributed and Client/Server Computing                              | 6  |
| 1.6  | The Internet and the World Wide Web  | 6  |
| 1.7  | Web 2.0  | 7  |
| 1.8  | Machine Languages, Assembly Languages and High-Level Languages                 | 8  |
| 1.9  | History of C and C++   | 9  |
| 1.10 | C++ Standard Library   | 10 |
| 1.11 | History of Java  | 11 |
| 1.12 | Fortran, COBOL, Pascal and Ada   | 11 |
| 1.13 | BASIC, Visual Basic, Visual C++, C# and .NET                                   | 12 |
| 1.14 | Key Software Trend: Object Technology  | 13 |
| 1.15 | Typical C++ Development Environment  | 14 |
| 1.16 | Notes About C++ and <i>C++ How to Program, 6/e</i>                             | 16 |
| 1.17 | Test-Driving a C++ Application   | 17 |
| 1.18 | Software Technologies  | 23 |
| 1.19 | Game Programming with the Ogre Libraries                                       | 24 |
| 1.20 | Future of C++: Open Source Boost Libraries, TR1 and C++0x                      | 25 |
| 1.21 | Software Engineering Case Study: Introduction to Object Technology and the UML | 25 |
| 1.22 | Wrap-Up  | 30 |
| 1.23 | Web Resources  | 31 |

### **2** Introduction to C++ Programming

43

|     |   |    |
|-----|---|----|
| 2.1 | Introduction                                  | 44 |
| 2.2 | First Program in C++: Printing a Line of Text | 44 |
| 2.3 | Modifying Our First C++ Program               | 48 |
| 2.4 | Another C++ Program: Adding Integers          | 49 |

## x Contents

|     |  |    |
|-----|--|----|
| 2.5 | Memory Concepts  | 53 |
| 2.6 | Arithmetic   | 55 |
| 2.7 | Decision Making: Equality and Relational Operators                                       | 58 |
| 2.8 | (Optional) Software Engineering Case Study: Examining the ATM Requirements Specification | 63 |
| 2.9 | Wrap-Up  | 72 |

## 3 Introduction to Classes and Objects 81

|      |   |     |
|------|---|-----|
| 3.1  | Introduction  | 82  |
| 3.2  | Classes, Objects, Member Functions and Data Members   | 82  |
| 3.3  | Overview of the Chapter Examples  | 84  |
| 3.4  | Defining a Class with a Member Function   | 84  |
| 3.5  | Defining a Member Function with a Parameter   | 88  |
| 3.6  | Data Members, <i>set</i> Functions and <i>get</i> Functions   | 91  |
| 3.7  | Initializing Objects with Constructors  | 98  |
| 3.8  | Placing a Class in a Separate File for Reusability  | 102 |
| 3.9  | Separating Interface from Implementation  | 106 |
| 3.10 | Validating Data with <i>set</i> Functions   | 112 |
| 3.11 | (Optional) Software Engineering Case Study: Identifying the Classes in the ATM Requirements Specification | 117 |
| 3.12 | Wrap-Up   | 125 |

## 4 Control Statements: Part I 131

|      |  |     |
|------|--|-----|
| 4.1  | Introduction   | 132 |
| 4.2  | Algorithms   | 132 |
| 4.3  | Pseudocode   | 133 |
| 4.4  | Control Structures   | 134 |
| 4.5  | <i>if</i> Selection Statement  | 138 |
| 4.6  | <i>if...else</i> Double-Selection Statement  | 139 |
| 4.7  | <i>while</i> Repetition Statement  | 144 |
| 4.8  | Formulating Algorithms: Counter-Controlled Repetition                                      | 146 |
| 4.9  | Formulating Algorithms: Sentinel-Controlled Repetition                                     | 152 |
| 4.10 | Formulating Algorithms: Nested Control Statements  | 163 |
| 4.11 | Assignment Operators   | 169 |
| 4.12 | Increment and Decrement Operators  | 169 |
| 4.13 | (Optional) Software Engineering Case Study: Identifying Class Attributes in the ATM System | 173 |
| 4.14 | Wrap-Up  | 177 |

