# Contents

## Part I  C++ programming basics  

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Introduction</td>
<td>3</td>
</tr>
<tr>
<td>1.1 Objective</td>
<td>3</td>
</tr>
<tr>
<td>1.2 Presentation</td>
<td>3</td>
</tr>
<tr>
<td>1.3 Why C++</td>
<td>4</td>
</tr>
<tr>
<td>1.4 C++ standards</td>
<td>6</td>
</tr>
<tr>
<td>1.5 Summary</td>
<td>7</td>
</tr>
<tr>
<td>1.6 How to use this text</td>
<td>7</td>
</tr>
<tr>
<td>1.7 Additional study aids</td>
<td>8</td>
</tr>
<tr>
<td>1.8 Additional and alternative software packages</td>
<td>8</td>
</tr>
<tr>
<td>2 Installing and running the Dev-C++ programming environment</td>
<td>10</td>
</tr>
<tr>
<td>2.1 Compiling and running a first program</td>
<td>10</td>
</tr>
<tr>
<td>2.2 Using the Dev-C++ debugger</td>
<td>12</td>
</tr>
<tr>
<td>2.3 Installing DISLIN and gsl</td>
<td>13</td>
</tr>
<tr>
<td>2.4 A first graphics program</td>
<td>14</td>
</tr>
<tr>
<td>2.5 The help system</td>
<td>15</td>
</tr>
<tr>
<td>2.6 Linux alternatives</td>
<td>16</td>
</tr>
<tr>
<td>2.7 Assignment</td>
<td>16</td>
</tr>
<tr>
<td>3 Introduction to computer and software architecture</td>
<td>17</td>
</tr>
<tr>
<td>3.1 Computational methods</td>
<td>17</td>
</tr>
<tr>
<td>3.2 Hardware architecture</td>
<td>18</td>
</tr>
<tr>
<td>3.3 Software architecture</td>
<td>20</td>
</tr>
<tr>
<td>3.4 The operating system and application software</td>
<td>23</td>
</tr>
<tr>
<td>3.5 Assignments</td>
<td>23</td>
</tr>
<tr>
<td>4 Fundamental concepts</td>
<td>25</td>
</tr>
<tr>
<td>4.1 Overview of program structure</td>
<td>25</td>
</tr>
<tr>
<td>4.2 Tokens, names, and keywords</td>
<td>25</td>
</tr>
<tr>
<td>4.3 Expressions and statements</td>
<td>26</td>
</tr>
<tr>
<td>4.4 Constants, variables, and identifiers</td>
<td>26</td>
</tr>
<tr>
<td>4.5 Declarations, definitions, and scope</td>
<td>27</td>
</tr>
</tbody>
</table>
4.6 rvalues and lvalues 28
4.7 Block structure 28
4.8 The const keyword 29
4.9 Operators – precedence and associativity 30
4.10 Formatting conventions 31
4.11 Comments 32
4.12 Assignments 33

5 Writing a first program 37
5.1 The main() function 37
5.2 Namespaces 37
5.3 #include Statements 38
5.4 Input and output streams 39
5.5 File streams 40
5.6 Constant and variable types 41
5.7 Casts 44
5.8 Operators 45
5.9 Control flow 46
5.10 Functions 47
5.11 Arrays and typedefs 47
5.12 A first look at scientific software development 48
5.13 Program errors 51
5.14 Numerical errors with floating-point types 53
5.15 Assignments 55

6 An introduction to object-oriented analysis 62
6.1 Procedural versus object-oriented programming 62
6.2 Problem definition 65
6.3 Requirements specification 66
6.4 UML diagrams 66
6.5 Use case diagram 67
6.6 Classes and objects 68
6.7 Object discovery 71
6.8 Sequence and collaboration diagrams 72
6.9 Aggregation and association 74
6.10 Inheritance 75
6.11 Object-oriented programming approaches 78
6.12 Assignments 79

7 C++ object-oriented programming syntax 83
7.1 Class declaration 83
7.2 Class definition and member functions 83
7.3 Object creation and polymorphism 86
7.4 Information hiding 87  
7.5 Constructors 89  
7.6 Wrapping legacy code 91  
7.7 Inheritance 92  
7.8 The ‘protected’ keyword 94  
7.9 Assignments 95  

8 Control logic and iteration 104  
8.1 The bool and enum types 104  
8.2 Logical operators 106  
8.3 if statements and implicit blocks 107  
8.4 else, else if, conditional and switch statements 108  
8.5 The exit() function 109  
8.6 Conditional compilation 109  
8.7 The for statement 110  
8.8 while and do...while statements 111  
8.9 The break and continue statements 112  
8.10 Assignments 112  

9 Basic function properties 119  
9.1 Principles of function operation 119  
9.2 Function declarations and prototypes 121  
9.3 Overloading and argument conversion 121  
9.4 Built-in functions and header files 122  
9.5 Program libraries 124  
9.6 Function preconditions and postconditions – the assert statement 128  
9.7 Multiple return statements 130  
9.8 Functions and global variables 130  
9.9 Use of const in functions 131  
9.10 Default parameters 132  
9.11 Inline functions 132  
9.12 Modular programming 133  
9.13 Recursive functions 134  
9.14 Assignments 134  

