

---

# Contents

Introduction	xvii
<b>1 Getting Started</b>	<b>1</b>
1.1 Installing your development environment	1
1.1.1 For Windows	1
1.1.2 For Unix	1
1.1.3 For MacOS X	1
1.2 Running an example program	2
1.3 Compiling and running the code	3
1.3.1 Compiling on Windows	4
1.3.2 Compiling on Unix	6
1.4 Understanding the example code	8
1.5 Configuring the compiler	12
1.6 Making decisions	13
1.7 Exercises	14
1.8 Summary	15
<b>2 Basic Data Types and Operators</b>	<b>17</b>
2.1 Memory terminology	17
2.2 Basic data types	18
2.2.1 Integers	18
2.2.2 Floating point numbers	20
2.2.3 Booleans	20
2.2.4 Characters	20
2.3 Casting	22
2.4 Memory addresses	26
2.5 Operators	28
2.5.1 The <code>sizeof</code> operator	28
2.5.2 Mathematical operations	28
2.5.3 Comparison operators	29
2.5.4 Logical operators	29
2.5.5 Bitwise operators	29
2.5.6 Combining operators	30
2.5.7 Assignment operators	30

2.5.8	If statements revisited . . . . .	32
2.6	Summary . . . . .	35
<b>3</b>	<b>Functions</b>	<b>37</b>
3.1	The C++ function syntax . . . . .	37
3.2	Recursion . . . . .	41
3.3	Libraries . . . . .	42
3.4	Declaring and defining functions . . . . .	42
3.5	Functions that don't return a value . . . . .	44
3.6	Specifying default values . . . . .	45
3.7	Overloading functions . . . . .	46
3.8	Global and local variables . . . . .	47
3.9	Namespaces . . . . .	48
3.10	Summary . . . . .	52
<b>4</b>	<b>Flow of Control</b>	<b>55</b>
4.1	while loops . . . . .	55
4.2	do-while loops . . . . .	57
4.3	for loops . . . . .	58
4.4	break, continue, return . . . . .	60
4.5	throw statements . . . . .	61
4.6	switch statements . . . . .	63
4.7	Scope . . . . .	65
4.8	Flow of control in operators . . . . .	65
4.8.1	Short circuit evaluation . . . . .	66
4.8.2	The ternary operator . . . . .	66
4.8.3	The comma operator . . . . .	67
4.9	Summary . . . . .	69
<b>5</b>	<b>Working with Multiple Files</b>	<b>71</b>
5.1	The project FMLib . . . . .	71
5.2	Header files . . . . .	72
5.3	Creating our project . . . . .	73
5.3.1	Creating the first header file . . . . .	73
5.3.2	Some code that uses the functions . . . . .	75
5.3.3	Write the definitions . . . . .	76
5.4	How header files work . . . . .	77
5.4.1	The meaning of include . . . . .	77
5.4.2	Pragma once . . . . .	77
5.4.3	Information hiding . . . . .	78
5.4.4	Inline . . . . .	80
5.5	A complete example . . . . .	81