## 5 Control Statements: Part 2 192

|     |   |     |
|-----|---|-----|
| 5.1 | Introduction                                | 193 |
| 5.2 | Essentials of Counter-Controlled Repetition | 193 |
| 5.3 | <i>for</i> Repetition Statement             | 195 |
| 5.4 | Examples Using the <i>for</i> Statement     | 200 |

|      |  |     |
|------|--|-----|
| 5.5  | do...while Repetition Statement  | 204 |
| 5.6  | switch Multiple-Selection Statement  | 206 |
| 5.7  | break and continue Statements  | 216 |
| 5.8  | Logical Operators  | 218 |
| 5.9  | Confusing the Equality (==) and Assignment (=) Operators   | 222 |
| 5.10 | Structured Programming Summary   | 223 |
| 5.11 | (Optional) Software Engineering Case Study: Identifying Objects' States and Activities in the ATM System | 228 |
| 5.12 | Wrap-Up  | 233 |

## **6 Functions and an Introduction to Recursion 244**

|      |  |     |
|------|--|-----|
| 6.1  | Introduction   | 245 |
| 6.2  | Program Components in C++  | 246 |
| 6.3  | Math Library Functions   | 247 |
| 6.4  | Function Definitions with Multiple Parameters  | 249 |
| 6.5  | Function Prototypes and Argument Coercion  | 254 |
| 6.6  | C++ Standard Library Header Files  | 256 |
| 6.7  | Case Study: Random Number Generation   | 258 |
| 6.8  | Case Study: Game of Chance; Introducing enum   | 264 |
| 6.9  | Storage Classes  | 268 |
| 6.10 | Scope Rules  | 271 |
| 6.11 | Function Call Stack and Activation Records   | 274 |
| 6.12 | Functions with Empty Parameter Lists   | 278 |
| 6.13 | Inline Functions   | 279 |
| 6.14 | References and Reference Parameters  | 281 |
| 6.15 | Default Arguments  | 286 |
| 6.16 | Unary Scope Resolution Operator  | 288 |
| 6.17 | Function Overloading   | 289 |
| 6.18 | Function Templates   | 292 |
| 6.19 | Recursion  | 294 |
| 6.20 | Example Using Recursion: Fibonacci Series  | 298 |
| 6.21 | Recursion vs. Iteration  | 301 |
| 6.22 | (Optional) Software Engineering Case Study: Identifying Class Operations in the ATM System | 304 |
| 6.23 | Wrap-Up  | 311 |

## **7 Arrays and Vectors 333**

|       |  |     |
|-------|--|-----|
| 7.1   | Introduction   | 334 |
| 7.2   | Arrays   | 335 |
| 7.3   | Declaring Arrays   | 337 |
| 7.4   | Examples Using Arrays  | 337 |
| 7.4.1 | Declaring an Array and Using a Loop to Initialize the Array's Elements | 337 |
| 7.4.2 | Initializing an Array in a Declaration with an Initializer List        | 338 |

|       |  |     |
|-------|--|-----|
| 7.4.3 | Specifying an Array's Size with a Constant Variable and Setting Array Elements with Calculations | 340 |
| 7.4.4 | Summing the Elements of an Array   | 342 |
| 7.4.5 | Using Bar Charts to Display Array Data Graphically   | 343 |
| 7.4.6 | Using the Elements of an Array as Counters   | 345 |
| 7.4.7 | Using Arrays to Summarize Survey Results   | 346 |
| 7.4.8 | Using Character Arrays to Store and Manipulate Strings   | 349 |
| 7.4.9 | Static Local Arrays and Automatic Local Arrays   | 351 |
| 7.5   | Passing Arrays to Functions  | 354 |
| 7.6   | Case Study: Class GradeBook Using an Array to Store Grades                                       | 358 |
| 7.7   | Searching Arrays with Linear Search  | 365 |
| 7.8   | Sorting Arrays with Insertion Sort   | 366 |
| 7.9   | Multidimensional Arrays  | 369 |
| 7.10  | Case Study: Class GradeBook Using a Two-Dimensional Array  | 372 |
| 7.11  | Introduction to C++ Standard Library Class Template vector                                       | 379 |
| 7.12  | (Optional) Software Engineering Case Study: Collaboration Among Objects in the ATM System        | 384 |
| 7.13  | Wrap-Up  | 391 |