10 Arrays and matrices 140  
10.1 Data structures and arrays 140  
10.2 Array definition and initialization 141  
10.3 Array manipulation and memory access 142  
10.4 Arrays as function parameters 144  
10.5 Returning manipulation and object arrays 145  
10.6 const arrays 146  
10.7 Matrices 146
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.8 Matrix storage and loop order</td>
<td>147</td>
</tr>
<tr>
<td>10.9 Matrices as function arguments</td>
<td>150</td>
</tr>
<tr>
<td>10.10 Assignments</td>
<td>150</td>
</tr>
<tr>
<td><strong>11 Input and output streams</strong></td>
<td></td>
</tr>
<tr>
<td>11.1 The <em>iostream</em> class and stream manipulators</td>
<td>158</td>
</tr>
<tr>
<td>11.2 File streams</td>
<td>161</td>
</tr>
<tr>
<td>11.3 The string class and string streams</td>
<td>163</td>
</tr>
<tr>
<td>11.4 The <em>toString</em>( ) class member</td>
<td>165</td>
</tr>
<tr>
<td>11.5 The <em>printf</em> function</td>
<td>167</td>
</tr>
<tr>
<td>11.6 Assignments</td>
<td>167</td>
</tr>
<tr>
<td><strong>Part II Numerical analysis</strong></td>
<td></td>
</tr>
<tr>
<td>12 Numerical error analysis – derivatives</td>
<td></td>
</tr>
<tr>
<td>12.1 The derivative operator</td>
<td>175</td>
</tr>
<tr>
<td>12.2 Error dependence</td>
<td>177</td>
</tr>
<tr>
<td>12.3 Graphical error analysis</td>
<td>177</td>
</tr>
<tr>
<td>12.4 Analytic error analysis – higher-order methods</td>
<td>179</td>
</tr>
<tr>
<td>12.5 Extrapolation</td>
<td>179</td>
</tr>
<tr>
<td>12.6 The derivative calculator class</td>
<td>180</td>
</tr>
<tr>
<td>12.7 Assignments</td>
<td>182</td>
</tr>
<tr>
<td>13 Integration</td>
<td></td>
</tr>
<tr>
<td>13.1 Discretization procedures</td>
<td>183</td>
</tr>
<tr>
<td>13.2 Implementation</td>
<td>184</td>
</tr>
<tr>
<td>13.3 Discretization error</td>
<td>188</td>
</tr>
<tr>
<td>13.4 Assignments</td>
<td>189</td>
</tr>
<tr>
<td>14 Root-finding procedures</td>
<td></td>
</tr>
<tr>
<td>14.1 Bisection method</td>
<td>191</td>
</tr>
<tr>
<td>14.2 Newton’s method</td>
<td>193</td>
</tr>
<tr>
<td>14.3 Assignments</td>
<td>194</td>
</tr>
<tr>
<td>15 Differential equations</td>
<td></td>
</tr>
<tr>
<td>15.1 Euler’s method</td>
<td>196</td>
</tr>
<tr>
<td>15.2 Error analysis</td>
<td>198</td>
</tr>
<tr>
<td>15.3 The spring class</td>
<td>199</td>
</tr>
<tr>
<td>15.4 Assignments</td>
<td>201</td>
</tr>
<tr>
<td>16 Linear algebra</td>
<td></td>
</tr>
<tr>
<td>16.1 Linear equation solvers</td>
<td>203</td>
</tr>
</tbody>
</table>
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>16.2</td>
<td>Errors and condition numbers</td>
<td>206</td>
</tr>
<tr>
<td>16.3</td>
<td>Eigenvalues and iterative eigenvalue solvers</td>
<td>207</td>
</tr>
<tr>
<td>16.4</td>
<td>Assignments</td>
<td>209</td>
</tr>
<tr>
<td><strong>Part III</strong></td>
<td><strong>Advanced object-oriented programming</strong></td>
<td>215</td>
</tr>
<tr>
<td>17</td>
<td>References</td>
<td>217</td>
</tr>
<tr>
<td>17.1</td>
<td>Basic properties</td>
<td>217</td>
</tr>
<tr>
<td>17.2</td>
<td>References as function arguments</td>
<td>218</td>
</tr>
<tr>
<td>17.3</td>
<td>Reference member variables</td>
<td>219</td>
</tr>
<tr>
<td>17.4</td>
<td><code>const</code> reference variables</td>
<td>220</td>
</tr>
<tr>
<td>17.5</td>
<td>Reference return values</td>
<td>221</td>
</tr>
<tr>
<td>17.6</td>
<td>Assignments</td>
<td>222</td>
</tr>
<tr>
<td>18</td>
<td>Pointers and dynamic memory allocation</td>
<td>227</td>
</tr>
<tr>
<td>18.1</td>
<td>Introduction to pointers</td>
<td>228</td>
</tr>
<tr>
<td>18.2</td>
<td>Initializing pointer variables</td>
<td>228</td>
</tr>
<tr>
<td>18.3</td>
<td>The address-of and dereferencing operators</td>
<td>229</td>
</tr>
<tr>
<td>18.4</td>
<td>Uninitialized pointer errors</td>
<td>230</td>
</tr>
<tr>
<td>18.5</td>
<td>NULL and void pointers</td>
<td>230</td>
</tr>
<tr>
<td>18.6</td>
<td>Dangling pointers</td>
<td>231</td>
</tr>
<tr>
<td>18.7</td>
<td>Pointers in function blocks</td>
<td>232</td>
</tr>
<tr>
<td>18.8</td>
<td>The <code>const</code> keyword and pointers</td>
<td>232</td>
</tr>
<tr>
<td>18.9</td>
<td>Pointer arithmetic</td>
<td>234</td>
</tr>
<tr>
<td>18.10</td>
<td>Pointers and arrays</td>
<td>234</td>
</tr>
<tr>
<td>18.11</td>
<td>Pointer comparisons</td>
<td>234</td>
</tr>
<tr>
<td>18.12</td>
<td>Pointers to pointers and matrices</td>
<td>235</td>
</tr>
<tr>
<td>18.13</td>
<td>String manipulation</td>
<td>235</td>
</tr>
<tr>
<td>18.14</td>
<td>Static and dynamic memory allocation</td>
<td>237</td>
</tr>
<tr>
<td>18.15</td>
<td>Memory leaks and dangling pointers</td>
<td>239</td>
</tr>
<tr>
<td>18.16</td>
<td>Dynamic memory allocation within functions</td>
<td>241</td>
</tr>
<tr>
<td>18.17</td>
<td>Dynamically allocated matrices</td>
<td>242</td>
</tr>
<tr>
<td>18.18</td>
<td>Dynamically allocated matrices as function arguments and parameters</td>
<td>243</td>
</tr>
<tr>
<td>18.19</td>
<td>Pointer data structures and linked lists</td>
<td>244</td>
</tr>
<tr>
<td>18.20</td>
<td>Assignments</td>
<td>248</td>
</tr>
<tr>
<td>19</td>
<td>Advanced memory management</td>
<td>261</td>
</tr>
<tr>
<td>19.1</td>
<td>The <code>this</code> pointer</td>
<td>261</td>
</tr>
<tr>
<td>19.2</td>
<td>The <code>friend</code> keyword</td>
<td>262</td>
</tr>
<tr>
<td>19.3</td>
<td>Operators</td>
<td>263</td>
</tr>
<tr>
<td>19.4</td>
<td>Destructors</td>
<td>265</td>
</tr>
</tbody>
</table>
19.5 Assignment operators 267
19.6 Copy constructors 269
19.7 Assignments 271