5.6	Summary . . . . .	82
<b>6</b>	<b>Unit Testing</b>	<b>85</b>
6.1	A testing framework for C++ . . . . .	86
6.2	Macros . . . . .	86
6.3	The macros in <code>testing.h</code> . . . . .	87
6.3.1	The <code>ASSERT</code> macro . . . . .	87
6.3.2	The <code>ASSERT_APPROX_EQUAL</code> macro . . . . .	87
6.3.3	The <code>INFO</code> macro . . . . .	88
6.3.4	The <code>DEBUG_PRINT</code> macro . . . . .	88
6.3.5	The <code>TEST</code> macro . . . . .	89
6.4	Using <code>testing.h</code> . . . . .	89
6.5	What have we gained? . . . . .	91
6.6	Testing <code>normcdf</code> . . . . .	92
6.7	Summary . . . . .	94
<b>7</b>	<b>Using C++ Classes</b>	<b>97</b>
7.1	Vectors . . . . .	97
7.2	Pass by reference and <code>const</code> . . . . .	100
7.2.1	Pass by reference . . . . .	101
7.2.2	The <code>const</code> keyword . . . . .	102
7.2.3	Pass by reference without <code>const</code> . . . . .	104
7.3	Using <code>ofstream</code> . . . . .	104
7.4	Working with <code>string</code> . . . . .	106
7.5	Building strings efficiently . . . . .	107
7.6	Writing a pie chart . . . . .	108
7.6.1	A web-based chart . . . . .	109
7.6.2	Create a header file . . . . .	111
7.6.3	Write a source file . . . . .	112
7.6.4	Enable testing in your files . . . . .	112
7.6.5	Write functions to generate the boiler plate . . . . .	112
7.6.6	Write a simple version of the chart data . . . . .	113
7.6.7	Write a test of what we've done so far . . . . .	114
7.6.8	Write the interesting code . . . . .	114
7.6.9	Testing the interesting code . . . . .	115
7.6.10	Wrap it all up into a single function . . . . .	116
7.7	The architecture of the World Wide Web . . . . .	117
7.8	Summary . . . . .	121
<b>8</b>	<b>User-Defined Types</b>	<b>123</b>
8.1	Terminology . . . . .	123
8.2	Writing your own class . . . . .	124

8.2.1	Writing the declaration . . . . .	124
8.2.2	Using a class . . . . .	126
8.2.3	Passing objects between functions . . . . .	127
8.2.4	How have classes helped? . . . . .	127
8.3	Adding functions to classes . . . . .	128
8.3.1	Using const on member functions . . . . .	130
8.4	A financial example . . . . .	131
8.4.1	What have we gained? . . . . .	133
8.5	Recommendations on writing classes . . . . .	134
8.6	Encapsulation . . . . .	135
8.6.1	Implementing PieChart . . . . .	137
8.6.2	Using PieChart . . . . .	137
8.7	Constructors . . . . .	138
8.7.1	Writing a default constructor . . . . .	139
8.7.2	An alternative, and superior syntax . . . . .	140
8.8	Constructors with parameters . . . . .	141
8.9	Summary . . . . .	144
<b>9</b>	<b>Monte Carlo Pricing in C++</b>	<b>145</b>
9.1	A function to simulate stock prices . . . . .	146
9.2	Writing a Monte Carlo pricer . . . . .	151
9.3	Generating random numbers for Monte Carlo . . . . .	154
9.4	Summary . . . . .	158
<b>10</b>	<b>Interfaces</b>	<b>159</b>
10.1	An interface for pricing options . . . . .	159
10.2	Describing an interface in C++ . . . . .	161
10.3	Examples of interfaces . . . . .	164
10.4	Interfaces in object-oriented programming . . . . .	166
10.5	What's wrong with if statements? . . . . .	168
10.6	An interface for integration . . . . .	169
10.7	Summary . . . . .	173
<b>11</b>	<b>Arrays, Strings, and Pointers</b>	<b>175</b>
11.1	Arrays, the C alternative to vector . . . . .	176
11.2	Pointers . . . . .	179
11.2.1	new and delete . . . . .	179
11.2.2	Pointer operators . . . . .	180
11.2.3	Looping with pointers . . . . .	182
11.2.4	Using pointers in practice . . . . .	185
11.3	Pointers to text . . . . .	185
11.4	Pass by pointer . . . . .	187