## **8**    Pointers and Pointer-Based Strings    **408**

|        |  |     |
|--------|--|-----|
| 8.1    | Introduction   | 409 |
| 8.2    | Pointer Variable Declarations and Initialization             | 410 |
| 8.3    | Pointer Operators  | 411 |
| 8.4    | Passing Arguments to Functions by Reference with Pointers    | 414 |
| 8.5    | Using const with Pointers                                    | 418 |
| 8.6    | Selection Sort Using Pass-by-Reference                       | 425 |
| 8.7    | sizeof Operator  | 428 |
| 8.8    | Pointer Expressions and Pointer Arithmetic                   | 431 |
| 8.9    | Relationship Between Pointers and Arrays                     | 434 |
| 8.10   | Arrays of Pointers   | 438 |
| 8.11   | Case Study: Card Shuffling and Dealing Simulation            | 439 |
| 8.12   | Function Pointers  | 445 |
| 8.13   | Introduction to Pointer-Based String Processing              | 450 |
| 8.13.1 | Fundamentals of Characters and Pointer-Based Strings         | 451 |
| 8.13.2 | String-Manipulation Functions of the String-Handling Library | 453 |
| 8.14   | Wrap-Up  | 461 |

## **9**    Classes: A Deeper Look, Part I    **487**

|     |  |     |
|-----|--|-----|
| 9.1 | Introduction   | 488 |
| 9.2 | Time Class Case Study                                      | 489 |
| 9.3 | Class Scope and Accessing Class Members                    | 494 |
| 9.4 | Separating Interface from Implementation                   | 496 |
| 9.5 | Access Functions and Utility Functions                     | 498 |
| 9.6 | Time Class Case Study: Constructors with Default Arguments | 500 |
| 9.7 | Destructors  | 506 |

|      |  |     |
|------|--|-----|
| 9.8  | When Constructors and Destructors Are Called   | 507 |
| 9.9  | Time Class Case Study: A Subtle Trap—Returning a Reference to a <code>private</code> Data Member | 510 |
| 9.10 | Default Memberwise Assignment  | 513 |
| 9.11 | (Optional) Software Engineering Case Study: Starting to Program the Classes of the ATM System    | 515 |
| 9.12 | Wrap-Up  | 523 |

## **10** Classes: A Deeper Look, Part 2 **530**

|       |   |     |
|-------|---|-----|
| 10.1  | Introduction  | 531 |
| 10.2  | <code>const</code> (Constant) Objects and <code>const</code> Member Functions     | 531 |
| 10.3  | Composition: Objects as Members of Classes  | 541 |
| 10.4  | <code>friend</code> Functions and <code>friend</code> Classes                     | 548 |
| 10.5  | Using the <code>this</code> Pointer   | 552 |
| 10.6  | Dynamic Memory Management with Operators <code>new</code> and <code>delete</code> | 557 |
| 10.7  | <code>static</code> Class Members   | 559 |
| 10.8  | Data Abstraction and Information Hiding   | 565 |
|       | 10.8.1 Example: Array Abstract Data Type  | 566 |
|       | 10.8.2 Example: String Abstract Data Type   | 567 |
|       | 10.8.3 Example: Queue Abstract Data Type  | 567 |
| 10.9  | Container Classes and Iterators   | 568 |
| 10.10 | Proxy Classes   | 568 |
| 10.11 | Wrap-Up   | 572 |

## **11** Operator Overloading; String and Array Objects **578**

|       |  |     |
|-------|--|-----|
| 11.1  | Introduction   | 579 |
| 11.2  | Fundamentals of Operator Overloading                         | 580 |
| 11.3  | Restrictions on Operator Overloading                         | 581 |
| 11.4  | Operator Functions as Class Members vs. Global Functions     | 583 |
| 11.5  | Overloading Stream Insertion and Stream Extraction Operators | 584 |
| 11.6  | Overloading Unary Operators                                  | 588 |
| 11.7  | Overloading Binary Operators                                 | 588 |
| 11.8  | Case Study: Array Class                                      | 589 |
| 11.9  | Converting between Types                                     | 601 |
| 11.10 | Case Study: String Class                                     | 602 |
| 11.11 | Overloading <code>++</code> and <code>--</code>              | 614 |
| 11.12 | Case Study: A Date Class                                     | 616 |
| 11.13 | Standard Library Class <code>string</code>                   | 620 |
| 11.14 | <code>explicit</code> Constructors                           | 624 |
| 11.15 | Wrap-Up  | 628 |

