

Contents

0	Goals of this Book and Global Overview	1
0.1	What is this book?	1
0.2	Why has this book been written?	1
0.3	For whom is this book intended?	2
0.4	Why should I read this book?	2
0.5	The structure of this book	2
0.6	What this book does not cover	2
0.7	More information and support	3
PART I C++ ESSENTIAL SKILLS		5
1	Introduction to C++ and Quantitative Finance	7
1.1	Introduction and objectives	7
1.2	A short history of C++	7
1.3	C++, a multi-paradigm language	8
1.3.1	Object-oriented paradigm	8
1.3.2	Generic programming	10
1.3.3	Procedural, modular and functional programming	11
1.4	C++ and quantitative finance: what's the relationship?	11
1.5	What is software quality?	11
1.6	Summary and conclusions	13
1.7	Exercises	14
2	The Mechanics of C++: from Source Code to a Running Program	15
2.1	Introduction and objectives	15
2.2	The compilation process	15
2.3	Header files and source files	16
2.4	Creating classes and using their objects	19
2.5	Template classes and template functions	22
2.6	Kinds of errors	25
2.6.1	Compiler errors	25
2.6.2	Linker errors	26
2.6.3	Run-time errors	26
2.7	The struct concept	27

2.8	Useful data conversion routines	27
2.9	Summary and conclusions	29
2.10	Exercises and projects	29
3	C++ Fundamentals and My First Option Class	31
3.1	Introduction and objectives	31
3.2	Class == member data + member functions	32
3.3	The header file (function prototypes)	33
3.4	The class body (code file)	35
3.5	Using the class	38
3.6	Examining the class in detail	40
3.6.1	Accessibility issues	40
3.6.2	Using standard libraries	40
3.6.3	The scope resolution operator '::'	41
3.6.4	Virtual destructor: better safe than sorry	41
3.7	Other paradigms	41
3.8	Summary and conclusions	45
3.9	Questions, exercises and projects	45
4	Creating Robust Classes	49
4.1	Introduction and objectives	49
4.2	Call by reference and call by value	49
4.3	Constant objects everywhere	52
4.3.1	Read-only (const) member functions	52
4.4	Constructors in detail	54
4.4.1	Member initialisation	55
4.5	Static member data and static member functions	55
4.6	Function overloading	57
4.7	Non-member functions	58
4.8	Performance tips and guidelines	58
4.8.1	The 'inline' keyword	58
4.8.2	Anonymous objects in function code	59
4.8.3	Loop optimisation	60
4.9	Summary and conclusions	60
4.10	Questions, exercises and projects	60
5	Operator Overloading in C++	63
5.1	Introduction and objectives	63
5.2	What is operator overloading and what are the possibilities?	63
5.3	Why use operator overloading? The advantages	65
5.4	Operator overloading: the steps	68
5.4.1	A special case: the assignment operator	70
5.5	Using operator overloading for simple I/O	71
5.6	Friend functions in general	72
5.6.1	Friend classes	73
5.7	Summary and conclusions	74
5.8	Exercise	74
	Appendix: useful data structures in C++	75

6	Memory Management in C++	79
6.1	Introduction and objectives	79
6.2	Single objects and arrays of objects on the stack	79
6.3	Special operators: 'new' and 'delete'	82
6.3.1	Single objects	82
6.3.2	Arrays of objects	83
6.4	Small application: working with complex numbers	84
6.5	Creating an array class	86
6.5.1	Memory allocation and deallocation	86
6.5.2	Accessing functions	87
6.5.3	Examples	88
6.5.4	The full header file	88
6.6	Summary and conclusions	89
6.7	Exercises	89
6.8	Review questions and comments	91
7	Functions, Namespaces and Introduction to Inheritance	93
7.1	Introduction and objectives	93
7.2	Functions and function pointers	93
7.2.1	Functions in financial engineering	94
7.2.2	Function categories	94
7.2.3	Modelling functions in C++	95
7.2.4	Application areas for function pointers, part I	96
7.3	An introduction to namespaces in C++	96
7.3.1	Some extra functionality	97
7.4	An introduction to the inheritance mechanism in C++	99
7.4.1	Basic inheritance structure	99
7.4.2	Adding new functionality	101
7.4.3	Overriding functionality: polymorphic and non-polymorphic behaviour	102
7.5	Multiple inheritance	104
7.6	Solution of nonlinear equations	104
7.7	Nonlinear solvers in C++: design and implementation	106
7.8	Applying nonlinear solvers: calculating volatility	108
7.9	Summary and conclusions	109
7.10	Exercises and projects	109
8	Advanced Inheritance and Payoff Class Hierarchies	113
8.1	Introduction and objectives	113
8.2	The virtual specifier and memory deallocation	113
8.3	Abstract and concrete classes	115
8.3.1	Using payoff classes	118
8.4	Lightweight payoff classes	119
8.5	Super lightweight payoff functions	121
8.6	The dangers of inheritance: a counterexample	123
8.7	Implementation inheritance and fragile base class problem	127
8.7.1	Deep hierarchies	127
8.7.2	Multiple inheritance problems	127

