

## Index

- 80–20 rule 60
- `#define` 16, 20–3, 42–5, 88–9, 145–6, 159–63, 204–5, 232–3, 306–7, 354–5, 357, 364, 396–7
- `#else` 355
- `#elsif` 355
- `#endif` 17–24, 43, 146, 205, 355
- `#ifdef` 355
- `#ifndef` 16, 20–3, 42–5, 88–9, 145–6, 159–63, 204–5, 306–7, 355
- `#include` 17–22, 25–6, 28–9, 35–6, 42–5, 88–9, 145–6, 163, 306–7, 354–5
- `&` 51, 55–7, 117–21
- `*` 64–5, 356–8
- `++` (increment operator) 64
- `+` (addition operator) 64–5
- `[]` 83–8
- `:` (constructor initialisation) 128–30
- `==` (equal comparison operator) 117–18
- `=` (assignment operator) 64–5, 70–1, 116–19
- `>>` (input operator) 64, 65, 71–2
- `<<` (output operator) 64, 65, 71–2
- `::` (scope resolution operator) 35, 41, 55–7, 68, 101–2, 116–18, 393–4
- `-` (subtraction operator) 64–5
- abstract classes
- COM interfaces 399
  - concepts 32–3, 115–19, 133–4, 210–13, 292–3, 380–1, 399
  - definition 115
- Abstract Factory pattern
- see also* creational patterns
  - concepts 234–42, 252–63, 267–82, 286–303
- accept 205
- accessibility issues, classes 33–4, 40–1
- accuracy characteristics, software quality 12–13, 223–5, 341
- ActiveX Template Library (ATL) 406
- adaptability characteristics, software quality 13
- adapter classes 173–4, 200, 236–42
- Adapter pattern 236–42
- see also* structural patterns
- add 207–8, 227–8
- Addins 373
- ADI *see* alternating direction implicit
- aggregation relationships
- see also* Whole-Part...
  - concepts 227–33, 238–42, 337, 348–9
- algorithms
- classes 171–2, 213–15
  - concepts 169–70, 172–85, 213–15, 241, 310–26
  - containers 169–70, 172–6, 178–83, 187–202, 213–15
  - efficiency issues 170–83, 299–300
  - interpolation 310–14
  - iterators 177–83
  - mutating/non-mutating algorithms 177
  - STL examples 179–83
- aliases, namespaces 98–9
- allocation/deallocation essentials, memory management 65, 79–92, 113–15, 245–63, 346–9, 353, 356–9
- alternating direction implicit (ADI) 283
- American options 250–1, 260–3, 281–2
- analysability characteristics, software quality 13
- analysis, software lifecycle 224–5
- annuities 42–7
- anonymous objects, optimization 59, 69–70
- antithetics 340, 343
- applications
- see also* quantitative finance; software activities 347
  - historical background 8
  - software layering 348–9
  - types 8–11, 347–9
- arbitrage 246

- Arithmetic Asian calls 339–40  
arithmetic operators 63–78  
*see also entries at start of index*  
list 63–5  
operator overloading 63–78  
**array** 75–8  
**arrays**  
*see also vectors*  
accessing functions 87–8, 196–202  
C 357, 363–71  
classes 86–92, 196–202, 364  
compile-time parametric arrays 81–2  
complex numbers 84–6  
concepts 75–8, 79–92, 136–7, 140, 153–4,  
178–83, 187–202, 226–7, 232–3, 299–303,  
306–14, 357, 363–71  
dynamic memory allocation 81–2, 83–6, 364  
lifecycle 363–71  
multi-dimensional arrays 363–71  
one-dimensional arrays 363–71  
pointers 79, 82–92, 136–7, 357  
range errors 80–1, 140–52, 200  
static arrays 80–6, 364  
templates 81–2, 196–202  
visualisation packages in Excel 373–89  
Asian options 339–40  
assemblies-parts relationships, property sets  
213–15  
**assert** 44–5  
assignment operators 33–4, 70–1, 116–19  
**AssocArray** 187–202  
association relationships, concepts 229–33,  
238–42, 348–9  
associative arrays, concepts 187–8, 196–202  