## **12** Object-Oriented Programming: Inheritance **640**

|      |              |     |
|------|--------------|-----|
| 12.1 | Introduction | 641 |
|------|--------------|-----|

|        |  |     |
|--------|--|-----|
| 12.2   | Base Classes and Derived Classes   | 642 |
| 12.3   | protected Members  | 645 |
| 12.4   | Relationship between Base Classes and Derived Classes                                    | 645 |
| 12.4.1 | Creating and Using a CommissionEmployee Class  | 646 |
| 12.4.2 | Creating a BasePlusCommissionEmployee Class Without Using Inheritance                    | 651 |
| 12.4.3 | Creating a CommissionEmployee–BasePlusCommissionEmployee Inheritance Hierarchy           | 657 |
| 12.4.4 | CommissionEmployee–BasePlusCommissionEmployee Inheritance Hierarchy Using protected Data | 662 |
| 12.4.5 | CommissionEmployee–BasePlusCommissionEmployee Inheritance Hierarchy Using private Data   | 669 |
| 12.5   | Constructors and Destructors in Derived Classes  | 677 |
| 12.6   | public, protected and private Inheritance  | 685 |
| 12.7   | Software Engineering with Inheritance  | 685 |
| 12.8   | Wrap-Up  | 687 |

## **13** Object-Oriented Programming: Polymorphism **693**

|        |   |     |
|--------|---|-----|
| 13.1   | Introduction  | 694 |
| 13.2   | Polymorphism Examples   | 696 |
| 13.3   | Relationships Among Objects in an Inheritance Hierarchy   | 697 |
| 13.3.1 | Invoking Base-Class Functions from Derived-Class Objects  | 697 |
| 13.3.2 | Aiming Derived-Class Pointers at Base-Class Objects   | 705 |
| 13.3.3 | Derived-Class Member-Function Calls via Base-Class Pointers   | 706 |
| 13.3.4 | Virtual Functions   | 708 |
| 13.3.5 | Summary of the Allowed Assignments Between Base-Class and Derived-Class Objects and Pointers                                    | 714 |
| 13.4   | Type Fields and switch Statements   | 715 |
| 13.5   | Abstract Classes and Pure virtual Functions   | 715 |
| 13.6   | Case Study: Payroll System Using Polymorphism   | 717 |
| 13.6.1 | Creating Abstract Base Class Employee   | 719 |
| 13.6.2 | Creating Concrete Derived Class SalariedEmployee  | 722 |
| 13.6.3 | Creating Concrete Derived Class HourlyEmployee  | 724 |
| 13.6.4 | Creating Concrete Derived Class CommissionEmployee  | 727 |
| 13.6.5 | Creating Indirect Concrete Derived Class BasePlusCommissionEmployee   | 729 |
| 13.6.6 | Demonstrating Polymorphic Processing  | 731 |
| 13.7   | (Optional) Polymorphism, Virtual Functions and Dynamic Binding “Under the Hood”   | 735 |
| 13.8   | Case Study: Payroll System Using Polymorphism and Runtime Type Information with Downcasting, dynamic_cast, typeid and type_info | 739 |
| 13.9   | Virtual Destructors   | 742 |
| 13.10  | (Optional) Software Engineering Case Study: Incorporating Inheritance into the ATM System                                       | 743 |
| 13.11  | Wrap-Up   | 751 |