8.8	Two-factor payoff functions and classes	127
8.9	Conclusions and summary	130
8.10	Exercises and projects	130
9	Run-Time Behaviour in C++	133
9.1	Introduction and objectives	133
9.2	An introduction to reflection and self-aware objects	133
9.3	Run-time type information (RTTI)	134
9.4	Casting between types	137
9.4.1	More esoteric casting	139
9.5	Client-server programming and exception handling	140
9.6	try, throw and catch: ingredients of the C++ exception mechanism	141
9.7	C++ implementation	141
9.8	Pragmatic exception mechanisms	145
9.8.1	An example of use	148
9.9	Conclusions and summary	149
9.10	Exercises and research	149
10	An Introduction to C++ Templates	153
10.1	Introduction and objectives	153
10.2	My first template class	154
10.2.1	Using template classes	156
10.2.2	(Massive) reusability of the range template class	157
10.3	Template functions	158
10.4	Consolidation: understanding templates	159
10.4.1	A test program	162
10.5	Summary and conclusions	163
10.6	Exercises and projects	164
PART II	DATA STRUCTURES, TEMPLATES AND PATTERNS	167
11	Introduction to Generic Data Structures and Standard Template Library (STL)	169
11.1	Introduction and objectives	169
11.2	Complexity analysis	170
11.2.1	Examples of complexities	171
11.3	An introduction to data structures	172
11.3.1	Lists	172
11.3.2	Stacks and queues	173
11.3.3	Sets and multisets	174
11.3.4	Maps and multimaps	176
11.4	Algorithms	176
11.5	Navigation in data structures: iterators in STL	177
11.6	STL by example: my first example	178
11.6.1	Constructors and memory allocation	179
11.6.2	Iterators	179
11.6.3	Algorithms	179

11.7	Conclusions and summary	183
11.8	Exercises and projects	183
12	Creating Simpler Interfaces to STL for QF Applications	187
12.1	Introduction and objectives	187
12.2	Maps and dictionaries	187
12.2.1	Iterating in maps	188
12.2.2	Erasing records in a map	189
12.3	Applications of maps	190
12.4	User-friendly sets	191
12.4.1	STL sets	192
12.4.2	User-defined and wrapper classes for STL sets	194
12.5	Associative arrays and associative matrices	196
12.6	Applications of associative data structures	199
12.7	Conclusions and summary	200
12.8	Exercises and projects	200
13	Data Structures for Financial Engineering Applications	203
13.1	Introduction and objectives	203
13.2	The property pattern	203
13.3	Property sets	205
13.3.1	Basic functionality	206
13.3.2	Addition and removal of properties	207
13.3.3	Navigating in a property set: creating your own iterators	208
13.3.4	Property sets with heterogeneous data types	209
13.4	Property sets and data modelling for quantitative finance	213
13.4.1	Fixed assemblies (assemblies-parts relationship)	213
13.4.2	Collection-members relationship	213
13.4.3	Container-contents relationship	215
13.4.4	Recursive and nested property sets	215
13.5	Lattice structures	215
13.5.1	Some simple examples	217
13.5.2	A simple binomial method solver	219
13.6	Conclusions and summary	221
13.7	Exercises and projects	221
14	An Introduction to Design Patterns	223
14.1	Introduction and objectives	223
14.2	The software lifecycle	223
14.3	Documentation issues	225
14.3.1	Generalisation and specialisation	225
14.3.2	Aggregation and composition	227
14.3.3	Associations	229
14.3.4	Other diagrams	232
14.4	An Introduction to design patterns	233
14.4.1	Creational patterns	233
14.4.2	Structural patterns	236
14.4.3	Behavioural patterns	237

14.5	Are we using the wrong design? Choosing the appropriate pattern	237
14.6	CADObject, a C++ library for computer graphics	238
14.7	Using patterns in CADObject	240
14.8	Conclusions and summary	241
14.9	Exercises and projects	242
PART III	QF APPLICATIONS	243
15	Programming the Binomial Method in C++	245
15.1	Introduction and objectives	245
15.2	Scoping the problem	246
15.3	A short overview of the binomial method	246
15.4	Software requirements for a binomial solver	249
15.5	Class design and class structure	250
15.5.1	UML class structure	250
15.5.2	Information flow	251
15.6	Applying design patterns	252
15.7	The builder and director classes	255
15.8	The process and the steps	258
15.9	Test cases and examples	259
15.10	Conclusions and summary	260
15.11	Exercises and questions	260
16	Implementing One-Factor Black Scholes in C++	265
16.1	Introduction and objectives	265
16.2	Scope and assumptions	265
16.3	Assembling the C++ building blocks	267
16.4	Modelling the black scholes PDE	267
16.4.1	Creating your own one-factor financial models	270
16.5	Finite difference schemes	270
16.5.1	Explicit schemes	272
16.5.2	Implicit schemes	273
16.5.3	A note on exception handling	275
16.5.4	Other schemes	275
16.6	Test cases and presentation in excel	276
16.6.1	Creating the user interface dialogue	276
16.6.2	Vector and matrix output	278
16.6.3	Presentation	278
16.7	Summary	279
16.8	Exercises and projects	279
17	Two-Factor Option Pricing: Basket and Other Multi-Asset Options	283
17.1	Introduction and objectives	283
17.2	Motivation and background	283
17.3	Scoping the problem: PDEs for basket options	285
17.4	Modelling basket option PDE in UML and C++	286
17.4.1	Data classes for instruments	286
17.4.2	Modelling the PDE with C++ classes	292