associative matrices  
concepts 176, 187–8, 196–202  
QF applications 199–200  
**AssocMatrix** 187–202  
asymptotic complexity  
*see also efficiency issues*  
concepts 170–83  
**AT&T** 7  
**ATL** *see* ActiveX Template Library  
**attributes**  
*see also member data*  
concepts 8–9, 31–2  
automatic memory  
*see also stack*  
concepts 55–7, 79–92, 364  
backward induction 220–1, 245–6, 255–63,  
316–18  
backward iterators 177–8  
*see also iterators*  
Backward in Time and Centred in Space (BTCS)  
273–4  
barrier options 127–30, 267–82, 338–40  
base classes  
*see also inheritance mechanism*  
concepts 99–111, 113–31, 134–7, 142–9,  
214–15, 225–33, 250–62, 270–82, 391–9  
fragile base class problem 127  
Orthogonal Base Class scenario 391–3  
basket options 128–30, 283–303, 321–2  
binomial method 321–2  
concepts 283–303, 321–2  
definition 283  
modeling 283–97  
PDEs 285–303  
post processing 301–2  
pricing 283–303, 321–2  
scoping the problem 285–6  
UML 286–7, 292–3  
bear spreads 118–19  
behavioural patterns  
CADOject 240–1  
concepts 223, 237–42, 252–62  
types 237  
best/worst options 128–30, 284–303  
betas 308–14  
‘big-O’ notation 170  
‘Bill of Materials’ 149–50  
binary operators, concepts 64–9, 182  
binomial method 31, 153, 158, 201, 203, 215–16,  
219–21, 245–63, 318, 321–2, 340–1, 347–9  
*see also lattice structures*  
basket options 321–2  
C++ 245–63, 347–9  
class design/structure 250–60  
classes 245–63  
comparisons with other methods 340–1  
concepts 31, 153, 158, 201, 203, 215–16,  
219–21, 245–63, 318, 321–2, 340–1, 347–9  
design patterns 252–62  
Monte Carlo simulation 340–1  
multidimensional binomial method 321–2  
overview 246–9  
performance issues 262–3, 340–1  
pricing 31, 153, 158, 201, 203, 215–16,  
219–21, 245–63, 321–2, 340–1, 347–9  
process steps 258–9  
simple solver 219–21, 249–63  
software requirements 249–50  
test cases 259–60  
Bisection method 93, 105–9  
Black-Scholes options pricing model 31–3, 37,  
246, 259–60, 265–82, 283–303, 309, 366,  
381–7  
*see also partial differential equations*  
assumptions 246, 265–70  
concepts 31–3, 37, 246, 259–60, 265–82,  
283–303, 309, 366, 381–7

- Excel presentation 276–8, 299, 301–2
- modeling 267–70
- test programs 276–8
- UML 265–7
- bonds 42–5, 175, 203, 209–10, 325
  - see also* fixed-income applications
  - yield curves 203
- Boolean algebra 65, 204–5, 231–2, 309–10
- bottom-up approaches 13, 353
- boundary conditions, Black-Scholes options
  - pricing model 266–82, 283–303, 374–7
- Boyle 333
- Bridge pattern 236–42, 267–82, 292–303
  - see also* structural patterns
- Brownian motion 246–9, 315–18, 334–6, 343
- BTCS *see* Backward in Time and Centred in Space
- Builder pattern 240–1, 255–62
- building blocks, software layering 348–9
- built-in types 23, 27–9, 71–2, 79–81, 353
  - creation 79–81
  - data conversions 27–9
- bull markets 322–3
- bull spreads 118–21
- butterfly spreads 118–19
- C# 9, 96
- C++
  - see also* object-oriented paradigm
  - advanced topics 373–406
  - binomial method 245–63, 347–9
  - Black-Scholes options pricing model 31–3, 37, 246, 259–60, 265–303, 309, 381–7
  - COM 27, 347–8, 373, 391–406
  - concepts 7–14, 15–30, 31–47, 55–7, 64–8, 106–9, 127, 203, 223–5, 345–9, 353, 391–9