20 The static keyword, multiple and virtual inheritance, templates, and the STL library 286
20.1 Static variables 286
20.2 Static class members 287
20.3 Pointer to class members 288
20.4 Multiple inheritance 288
20.5 Virtual functions 289
20.6 Heterogeneous object collections and runtime type identification 291
20.7 Abstract base classes and interfaces 292
20.8 Virtual inheritance 293
20.9 User-defined conversions 294
20.10 Function templates 295
20.11 Templates and classes 296
20.12 The complex class 298
20.13 The standard template library 299
20.14 Structures and unions 303
20.15 Bit fields and operators 305
20.16 Assignments 306

21 Program optimization in C++ 319
21.1 Compiling 319
21.2 Critical code segments 319
21.3 Virtual functions 321
21.4 Function pointers and functors 326
21.5 Aliasing 327
21.6 High-performance template libraries 327
21.7 Assignments 329

Part IV Scientific programming examples 331

22 Monte Carlo methods 333
22.1 Monte Carlo integration 333
22.2 Monte Carlo evaluation of distribution functions 334
22.3 Importance sampling 339
22.4 The metropolis algorithm 343
22.5 Multicanonical methods 347
22.6 Assignments 352
# Contents

### 23 Parabolic partial differential equation solvers
- 23.1 Partial differential equations in scientific applications
- 23.2 Direct solution methods
- 23.3 The Crank–Nicholson method
- 23.4 Assignments

### Appendix A Overview of MATLAB

### Appendix B The Borland C++ Compiler
- B.1 Borland C++ installation
- B.2 Compiling and running a first program
- B.3 Installing the optional program editor
- B.4 Using the Borland turbo debugger
- B.5 Installing DISLIN
- B.6 A first graphics program
- B.7 The help system

### Appendix C The Linux/Windows Command-Line C++ Compiler and Profiler

### Appendix D Calling FORTRAN programs from C++

### Appendix E C++ coding standard
- E.1 Program design language
- E.2 Comments
- E.3 Layout
- E.4 Continuation lines
- E.5 Constants and literals
- E.6 Variables and definitions
- E.7 Functions
- E.8 Operators
- E.9 Classes
- E.10 Typedefs
- E.11 Macros
- E.12 Templates
- E.13 Control structures

### References and further reading

### Index