|           |   |            |
|-----------|---|------------|
| <b>14</b> | <b>Templates</b>  | <b>756</b> |
| 14.1      | Introduction  | 757        |
| 14.2      | Function Templates  | 758        |
| 14.3      | Overloading Function Templates  | 761        |
| 14.4      | Class Templates   | 761        |
| 14.5      | Nontype Parameters and Default Types for Class Templates  | 768        |
| 14.6      | Notes on Templates and Inheritance  | 769        |
| 14.7      | Notes on Templates and Friends  | 769        |
| 14.8      | Notes on Templates and <code>static</code> Members  | 770        |
| 14.9      | Wrap-Up   | 771        |
| <b>15</b> | <b>Stream Input/Output</b>  | <b>776</b> |
| 15.1      | Introduction  | 777        |
| 15.2      | Streams   | 778        |
| 15.2.1    | Classic Streams vs. Standard Streams  | 779        |
| 15.2.2    | <code>iostream</code> Library Header Files  | 779        |
| 15.2.3    | Stream Input/Output Classes and Objects   | 779        |
| 15.3      | Stream Output   | 782        |
| 15.3.1    | Output of <code>char *</code> Variables   | 782        |
| 15.3.2    | Character Output Using Member Function <code>put</code>   | 782        |
| 15.4      | Stream Input  | 783        |
| 15.4.1    | <code>get</code> and <code>getline</code> Member Functions  | 784        |
| 15.4.2    | <code>istream</code> Member Functions <code>peek</code> , <code>putback</code> and <code>ignore</code>  | 787        |
| 15.4.3    | Type-Safe I/O   | 787        |
| 15.5      | Unformatted I/O Using <code>read</code> , <code>write</code> and <code>gcount</code>                    | 787        |
| 15.6      | Introduction to Stream Manipulators   | 788        |
| 15.6.1    | Integral Stream Base: <code>dec</code> , <code>oct</code> , <code>hex</code> and <code>setbase</code>   | 789        |
| 15.6.2    | Floating-Point Precision ( <code>precision</code> , <code>setprecision</code> )                         | 790        |
| 15.6.3    | Field Width ( <code>width</code> , <code>setw</code> )  | 791        |
| 15.6.4    | User-Defined Output Stream Manipulators   | 793        |
| 15.7      | Stream Format States and Stream Manipulators  | 794        |
| 15.7.1    | Trailing Zeros and Decimal Points ( <code>showpoint</code> )  | 795        |
| 15.7.2    | Justification ( <code>left</code> , <code>right</code> and <code>internal</code> )                      | 796        |
| 15.7.3    | Padding ( <code>fill</code> , <code>setfill</code> )  | 798        |
| 15.7.4    | Integral Stream Base ( <code>dec</code> , <code>oct</code> , <code>hex</code> , <code>showbase</code> ) | 799        |
| 15.7.5    | Floating-Point Numbers; Scientific and Fixed Notation ( <code>scientific</code> , <code>fixed</code> )  | 800        |
| 15.7.6    | Uppercase/Lowercase Control ( <code>uppercase</code> )  | 800        |
| 15.7.7    | Specifying Boolean Format ( <code>boolalpha</code> )  | 802        |
| 15.7.8    | Setting and Resetting the Format State via Member Function <code>flags</code>                           | 803        |
| 15.8      | Stream Error States   | 804        |
| 15.9      | Tying an Output Stream to an Input Stream   | 807        |
| 15.10     | Wrap-Up   | 807        |

|           |   |            |
|-----------|---|------------|
| <b>16</b> | <b>Exception Handling</b>                                       | <b>817</b> |
| 16.1      | Introduction  | 818        |
| 16.2      | Exception-Handling Overview                                     | 819        |
| 16.3      | Example: Handling an Attempt to Divide by Zero                  | 819        |
| 16.4      | When to Use Exception Handling                                  | 825        |
| 16.5      | Rethrowing an Exception   | 826        |
| 16.6      | Exception Specifications  | 828        |
| 16.7      | Processing Unexpected Exceptions                                | 829        |
| 16.8      | Stack Unwinding   | 829        |
| 16.9      | Constructors, Destructors and Exception Handling                | 831        |
| 16.10     | Exceptions and Inheritance                                      | 832        |
| 16.11     | Processing new Failures   | 832        |
| 16.12     | Class <code>auto_ptr</code> and Dynamic Memory Allocation       | 836        |
| 16.13     | Standard Library Exception Hierarchy                            | 839        |
| 16.14     | Other Error-Handling Techniques                                 | 840        |
| 16.15     | Wrap-Up   | 841        |
| <b>17</b> | <b>File Processing</b>  | <b>848</b> |
| 17.1      | Introduction  | 849        |
| 17.2      | Data Hierarchy  | 849        |
| 17.3      | Files and Streams   | 851        |
| 17.4      | Creating a Sequential File                                      | 852        |
| 17.5      | Reading Data from a Sequential File                             | 856        |
| 17.6      | Updating Sequential Files                                       | 863        |
| 17.7      | Random-Access Files   | 863        |
| 17.8      | Creating a Random-Access File                                   | 864        |
| 17.9      | Writing Data Randomly to a Random-Access File                   | 869        |
| 17.10     | Reading from a Random-Access File Sequentially                  | 871        |
| 17.11     | Case Study: A Transaction-Processing Program                    | 874        |
| 17.12     | Overview of Object Serialization                                | 881        |
| 17.13     | Wrap-Up   | 881        |
| <b>18</b> | <b>Class <code>string</code> and String Stream Processing</b>   | <b>893</b> |
| 18.1      | Introduction  | 894        |
| 18.2      | <code>string</code> Assignment and Concatenation                | 895        |
| 18.3      | Comparing strings   | 897        |
| 18.4      | Substrings  | 900        |
| 18.5      | Swapping strings  | 901        |
| 18.6      | <code>string</code> Characteristics                             | 902        |
| 18.7      | Finding Substrings and Characters in a <code>string</code>      | 904        |
| 18.8      | Replacing Characters in a <code>string</code>                   | 906        |
| 18.9      | Inserting Characters into a <code>string</code>                 | 908        |
| 18.10     | Conversion to C-Style Pointer-Based <code>char *</code> Strings | 909        |
| 18.11     | Iterators   | 911        |

|       |                          |     |
|-------|--------------------------|-----|
| 18.12 | String Stream Processing | 912 |
| 18.13 | Wrap-Up                  | 915 |