  - critique 7–8, 11–12, 13, 57–8, 64–8, 203, 223–5, 345–9, 363–71, 391–9
  - Excel 373–89
  - fundamentals 31–47
  - historical background 7–8, 353
  - introduction 7–14
  - maintainability problems 13
  - mechanics 15–30
  - memory usage 55–7
  - Monte Carlo simulation 337–40, 347–9
  - multi-paradigm language 8–11
  - nonlinear solvers 106–9
  - numerical analysis applications in finance 305–14, 347–9
  - overview 345–9
  - QF relationship 11–14, 203, 223–5
  - roadmap 345–9
  - run-time behaviour 133–52, 346–9
  - skills' development 345–9, 359, 363–71
  - tridiagonal matrices 306–9
  - trinomial method 315–26, 347–9
- C 7, 27, 28–9, 82, 123, 158, 346, 347–9, 353–61, 363–71
  - advanced syntax 363–71
  - arrays 357, 363–71
  - basics 353–61, 363–71
  - concepts 353–61, 363–71
  - critique 357–9
  - data conversions 28–9
  - data types 353–4
  - K & R reference 353
  - legacy applications 28–9
  - libraries 366–71
  - linear regression 363, 367–70
  - math.h library 363, 366–7
  - multi-dimensional arrays 363–71
  - numerical integration 358–9
  - one-dimensional arrays 363–71
  - pointers 353, 355–6, 363–71
  - preprocessor 354–5
  - problem areas 357–8
  - references 355–6
  - struct concept 27, 357–8, 363, 365–70, 396–7
  - syntax 346, 347–9, 353–61, 363–71
  - test programs 358–9, 367–70
  - time.h library 367
  - union 357–8, 366
- CAD *see* Computer Aided Design
- CADObject 223, 238–42
- call by reference, concepts 49–51, 55
- call by value, concepts 49–51
- call options 9, 31, 33–4, 39–40, 108, 117–21, 234–42, 246–63, 333–43
- candlestick charts 322–5
- canonical header files, concepts 116–19
- Cartesian space 52
- cash-flows 66–7
- casting
  - see also* dynamic...; static...
  - concepts 137–52, 212–13, 346–9
  - types 137
- catch 144–52, 302, 378–9
- catch block, exception-handling mechanism 141–52, 302, 378–9
- Cayley map 262–3, 280, 326
- CD 43, 57, 75, 86, 108, 223, 232, 240–1, 274, 285, 308, 309, 313–14, 315, 318, 323, 336–7, 349, 364, 376, 384, 389
- ceil 367
- changeability characteristics, software quality 13
- char 18, 354
- Cholesky decomposition 201
- chooser options 119–20
- cin 18–19, 24–5, 71–2, 119, 122–3

- circles 237–8
- circular lists 172–3
- `class` 21–5
- class libraries 8, 40–1
- class-based object-oriented languages, concepts 32–3
- classes
  - see also* template classes
  - abstract concepts 32–3, 115–19, 133–4, 210–13, 380–1, 399
  - accessibility issues 33–4, 40–1
  - aggregation relationships 227–33, 238–42, 348–9
  - algorithms 171–2, 213–15
  - arrays 86–92, 196–202, 364
  - association relationships 229–33, 238–42, 348–9
  - binomial method 245–63
  - concepts 7–11, 16–17, 19–30, 31–47, 49–62, 75–8, 86–92, 99–111, 115–19, 133–52, 171–2, 225–33, 346–9
  - concrete classes 115–19, 210–13, 358–9
  - creation 19–22, 32–47, 49–62, 79–92, 154–65
  - deep hierarchies 127
  - definition 9, 32
  - detailed examination 40–1
  - documentation issues 225–33, 245–6
  - European options 31–47, 246–63
  - examples 9–10, 31–47
  - foundation classes 75–8, 169–70, 348–9
  - friend classes 73–4
  - generalisation/specialisation relationships 113, 225–33, 238–42
  - lifecycles 32–3, 54–5
  - object determination 133–52
  - origins 8
  - parametrised classes 232–3
  - payoff classes 113–31
  - pragmatic exception mechanisms 145–9
