Contents

1 Getting Started 1
- 1.1 A Brief Introduction to C++ 1
  - 1.1.1 C++ is "Object-Oriented" 2
  - 1.1.2 Why You Should Write Scientific Programs in C++ 3
  - 1.1.3 Why You Should Not Write Scientific Programs in C++ 4
  - 1.1.4 Scope of This Book 5
- 1.2 A First C++ Program 5
- 1.3 Compiling a C++ Program 6
  - 1.3.1 Integrated Development Environments 7
  - 1.3.2 Compiling at the Command Line 8
  - 1.3.3 Compiler Flags 9
- 1.4 Variables 10
  - 1.4.1 Basic Numerical Variables 10
  - 1.4.2 Other Numerical Variables 12
  - 1.4.3 Mathematical Operations on Numerical Variables 14
  - 1.4.4 Division of Integers 16
  - 1.4.5 Arrays 17
  - 1.4.6 ASCII Characters 18
  - 1.4.7 Boolean Variables 19
  - 1.4.8 Strings 19
- 1.5 Simple Input and Output 20
  - 1.5.1 Basic Console Output 20
  - 1.5.2 Keyboard Input 21
- 1.6 The `assert` Statement 22
- 1.7 Tips: Debugging Code 24
- 1.8 Exercises 25

2 Flow of Control 27
- 2.1 The `if` Statement 28
  - 2.1.1 A Single **`if`** Statement 28
  - 2.1.2 Example: Code for a Single **`if`** Statement 29

2.1.3 **if–else** Statements 29
2.1.4 Multiple **if** Statements 30
2.1.5 Nested **if** Statements 30
2.1.6 Boolean Variables 31
2.2 Logical and Relational Operators 31
2.3 The while Statement 33
2.4 Loops Using the **for** Statement 35
2.4.1 Example: Calculating the Scalar Product of Two Vectors 36
2.5 The switch Statement 37
2.6 Tips: Loops and Branches 38
2.6.1 Tip 1: A Common Novice Coding Error 38
2.6.2 Tip 2: Counting from Zero 38
2.6.3 Tip 3: Equality Versus Assignment 39
2.6.4 Tip 4: Never Ending **while** Loops 41
2.6.5 Tip 5: Comparing Two Floating Point Numbers 41
2.7 Exercises 42

3 File Input and Output 47
3.1 Redirecting Console Output to File 47
3.2 Writing to File 48
3.2.1 Setting the Precision of the Output 51
3.3 Reading from File 51
3.4 Checking Input and Output are Successful 53
3.5 Reading from the Command Line 54
3.6 Tips: Controlling Output Format 55
3.7 Exercises 56

4 Pointers 59
4.1 Pointers and the Computer's Memory 59
4.1.1 Addresses 59
4.1.2 Pointer Variables 60
4.1.3 Example Use of Pointers 61
4.1.4 Warnings on the Use of Pointers 61
4.2 Dynamic Allocation of Memory for Arrays 62
4.2.1 Vectors 63
4.2.2 Matrices 64
4.2.3 Irregularly Sized Matrices 65
4.3 Tips: Pointers 66
4.3.1 Tip 1: Pointer Aliasing 66
4.3.2 Tip 2: Safe Dynamic Allocation 67
4.3.3 Tip 3: Every **new** Has a **delete** 68

4.4 Modern C++ Memory Management 69
4.4.1 The `unique_ptr` Smart Pointer 69
4.4.2 The `shared_ptr` Smart Pointer 71
4.5 Exercises 72

5 Blocks, Functions and Reference Variables 75
5.1 Blocks 75
5.2 Functions 77
5.2.1 Simple Functions 77
5.2.2 Returning Pointer Variables from a Function 79
5.2.3 Use of Pointers as Function Arguments 80
5.2.4 Sending Arrays to Functions 82
5.2.5 Example: A Function to Calculate the Scalar Product of Two Vectors 84
5.3 Reference Variables 85
5.4 Default Values for Function Arguments 86
5.5 Function Overloading 87
5.6 Declaring Functions Without Prototypes 89
5.7 Function Pointers 89
5.8 Recursive Functions 92
5.9 Modules 93
5.10 Tips: Code Documentation 94
5.11 Exercises 96

6 An Introduction to Classes 99
6.1 The *Raison d'Être* for Classes 99
6.1.1 Problems That May Arise When Using Modules 100
6.1.2 Abstraction, Encapsulation and Modularity Properties of Classes 100
6.2 A First Example Simple Class: A Class of Books 101
6.2.1 Basic Features of Classes 101
6.2.2 Header Files 103
6.2.3 Setting and Accessing Variables 104
6.2.4 Compiling Multiple Files 107
6.2.5 Access Privileges 109
6.2.6 Including Function Implementations in Header Files 110
6.2.7 Constructors and Destructors 110
6.2.8 Pointers to Classes 115
6.3 The **`friend`** Keyword 116
6.4 A Second Example Class: A Class of Complex Numbers 117
6.4.1 Operator Overloading 118
6.4.2 The Class of Complex Numbers 119