## **19 Searching and Sorting** **922**

|        |   |     |
|--------|---|-----|
| 19.1   | Introduction                            | 923 |
| 19.2   | Searching Algorithms                    | 923 |
| 19.2.1 | Efficiency of Linear Search             | 923 |
| 19.2.2 | Binary Search                           | 925 |
| 19.3   | Sorting Algorithms                      | 931 |
| 19.3.1 | Efficiency of Selection Sort            | 931 |
| 19.3.2 | Efficiency of Insertion Sort            | 931 |
| 19.3.3 | Merge Sort (A Recursive Implementation) | 932 |
| 19.4   | Wrap-Up                                 | 939 |

## **20 Data Structures** **945**

|      |   |     |
|------|---|-----|
| 20.1 | Introduction                                  | 946 |
| 20.2 | Self-Referential Classes                      | 947 |
| 20.3 | Dynamic Memory Allocation and Data Structures | 948 |
| 20.4 | Linked Lists                                  | 948 |
| 20.5 | Stacks  | 963 |
| 20.6 | Queues  | 968 |
| 20.7 | Trees   | 972 |
| 20.8 | Wrap-Up                                       | 980 |

## **21 Bits, Characters, C Strings and structs** **1004**

|       |   |      |
|-------|---|------|
| 21.1  | Introduction  | 1005 |
| 21.2  | Structure Definitions   | 1005 |
| 21.3  | Initializing Structures   | 1008 |
| 21.4  | Using Structures with Functions                                 | 1008 |
| 21.5  | typedef   | 1008 |
| 21.6  | Example: High-Performance Card Shuffling and Dealing Simulation | 1009 |
| 21.7  | Bitwise Operators   | 1012 |
| 21.8  | Bit Fields  | 1021 |
| 21.9  | Character-Handling Library                                      | 1025 |
| 21.10 | Pointer-Based String-Conversion Functions                       | 1031 |
| 21.11 | Search Functions of the Pointer-Based String-Handling Library   | 1036 |
| 21.12 | Memory Functions of the Pointer-Based String-Handling Library   | 1041 |
| 21.13 | Wrap-Up   | 1046 |

## **22 Standard Template Library (STL)** **1057**

|        |   |      |
|--------|---|------|
| 22.1   | Introduction to the Standard Template Library (STL) | 1059 |
| 22.1.1 | Introduction to Containers                          | 1060 |
| 22.1.2 | Introduction to Iterators                           | 1064 |

|         |   |      |
|---------|---|------|
| 22.1.3  | Introduction to Algorithms                            | 1069 |
| 22.2    | Sequence Containers                                   | 1071 |
| 22.2.1  | vector Sequence Container                             | 1072 |
| 22.2.2  | list Sequence Container                               | 1080 |
| 22.2.3  | deque Sequence Container                              | 1083 |
| 22.3    | Associative Containers                                | 1085 |
| 22.3.1  | multiset Associative Container                        | 1086 |
| 22.3.2  | set Associative Container                             | 1089 |
| 22.3.3  | multimap Associative Container                        | 1090 |
| 22.3.4  | map Associative Container                             | 1092 |
| 22.4    | Container Adapters                                    | 1094 |
| 22.4.1  | stack Adapter   | 1094 |
| 22.4.2  | queue Adapter   | 1096 |
| 22.4.3  | priority_queue Adapter                                | 1098 |
| 22.5    | Algorithms  | 1099 |
| 22.5.1  | fill, fill_n, generate and generate_n                 | 1100 |
| 22.5.2  | equal, mismatch and lexicographical_compare           | 1101 |
| 22.5.3  | remove, remove_if, remove_copy and remove_copy_if     | 1104 |
| 22.5.4  | replace, replace_if, replace_copy and replace_copy_if | 1106 |
| 22.5.5  | Mathematical Algorithms                               | 1109 |
| 22.5.6  | Basic Searching and Sorting Algorithms                | 1112 |
| 22.5.7  | swap, iter_swap and swap_ranges                       | 1114 |
| 22.5.8  | copy_backward, merge, unique and reverse              | 1116 |
| 22.5.9  | inplace_merge, unique_copy and reverse_copy           | 1118 |
| 22.5.10 | Set Operations  | 1120 |
| 22.5.11 | lower_bound, upper_bound and equal_range              | 1123 |
| 22.5.12 | Heapsort  | 1125 |
| 22.5.13 | min and max   | 1128 |
| 22.5.14 | STL Algorithms Not Covered in This Chapter            | 1128 |
| 22.6    | Class bitset  | 1130 |
| 22.7    | Function Objects                                      | 1134 |
| 22.8    | Wrap-Up   | 1137 |
| 22.9    | STL Web Resources                                     | 1138 |