  - relationships 225–42
  - robust classes 49–62, 346–9
  - scope resolution operator (`::`) 35, 41, 101–2, 116–18, 393–4
  - syntax 40–1, 49–62, 154–65, 170
  - test programs 38–40, 162–3
  - usage 38–40
- classless object-oriented languages 9
- client-server programming
  - concepts 140–9, 228–33
  - exception-handling mechanism 140
- COBOL 8, 14, 365
- code files
  - see also* source code
  - concepts 16–22, 31–8
- cognitive psychology 8–9
- collection-members relationship, property sets 210, 213–15
- COM *see* Component Object Model
- comment lines 16–19
- Common Base Class scenario 394–6
- compilation
  - concepts 7–8, 15, 18, 20–2, 25–6, 58, 104, 134, 139, 154, 159, 354
  - errors 25–6, 104, 134, 139, 154, 159
- compile-time parametric arrays 81–2
- `complex` 64–78, 306
- complex numbers
  - arrays 84–6
  - concepts 63–78, 84–92, 279
  - financial applications 84
- complexity analysis
  - concepts 170–83, 346–9
  - examples 171–2
- complexity classes 172
- compliance characteristics, software quality 12–13
- Component Object Model (COM) 27, 347–8, 373, 391–406
  - concepts 391–406
  - fundamentals 401–6
  - golden rule 406
  - historical background 391
  - interfaces 401–6
  - `IUnknown` interface 401–6
  - multiple inheritance 391–9
  - new versions 406
  - objectives 391
- Composite pattern 236–42
  - see also* structural patterns
- composition 158, 164, 227–33, 238–42
- Computer Aided Design (CAD) 8, 227, 238–42, 312
- Computer Graphics 8, 238–42, 312–13, 347
- computer sciences 169–70, 172–6, 346–9
  - see also* foundation classes
  - data structures 169–70, 172–6, 346–9
- concept map, Monte Carlo simulation 224
- concepts
  - hierarchies 9
  - theory of concepts 8–9
- concrete classes 115–19, 210–13, 358–9
- conditional compilation directives 355
- conformance characteristics, software quality 13
- `const` 21, 34, 52–4, 58, 60, 69, 100, 103–4, 117–18, 135–7, 188–9, 194–6, 206–10, 217–33
- constant algorithms, concepts 171–2
- `const_cast` 139
- constructors
  - see also* copy...; default...

- concepts 32–4, 54–7, 75–8, 81–92, 115–19, 124–30, 154–5, 159–60, 176–85, 192–202, 204–7, 227–33, 239–42, 255–63, 306–9
- detailed examination 54–5
- kinds 54–5, 57, 81, 124–5, 176–85
- container-contents relationships, property sets 215
- containers 27, 40–1, 44–5, 60, 75–8, 147–8, 169, 172–6, 178–83, 187–202, 203–22, 245–63
  - see also* matrices; vectors
  - algorithms 169–70, 172–6, 178–83, 187–202, 213–15
  - types 41, 75, 169, 172–6, 178–9, 187–202
- contingent claims 373
- continuous compounding 42–5
- continuous dividends 260–3, 315–18
- continuous spaces 94–6, 260–3, 305, 378–9
- control variates 340
- convection-diffusion equations 268–82, 283–303, 334–6
  - see also* Black-Scholes options pricing model
- conversion routines, data 27–9, 137–9, 181–2
- coordinates 159–63
- copy 34, 37, 177, 180
- copy constructors
  - see also* constructors
  - concepts 54–5, 57, 86–7, 116–19, 161–3, 306–9
- correlation options 127–30, 283–303
  - see also* basket options; multi-factor option payoffs
- cosine function 73–4
- cost-of-carry 9–10, 32–47, 234–42, 269–82, 365–6, 380–7
- count 177
- cout 18–19, 21–2, 28–9, 43–5, 50, 53–4, 60–1, 66–7, 71–2, 80–2, 97, 100–3, 108, 119, 122–7, 135–7, 405–6
- Covey, Stephen 345