17.5	The finite difference method for two-factor problems	297
17.6	Discrete boundary and initial conditions	298
17.7	Assembling the system of equations	299
17.8	Post processing and output	301
17.9	Summary and conclusions	302
17.10	Exercises and projects	302
18	Useful C++ Classes for Numerical Analysis Applications in Finance	305
18.1	Introduction and objectives	305
18.2	Solving tridiagonal systems	305
18.2.1	A tridiagonal solver in C++	306
18.3	An introduction to interpolation	310
18.3.1	Polynomial interpolation	310
18.3.2	Rational function interpolation	311
18.3.3	Cubic spline interpolation	312
18.4	Summary and conclusions	313
19	Other Numerical Methods in Quantitative Finance	315
19.1	Introduction and objectives	315
19.2	The trinomial method for assets	315
19.3	Lattice data structures	318
19.4	Trinomial tree for the short rate	318
19.5	The multidimensional binomial method	321
19.6	Generic lattice structures	322
19.6.1	Candlestick charts	322
19.6.2	(Highly) generic lattice structures	324
19.7	Approximating exponential functions	326
19.8	Summary and conclusions	326
19.9	Exercises	326
20	The Monte Carlo Method Theory and C++ Frameworks	327
	<i>Dr. Joerg Kieritz and Daniel J. Duffy</i>	
20.1	Introduction and objectives	327
20.2	A short history of the Monte Carlo (MC) method	327
20.3	Examples of the application of the Monte Carlo method	327
20.3.1	Calculation of definite integrals	328
20.3.2	Device reliability	330
20.4	The Monte Carlo method in quantitative finance	333
20.4.1	An overview of stochastic differential equations (SDE)	333
20.4.2	Other background information	336
20.5	Software architecture for the Monte Carlo method	337
20.6	Examples and test cases	338
20.6.1	Plain options	338
20.6.2	Barrier options	338
20.6.3	Asian options	339
20.7	Summary and conclusions	340
20.8	Appendix: comparing Monte Carlo with other numerical methods	340
20.9	Exercises and projects	341

21 Skills Development: from White Belt to Black Belt	345
21.1 Introduction and objectives	345
21.2 Review of book	345
21.3 Part I: C++ essential skills	345
21.4 Part II: data structures, templates and patterns	346
21.5 Part III: applications in quantitative finance	347
21.6 Part IV: background information	347
21.7 Choosing a programming paradigm	348
21.7.1 Layer 1: foundation classes	348
21.7.2 Layer 2: mechanisms	348
21.7.3 Layer 3: building blocks	349
21.7.4 Layer 4: application level	349
21.8 Summary and conclusions	349
PART IV BACKGROUND INFORMATION	351
22 Basic C Survival Guide	353
22.1 Introduction and objectives	353
22.2 Basic data types	353
22.3 The C preprocessor	354
22.4 Pointers and references	355
22.5 Other useful bits and pieces	357
22.6 What to avoid in C	357
22.7 Test case: numerical integration in one dimension	358
22.8 Conclusions and summary	359
22.9 Exercises	359
23 Advanced C Syntax	363
23.1 Introduction and objectives	363
23.2 Fixed-size and dynamic arrays	364
23.3 Multi-dimensional arrays	364
23.4 Introduction to structures	365
23.5 Unions	366
23.6 Useful C libraries	366
23.7 Test case: linear regression	367
23.8 Conclusions and summary	370
23.9 Exercises	370
24 Datasim Visualisation Package in Excel: Drivers and Mechanisms	373
24.1 Introduction and objectives	373
24.2 Fundamental functionality in the package	373
24.3 Basic driver functionality	373
24.4 Excel mechanisms	376
24.5 Option values and sensitivities in excel	380
24.6 Finite difference method	384
24.7 Summary and conclusions	387
24.8 Exercises and projects	387

	Contents	xiii
25 Motivating COM and Emulation in C++		391
25.1 Introduction and objectives		391
25.2 A short history of COM		391
25.3 An introduction to multiple inheritance (MI)		391
25.4 Interfaces and multiple inheritance		396
25.5 Virtual function tables		399
25.6 Summary		399
26 COM Fundamentals		401
26.1 Introduction and objectives		401
26.2 Interfaces in COM		401
26.3 The IUnknown interface		401
26.4 Using IUnknown		402
26.4.1 Testing the interface		405
26.5 Defining new versions of components		406
26.6 Summary		406
References		407
Index		409