6.5 Some Additional Remarks on Operator Overloading 125
6.6 Tips: Coding to a Standard 125
6.7 Exercises 127

7 Inheritance and Derived Classes 129
7.1 Inheritance, Extensibility and Polymorphism 129
7.2 Example: A Class of E-books Derived from a Class of Books 130
7.3 Access Privileges for Derived Classes 133
7.4 Classes Derived from Derived Classes 134
7.5 Run-Time Polymorphism 134
7.6 The Abstract Class Pattern 137
7.7 Tips: Using a Debugger 138
7.8 Exercises 139

8 Templates 145
8.1 Templates to Control Dimensions and Verify Sizes 145
8.2 Templates for Polymorphism 147
8.3 A Brief Survey of the Standard Template Library 148
8.3.1 Vectors 148
8.3.2 Sets 151
8.4 A Survey of Some New Functionality in Modern C++ 153
8.4.1 The `auto` Type 154
8.4.2 Some Useful Container Types with Unified Functionality 155
8.4.3 Range-based `for` Loops 157
8.4.4 Mapping Lambda Functions 158
8.5 Tips: Template Compilation 159
8.6 Exercises 160

9 Errors, Exceptions and Testing 163
9.1 Preconditions 164
9.1.1 Example: Two Implementations of a Graphics Function 164
9.2 Three Levels of Errors 166
9.3 Introducing the Exception 167
9.4 Using Exceptions 168
9.5 Testing Software 169
9.5.1 Unit Testing 170
9.5.2 Extending Software 171
9.5.3 Black Box Testing 172
9.5.4 White Box Testing 176
9.5.5 Test Driven Development 177
9.6 Tips: Writing Appropriate Tests 178
9.7 Exercises 179

10 Developing Classes for Linear Algebra Calculations 183
10.1 Requirements of the Linear Algebra Classes 183
10.2 Constructors and Destructors 188
10.2.1 The Default Constructor 188
10.2.2 The Copy Constructor 188
10.2.3 A Specialised Constructor 189
10.2.4 Destructor 189
10.3 Accessing Private Class Members 189
10.3.1 Accessing the Size of a Vector 190
10.3.2 Overloading the Square Bracket Operator 190
10.3.3 Read-Only Access to Vector Entries 190
10.3.4 Overloading the Round Bracket Operator 190
10.4 Operator Overloading for Vector Operations 190
10.4.1 The Assignment Operator 191
10.4.2 Unary Operators 191
10.4.3 Binary Operators 191
10.5 Functions 191
10.5.1 Members Versus Friends 191
10.6 Tips: Memory Debugging Tools 192
10.7 Exercises 193

11 An Introduction to Parallel Programming Using MPI 197
11.1 Distributed Memory Architectures 197
11.2 Installing MPI 199
11.3 A First Program Using MPI 199
11.3.1 Essential MPI Functions 201
11.3.2 Compiling and Running MPI Code 201
11.4 Basic MPI Communication 203
11.4.1 Point-to-Point Communication 203
11.4.2 Collective Communication 206
11.5 Example MPI Applications 213
11.5.1 Summation of Series 213
11.5.2 Parallel Linear Algebra 215
11.6 Tips: Debugging a Parallel Program 218
11.6.1 Tip 1: Make an Abstract Program 219
11.6.2 Tip 2: Datatype Mismatch 219
11.6.3 Tip 3: Intermittent Deadlock 220
11.6.4 Tip 4: Almost Collective Communication 220
11.7 Exercises 221

12 Designing Object-Oriented Numerical Libraries 225
12.1 Developing the Library for Ordinary Differential Equations . . . 226
12.1.1 Model Problems 226
12.1.2 Finite Difference Approximation to Derivatives 227

12.1.3 Application of Finite Difference Methods to Boundary Value Problems 229
12.1.4 Concluding Remarks on Boundary Value Problems in One Dimension 231
12.2 Designing a Library for Solving Boundary Value Problems . . . 232
12.2.1 The Class **`SecondOrderOde`** 233
12.2.2 The Class **`BoundaryConditions`** 234
12.2.3 The Class **`FiniteDifferenceGrid`** 235
12.2.4 The Class **`BvpOde`** 236
12.2.5 Using the Class **`BvpOde`** 237
12.3 Extending the Library to Two Dimensions 238
12.3.1 Model Problem for Two Dimensions 239
12.3.2 Finite Difference Methods for Boundary Value Problems in Two Dimensions 239
12.3.3 Setting Up the Linear System for the Model Problem 241
12.3.4 Developing the Classes Required 242
12.4 Tips: Using Well-Written Libraries 243
12.5 Exercises 243

Appendix A: Linear Algebra 245

Appendix B: Other Programming Constructs You Might Meet 257

Appendix C: Solutions to Exercises 263

Further Reading 281

Index 283