- Cox, Ross and Rubinstein (CRR) 245, 248–63
- cpp files 16–18, 19–26, 28, 35–9, 56, 58, 160–3, 374
- Crank-Nicolson scheme 275, 280, 301, 336
  - see also* finite difference schemes
- creational patterns, concepts 223, 233–42, 267–82, 286–303, 375–6
- credit modeling 340
- cross correlation 287–303
- CRR *see* Cox, Ross and Rubinstein
- cubic algorithms, concepts 171–2, 203, 310–14
- cubic splines
  - see also* interpolation
  - concepts 203, 310–14
- current 190–1
- curve-fitting functions 359, 367–70
- data
  - see also* member data
  - conversion routines 27–9, 137–9, 181–2
  - header files 16–19, 31–4
  - initialisation 16, 34, 55, 255–63, 375–6
  - robustness issues 49–62
  - security issues 49–62
  - structures 75–8
  - types 16–19, 32–3, 209–13, 353–61
  - variable name 16–19
- data modeling, property sets 213–15
- data structures 32–3, 153–4, 169–85, 187–202, 299–303, 315–26, 346–9
  - computer sciences 169–70, 172–6, 346–9
  - concepts 169–85, 187–202, 203–22, 315–26, 346–9
  - efficiency issues 170–83, 299–300
  - information access 176–85, 251–2
  - introduction 172–6
  - iterators 177–83, 208–13
  - navigation 177–83, 208–13
  - QF applications 187–202, 203–22, 299–303, 315–26
  - types 172–6, 187–202, 203
- Datasim visualisation package 373–89
- date 65
- DDE *see* Dynamic Data Exchange
- deallocation essentials, memory management 65, 79–92, 113–15, 346–9, 356–9
- debug facilities 80, 353, 357–8
- declarations/directives, concepts 97–9, 354
- ‘deep’ copies
  - see also* assignment. . .
  - objects 33–4, 70–1, 205
- deep hierarchies, problems 127
- default constructors
  - see also* constructors
  - concepts 54–5, 57, 76–8, 81–2, 86–7, 116–19, 124–7, 160–3, 192–202, 206–7, 227–33, 239–42, 255–63, 306–9
- defining attribute view 8–9, 31
- definite integrals, calculation 328–30, 358–9
- delegation, inheritance mechanism 101, 158
- delete (deallocate memory) 65, 79, 82–92, 114–15, 346, 358
- deltas 31–47, 245–63, 278, 282, 381–7
- deques 173–4
- dereferencing, pointers 82–92, 356–7
- derivatives 8, 9–10, 31–47, 108–9, 175, 209–10, 213–15
  - see also* futures. . . ; options; swaps
- derived classes
  - see also* inheritance mechanism
  - concepts 99–111, 113–31, 134–7, 225–33, 270–82, 287–303, 391–9
- design blueprints 225

- design issues 14, 203–22, 223–42
- design patterns  
*see also* behavioural. . . ; creational. . . ;  
 structural. . . ; Unified Modeling Language  
 appropriate patterns 237–8  
 basket options 286–303  
 binomial method 252–62  
 Black-Scholes options pricing model 265–82,  
 283–303, 384–7  
 concepts 210–13, 223–4, 233–42, 245–63,  
 337–40, 346–9, 384–7  
 documentation issues 225–33, 245–6  
 introduction 223–4, 233–42  
 Monte Carlo simulation 337–40  
 types 223, 233–7
- destructors 32–4, 41, 57, 87, 114–19, 154–5,  
 159–60, 196–202, 204–7, 239–42
- detailed design, software lifecycle 224–5
- development phases, software 223–5
- device-reliability applications, Monte Carlo  
 simulation 330–2
- dictionaries *see* maps
- diffusion 96–8, 246–63, 268–303, 334–6  
*see also* convection-diffusion equations
- directives, concepts 97–9, 354
- Director classes 251, 255–62
- directories  
*see also* files  
 structure 19–22  
 types 19–20
- Dirichlet boundary conditions 267–82, 298–303
- discount factor 255–62