11.5	Don't return pointers to local variables . . . . .	189
11.6	Using pointers to share data . . . . .	191
11.6.1	Sharing with <code>shared_ptr</code> . . . . .	194
11.7	Sharing data with references . . . . .	197
11.8	The C++ memory model . . . . .	199
11.8.1	The stack . . . . .	200
11.8.2	The heap . . . . .	202
11.9	Summary . . . . .	204
<b>12</b>	<b>More Sophisticated Classes</b>	<b>205</b>
12.1	Inlining member functions . . . . .	205
12.2	The <code>this</code> keyword . . . . .	206
12.3	Inheritance . . . . .	207
12.3.1	What have we gained? . . . . .	209
12.3.2	Terminology . . . . .	209
12.4	Overriding methods — the <code>virtual</code> keyword . . . . .	210
12.4.1	A note on the keyword <code>virtual</code> . . . . .	211
12.5	Abstract functions <code>=0</code> . . . . .	212
12.6	Multiple layers . . . . .	212
12.6.1	UML . . . . .	213
12.6.2	Another object hierarchy . . . . .	215
12.6.3	Multiple inheritance . . . . .	215
12.6.4	Calling superclass methods . . . . .	216
12.7	Forward declarations and the structure of <code>cpp</code> files . . . . .	217
12.8	The <code>static</code> keyword . . . . .	218
12.9	The <code>protected</code> keyword . . . . .	220
12.10	Summary . . . . .	222
<b>13</b>	<b>The Portfolio Class</b>	<b>223</b>
13.1	The <code>Priceable</code> interface . . . . .	223
13.2	The <code>Portfolio</code> interface and implementation . . . . .	224
13.2.1	Implementation of <code>PortfolioImpl</code> . . . . .	227
13.3	Testing . . . . .	228
13.4	UML . . . . .	230
13.5	Limitations . . . . .	231
13.6	Summary . . . . .	232
<b>14</b>	<b>Delta Hedging</b>	<b>233</b>
14.1	Discrete-time delta hedging . . . . .	233
14.2	Implementing the delta hedging strategy in C++ . . . . .	235
14.2.1	Class declaration . . . . .	235
14.2.2	Implementation of <code>runSimulation</code> . . . . .	237

14.2.3	Implementing the other methods of HedgingSimulator	238
14.2.4	Changes to CallOption	240
14.3	Testing the simulation	241
14.4	Interpreting and extending our simulation	241
14.5	Summary	244
<b>15</b>	<b>Debugging and Development Tools</b>	<b>245</b>
15.1	Debugging strategies	245
15.1.1	Unit tests	245
15.1.2	Reading your code	246
15.1.3	Logging statements	246
15.1.4	Using a debugger	247
15.1.5	Divide and conquer	247
15.2	Debugging with Visual Studio	248
15.2.1	Obtaining a stack trace in Visual Studio	248
15.2.2	Breakpoints and single stepping in Visual Studio	250
15.3	Debugging with GDB	252
15.3.1	Using GDB to obtain a stack trace	253
15.3.2	Breakpoints and single stepping with GDB	256
15.3.3	Other commands and features	257
15.4	Other development tools and practices	258
15.4.1	Version control	258
15.4.2	Bug tracking	259
15.4.3	Testing framework	259
15.4.4	Automated build	260
15.4.5	Continuous integration	261
15.4.6	Logging	261
15.4.7	Static analysis	261
15.4.8	Memory-leak detection	262
15.4.9	Profiling tools	262
15.4.10	Example	263
15.5	Summary	264
<b>16</b>	<b>A Matrix Class</b>	<b>267</b>
16.1	Basic functionality of Matrix	267
16.2	The constructor and destructor of Matrix	269
16.2.1	Virtual destructors	271
16.2.2	When is a destructor needed?	272
16.2.3	Additional constructors	273
16.3	Const pointers	274
16.4	Operator overloading	275
16.4.1	Overloading +	275
16.4.2	Overloading other arithmetic operators	277