## **23** Game Programming with Ogre

**1148**

|        |                                    |      |
|--------|------------------------------------|------|
| 23.1   | Introduction                       | 1149 |
| 23.2   | Installing Ogre, OgreAL and OpenAL | 1149 |
| 23.3   | Basics of Game Programming         | 1149 |
| 23.4   | The Game of Pong: Code Walkthrough | 1152 |
| 23.4.1 | Ogre Initialization                | 1153 |
| 23.4.2 | Creating a Scene                   | 1162 |
| 23.4.3 | Adding to the Scene                | 1164 |
| 23.4.4 | Animation and Timers               | 1175 |
| 23.4.5 | User Input                         | 1176 |
| 23.4.6 | Collision Detection                | 1178 |

|        |                    |      |
|--------|--------------------|------|
| 23.4.7 | Sound              | 1184 |
| 23.4.8 | Resources          | 1185 |
| 23.4.9 | Pong Driver        | 1185 |
| 23.5   | Wrap-Up            | 1186 |
| 23.6   | Ogre Web Resources | 1187 |

## **24 Boost Libraries, Technical Report 1 and C++0x** **1197**

|        |   |      |
|--------|---|------|
| 24.1   | Introduction  | 1198 |
| 24.2   | Deitel Online C++ and Related Resource Centers                | 1198 |
| 24.3   | Boost Libraries   | 1199 |
| 24.4   | Adding a New Library to Boost                                 | 1199 |
| 24.5   | Installing the Boost Libraries                                | 1200 |
| 24.6   | Boost Libraries in Technical Report 1 (TR1)                   | 1200 |
| 24.7   | Regular Expressions with the <code>Boost.Regex</code> Library | 1203 |
| 24.7.1 | Regular Expression Example                                    | 1204 |
| 24.7.2 | Validating User Input with Regular Expressions                | 1206 |
| 24.7.3 | Replacing and Splitting Strings                               | 1209 |
| 24.8   | Smart Pointers with <code>Boost.Smart_ptr</code>              | 1212 |
| 24.8.1 | Reference Counted <code>shared_ptr</code>                     | 1212 |
| 24.8.2 | <code>weak_ptr</code> : <code>shared_ptr</code> Observer      | 1217 |
| 24.9   | Technical Report 1  | 1223 |
| 24.10  | C++0x   | 1224 |
| 24.11  | Core Language Changes   | 1224 |
| 24.12  | Wrap-Up   | 1229 |

## **25 Other Topics** **1238**

|      |   |      |
|------|---|------|
| 25.1 | Introduction  | 1239 |
| 25.2 | <code>const_cast</code> Operator                                      | 1239 |
| 25.3 | namespaces  | 1241 |
| 25.4 | Operator Keywords   | 1245 |
| 25.5 | <code>mutable</code> Class Members                                    | 1247 |
| 25.6 | Pointers to Class Members ( <code>.*</code> and <code>-&gt;*</code> ) | 1249 |
| 25.7 | Multiple Inheritance  | 1251 |
| 25.8 | Multiple Inheritance and <code>virtual</code> Base Classes            | 1256 |
| 25.9 | Wrap-Up   | 1260 |

## **A Operator Precedence and Associativity Chart** **1266**

|     |                     |      |
|-----|---------------------|------|
| A.1 | Operator Precedence | 1266 |
|-----|---------------------|------|