- discrete initial conditions, basket options  
 298–303
- discrete spaces 94–6, 245–60, 305, 378–9
- dividends 260–3, 287–303, 333–43
- DLLs 232–3
- documentation issues 225–33, 245–6  
*see also* Unified Modeling Language
- dot and cross products 67–8
- dot notation 38–40
- double 354, 357–8, 365–70
- double-ended queues (deques) 173–4
- doubles  
 concepts 10, 16–19, 28–9, 32–8, 42–5, 94–6,  
 210–13, 354  
 examples 10, 16–19, 28–9, 34, 94–6
- doubly linked lists 172–3
- downcasting, concepts 138–9
- drift terms 96–8, 246–63, 283–303, 320–1, 334–6
- dual-strike options 128–30, 284–303
- dynamic casting, concepts 133, 137–52, 212–13
- Dynamic Data Exchange (DDE) 391
- dynamic memory allocation, arrays 81–2, 83–6,  
 364
- dynamic type checking 133–52
- `dynamic_cast` 137–9, 151–2
- efficiency issues  
*see also* asymptotic complexity;  
 performance. . .  
 algorithms/data structures 170–83, 299–300  
 libraries 203  
 software quality 12–13, 223–5, 341
- elasticity 381–7
- encapsulation, concepts 32–3, 75–8, 120–1,  
 187–202, 287–303
- end of life, software lifecycle 225
- energy markets 342
- entities, theory of concepts 8–9
- enum 357
- enumeration 357
- Equal Probability jumps (EQP) 249, 253–5,  
 257–8
- erase 189
- erasure, map records 189
- Err object 147
- errors 25–7, 104, 123–7, 133–5, 139, 140–9  
*see also* exception-handling mechanism  
 compilation 25–6, 104, 134, 139, 154, 159  
 debug facilities 80, 353, 357–8  
 inheritance mechanism 123–7  
 kinds 25–7, 133–5, 140–9  
 linker errors 26  
 memory management 26–7, 80  
 run-time errors 26–7, 133–5, 140–9  
 zero-divides errors 140–5, 148
- esoteric applications 133–52
- Euler finite difference schemes 272–3, 279–82,  
 297–303, 336, 343
- European options 31–47, 246–63  
*see also* options
- Excel 9–10, 149, 176, 187, 199–200, 276–8, 299,  
 301–2, 343, 347, 373–89, 401  
 C++ 373–89  
 COM interfaces 401  
 concepts 373–89, 401  
 Datasim visualisation package 373–89  
 exported data 373–89  
 finite difference method 384–7  
 mechanisms 376–80  
 option values 380–7  
 QF applications 376–89  
 sensitivities (the Greeks) 380–7
- ExcelDriver 373–89
- ExcelMechanisms 373–89
- exception 144–9
- exception specification 150
- exception-handling mechanism 45, 87, 133,  
 140–52, 200, 302, 346–9, 374, 378–9

- see also* errors
- client-server programming 140
- concepts 140–52, 200, 275, 302, 346–9, 374, 378–9
- finite difference schemes 275, 302
- pragmatic exception mechanisms 145–9
- try, throw and catch 141–52, 302, 378–9
- exchange options 128–30, 284–303
- exotic options 127–30, 267–82, 283–303, 321–2, 338–40
- expected values 330–1
- expiry dates 9–10, 31–47, 234–42, 265–82, 287–303, 333–43, 365–6, 380–7
- explicit Euler finite difference schemes 272–3, 279–82, 297–303, 336
- explicit finite difference scheme, concepts 272–5, 279–82, 297–303, 336
- exponential algorithms, concepts 171–2
- exponential functions 72–4, 262–3, 326
- exported data, Excel 373–89
- Extensible Markup Language (XML) 96, 224–5
- extension of a concept, defining attribute view 8–9
- Extension pattern 237–42
  - see also* behavioural patterns
- extrapolation 275, 336, 359
  
- Façade pattern 236–42
  - see also* structural patterns
- Factory pattern
  - see also* creational patterns