16.4.3	Overloading comparison operators . . . . .	278
16.4.4	Overloading the << operator . . . . .	279
16.4.4.1	Remarks on return by reference . . . . .	280
16.4.5	Overloading the () operator . . . . .	280
16.4.6	Overloading += . . . . .	281
16.5	The rule of three . . . . .	282
16.5.1	Overriding the assignment operator . . . . .	282
16.5.2	Writing a copy constructor . . . . .	283
16.5.3	The easy way to abide by the rule of three . . . . .	284
16.5.4	Move operators . . . . .	285
16.6	Completing the Matrix class . . . . .	285
16.7	Array Programming . . . . .	286
16.7.1	Implementing an efficient matrix class . . . . .	286
16.7.2	Array programming . . . . .	287
16.7.3	Array programming in the option classes . . . . .	288
16.7.4	Array programming for the BlackScholesModel . . . . .	289
16.7.5	Array programming the Monte Carlo pricer . . . . .	290
16.7.6	Performance . . . . .	290
16.8	Summary . . . . .	292
<b>17 An Overview of Templates</b>		<b>295</b>
17.1	Template functions . . . . .	295
17.2	Template classes . . . . .	297
17.3	Templates as an alternative to interfaces . . . . .	299
17.4	Summary . . . . .	302
<b>18 The Standard Template Library</b>		<b>303</b>
18.1	<code>typedef</code> . . . . .	304
18.2	<code>auto</code> . . . . .	306
18.3	Using iterators with vectors . . . . .	307
18.4	<code>for</code> loops and containers . . . . .	309
18.5	The container <code>set</code> . . . . .	310
18.6	The container <code>vector</code> . . . . .	311
18.7	The container <code>list</code> . . . . .	312
18.8	The container <code>initializer_list</code> . . . . .	315
18.9	The containers <code>map</code> and <code>unordered_map</code> . . . . .	315
18.9.1	How a <code>map</code> works . . . . .	317
18.9.2	How an <code>unordered_map</code> works . . . . .	318
18.10	Storing complex types in containers . . . . .	320
18.11	A mathematical model for multiple stocks . . . . .	320
18.12	Using the Standard Template Library in FMLib . . . . .	322
18.13	Summary . . . . .	327

<b>19</b>	<b>Function Objects and Lambda Functions</b>	<b>329</b>
19.1	Function objects . . . . .	329
19.2	Lambda functions . . . . .	330
19.3	Function pointers . . . . .	333
19.4	Sorting with lambda functions . . . . .	334
19.5	Summary . . . . .	336
<b>20</b>	<b>Threads</b>	<b>337</b>
20.1	Concurrent programming in C++ . . . . .	338
20.1.1	Creating threads . . . . .	338
20.1.2	Mutual exclusion . . . . .	339
20.1.3	Global variables and race conditions . . . . .	342
20.1.4	Problems with locking . . . . .	343
20.2	The command design pattern . . . . .	346
20.3	Monte Carlo pricing . . . . .	347
20.3.1	Random number generation with multiple threads . . . . .	348
20.3.2	A multi-threaded pricer . . . . .	349
20.3.3	Implementing <code>Task</code> . . . . .	350
20.3.4	Using the <code>Executor</code> . . . . .	351
20.3.5	Remarks upon the design . . . . .	351
20.4	Coordinating threads . . . . .	352
20.4.1	The Pipeline pattern . . . . .	352
20.4.2	How <code>Pipeline</code> is implemented . . . . .	355
20.5	Summary . . . . .	358
<b>21</b>	<b>Next Steps</b>	<b>359</b>
21.1	Programming . . . . .	359
21.1.1	Libraries . . . . .	359
21.1.2	Software development . . . . .	359
21.1.3	C++ language features . . . . .	360
21.1.4	Other languages . . . . .	360
21.2	Financial mathematics . . . . .	361
<b>A</b>	<b>Risk-Neutral Pricing</b>	<b>363</b>
A.1	The players in financial markets . . . . .	363
A.2	Derivatives contracts . . . . .	366
A.3	Risk-neutral pricing . . . . .	370
A.4	Modelling stock prices . . . . .	372
A.5	Monte Carlo pricing . . . . .	377
A.6	Hedging . . . . .	379
A.7	Summary . . . . .	382



**Bibliography**

**383**

**Index**

**385**