## **B ASCII Character Set** **1269**

## **C Fundamental Types** **1270**

|          |   |             |
|----------|---|-------------|
| <b>D</b> | <b>Number Systems</b>   | <b>1272</b> |
| D.1      | Introduction  | 1273        |
| D.2      | Abbreviating Binary Numbers as Octal and Hexadecimal Numbers                | 1276        |
| D.3      | Converting Octal and Hexadecimal Numbers to Binary Numbers                  | 1277        |
| D.4      | Converting from Binary, Octal or Hexadecimal to Decimal                     | 1277        |
| D.5      | Converting from Decimal to Binary, Octal or Hexadecimal                     | 1278        |
| D.6      | Negative Binary Numbers: Two's Complement Notation                          | 1280        |
| <b>E</b> | <b>C Legacy Code Topics</b>   | <b>1285</b> |
| E.1      | Introduction  | 1286        |
| E.2      | Redirecting Input/Output on UNIX/Linux/Mac OS X and Windows Systems         | 1286        |
| E.3      | Variable-Length Argument Lists  | 1287        |
| E.4      | Using Command-Line Arguments  | 1290        |
| E.5      | Notes on Compiling Multiple-Source-File Programs                            | 1291        |
| E.6      | Program Termination with <code>exit</code> and <code>atexit</code>          | 1293        |
| E.7      | Type Qualifier <code>volatile</code>  | 1295        |
| E.8      | Suffixes for Integer and Floating-Point Constants                           | 1295        |
| E.9      | Signal Handling   | 1295        |
| E.10     | Dynamic Memory Allocation with <code>calloc</code> and <code>realloc</code> | 1298        |
| E.11     | Unconditional Branch: <code>goto</code>                                     | 1299        |
| E.12     | Unions  | 1300        |
| E.13     | Linkage Specifications  | 1303        |
| E.14     | Wrap-Up   | 1304        |
| <b>F</b> | <b>Preprocessor</b>   | <b>1311</b> |
| F.1      | Introduction  | 1312        |
| F.2      | The <code>#include</code> Preprocessor Directive                            | 1312        |
| F.3      | The <code>#define</code> Preprocessor Directive: Symbolic Constants         | 1313        |
| F.4      | The <code>#define</code> Preprocessor Directive: Macros                     | 1313        |
| F.5      | Conditional Compilation   | 1315        |
| F.6      | The <code>#error</code> and <code>#pragma</code> Preprocessor Directives    | 1316        |
| F.7      | Operators <code>#</code> and <code>##</code>                                | 1317        |
| F.8      | Predefined Symbolic Constants   | 1317        |
| F.9      | Assertions  | 1318        |
| F.10     | Wrap-Up   | 1318        |
| <b>G</b> | <b>ATM Case Study Code</b>  | <b>1323</b> |
| G.1      | ATM Case Study Implementation   | 1323        |
| G.2      | Class <code>ATM</code>  | 1324        |
| G.3      | Class <code>Screen</code>   | 1331        |
| G.4      | Class <code>Keypad</code>   | 1332        |
| G.5      | Class <code>CashDispenser</code>  | 1333        |

|      |                               |      |
|------|-------------------------------|------|
| G.6  | Class DepositSlot             | 1335 |
| G.7  | Class Account                 | 1336 |
| G.8  | Class BankDatabase            | 1338 |
| G.9  | Class Transaction             | 1342 |
| G.10 | Class BalanceInquiry          | 1344 |
| G.11 | Class Withdrawal              | 1346 |
| G.12 | Class Deposit                 | 1351 |
| G.13 | Test Program ATMCASEStudy.cpp | 1354 |
| G.14 | Wrap-Up                       | 1354 |

## **H** UML 2: Additional Diagram Types **1355**

|     |                          |      |
|-----|--------------------------|------|
| H.1 | Introduction             | 1355 |
| H.2 | Additional Diagram Types | 1355 |

## **I** Using the Visual Studio Debugger **1357**

|     |  |      |
|-----|--|------|
| I.1 | Introduction   | 1358 |
| I.2 | Breakpoints and the <b>Continue</b> Command  | 1358 |
| I.3 | <b>Locals</b> and <b>Watch</b> Windows   | 1363 |
| I.4 | Controlling Execution Using the <b>Step Into</b> , <b>Step Over</b> , <b>Step Out</b> and <b>Continue</b> Commands | 1366 |
| I.5 | <b>Autos</b> Window  | 1369 |
| I.6 | Wrap-Up  | 1370 |

## **J** Using the GNU C++ Debugger **1373**

|     |  |      |
|-----|--|------|
| J.1 | Introduction   | 1374 |
| J.2 | Breakpoints and the <b>run</b> , <b>stop</b> , <b>continue</b> and <b>print</b> Commands | 1374 |
| J.3 | <b>print</b> and <b>set</b> Commands   | 1381 |
| J.4 | Controlling Execution Using the <b>step</b> , <b>finish</b> and <b>next</b> Commands     | 1383 |
| J.5 | <b>watch</b> Command   | 1385 |
| J.6 | Wrap-Up  | 1387 |

## **Bibliography** **1390**

## **Index** **1396**