  - concepts 234–42, 252–63, 267–82, 286–303
- FAILED macro, IUnknown interface 402–6
- far field boundary 234–42, 287–303
- fault tolerance 12, 341
- FIFO structures, queues 174
- files 16–30, 31–4, 162–3
  - see also* directories; header files
  - confusions 18
- financial engineering 11–14, 41–5, 60, 84, 93–111, 113–31, 169–85, 203–22, 245–303, 305–14, 327–43, 345–9, 367–70
  - complex numbers 84
  - curve-fitting functions 359, 367–70
  - functions 94–6, 104–9, 113–31, 367–70
- Financial Instrument Pricing in C++* (author) 7, 11, 128
- find 177
- finite difference schemes 31, 128, 158, 236–7, 265–82, 283–303, 309, 333–6, 340–1, 347–9, 384–7
  - basket options 283–303
  - comparisons with other methods 340–1
  - concepts 265–82, 283–303, 309, 333–6, 340–1, 347–9, 384–7
  - data members 270–5
  - Excel 384–7
  - exception-handling mechanism 275, 302
  - Monte Carlo simulation 340–1
  - SDEs 333–6
  - two-factor problems 297–303, 343
  - types 266, 272–5, 297–303
  - visualisation packages in Excel 384–7
- fixed assemblies, property sets 213–15
- fixed-income applications 42–5, 66, 93, 104–9, 325
  - see also* bonds
  - volatility calculations 104–9
- flexible payoff hierarchy 302–3
- float 354
- float.h library 367
- floats 28–9, 306, 354
- floor 367
- fmod 367
- for loop 44–5, 50–1, 66, 80, 87–8, 103–4, 136, 179–82, 191, 212–13, 256, 317–18, 323–4, 358–9, 364–70
- foreign exchange options 128–30, 284–303
- Fortran 8, 11, 178, 203
- Forward Differencing in Time and Centred Differencing in Space (FTCS) 272–3
- forward induction 219–21, 245–63, 316–18
- forward iterators 177–8
  - see also* iterators
- forward references 147–8
- foundation classes
  - see also* computer sciences
  - concepts 75–8, 169–70, 348–9
- fragile base class problem 127
- free 358
- free store
  - see also* heap
  - concepts 55–6, 79–92
- Frege, G. 8
- friend 69–70, 194–6
- friend classes, concepts 73–4
- friend functions 69–70, 72–4
- FTCS *see* Forward Differencing in Time and Centred Differencing in Space
- function overloading, concepts 57
- function pointers
  - advantages 96
  - application areas 96, 104–9
  - C 357, 358–9, 363–71
  - concepts 93–6, 98–9, 104–9, 117–23, 357, 358–9, 363–71
  - examples 95–6, 104–9
  - namespaces 98–9
- function prototypes, concepts 16, 31–4, 58
- functional programming styles 11, 326, 346–9

- functionality characteristics, software quality 12–13
- functions  
*see also* member functions  
 calls 49–51  
 categories 94–5  
 concepts 9–11, 16–30, 31–47, 93–111, 158–65, 176–85  
 financial engineering 94–6, 104–9, 113–31, 367–70  
 friend functions 69–70, 72–4  
 global functions 58  
 header files 16–19, 31–4, 88–9  
 input arguments 16–17, 363–71  
 names 16–17  
 non-member functions 57, 58, 69–71, 96, 158, 160–3  
 principle of substitutability 116–17  
 return types 16–17  
 scope resolution operator (::) 35, 41, 101–2, 116–18, 393–4  
 signatures 16, 18  
 template functions 10–11, 22–5, 27–9, 158–65, 179–83, 201–2  
 future values 42–7  
 futures markets, technical analysis 322–5
- gammas 31–47, 245–63, 278, 282, 308–14, 381–7  
 ‘Gang of Four’ (GOF) 223, 228–33, 238, 241, 255, 260, 375
- GBM *see* Geometric Brownian Motion
- generalisation/specialisation relationships 113, 225–33, 238–42  
*see also* inheritance. . .  
 classes 113, 225–33
- generic data structures 32–3, 153–4, 169–85, 187–202, 203–22, 315–26  
*see also* data structures  
 concepts 169–85, 315–26
- generic programming  
 concepts 10–11, 130, 153–65, 346–9  
 relative unpopularity 154
- Geometric Brownian Motion (GBM) 246–9, 315–18
- geometric processes, SDEs 334–6
- get 58
- getString 27–9
- global functions  
*see also* non-member functions  
 concepts 58, 96
- Glockenspiel 7
- GOF *see* ‘Gang of Four’
- goto 107
- graphics 8, 65–6, 67–8, 238–42, 347, 373–89
- The Greeks 260–3, 278, 282, 308, 373, 381–7  
*see also* deltas; gammas; rho; thetas; vegas
- h file extension 35
- Halton numbers 340
- header files 15, 16–30, 31–8, 40–5, 58, 88–9, 353–4, 374–7  
 array classes 88–9  
 canonical header files 116–19  
 compilation errors 25–6  
 concepts 16–22, 31–8, 42–5, 58, 88–9, 353–4  
 confusions 18  
 contents 16  
 private part 33–4, 40, 52–4, 56  
 public part 33–4, 40–1, 56  
 templates 23
- heap 56, 65, 79–92, 177, 346–9, 356, 364, 369–70  
*see also* free store  
 algorithms 177  
 concepts 79–80, 82–92, 177, 346–9, 356, 364, 369–70
- hedging 9–10, 397–8, 403–6
- helper functions 54–5
- heterogeneous data types, property sets 209–13
- hierarchies, concepts 9, 238–41
- historical background, C++ 7–8
- hollow circles 237–8
- hpp files 16–17, 19–23, 34–9, 42–5, 56, 58, 88, 146–9, 160, 204, 374
- HRESULT, IUnknown interface 401–6
- hyperbolic functions 72–4
- IDispatch 401
- if statement 10–11, 16–25, 38, 49–50, 117–19, 137–8, 155–7, 162–3, 235–42, 271–2, 308–10, 381–3, 404–6
- IHedge 397–8, 403–6
- IID, IUnknown interface 401–6
- IInstrument 397–8, 402–6
- implementation, software lifecycle 225
- implementation inheritance, concepts 113
- implicit Euler finite difference schemes 273–4, 279–82, 301–2
- implicit finite difference scheme, concepts 266, 273–5, 279–82, 301–2
- implied trees 203
- implied volatility  
*see also* volatility  
 calculations 108–9, 202  
 concepts 108–9, 202
- information-hiding principle, concepts 89, 101, 193–4, 358
- inheritance mechanism 13, 79, 93, 96, 99–111, 113–31, 134–7, 158, 209–15, 225–33, 346–9, 391–9  
*see also* heap; polymorphism  
 advanced inheritance 113–31, 346–9  
 application areas 104–9, 113–31  
 basic structure 99–109

- concepts 93, 99–111, 113–31, 134–7, 158, 209–10, 225–33, 346–9, 391–9
  - dangers 113, 123–7
  - definition 99
  - errors 123–7
  - examples 93, 99–109
  - multiple inheritance 104–6, 127, 391–9
  - overuse dangers 13
- `init` 37, 54–5
- initialisation 16, 34, 55, 255–63, 375–6
- `inline` 58, 61
- inner product spaces 76–8
- input arguments, functions 16–17, 363–71
- input/output mechanisms (I/O) 18–19, 21–5, 28–9, 43–5, 50, 53–4, 60–1, 65–7, 71–2, 80–2, 97, 100–2, 108, 122–7, 135–7, 363–71, 373–406
- inputs, application activities 347
- `insert` 190–6
- installability characteristics, software quality 13
- instances 8–11, 32–3, 115–19, 176–85
  - see also* classes
  - examples 9–10
- instantiation
  - see also* template classes
  - concepts 163, 169–70, 176, 392–9
- `int` 354
- `int main` 18–19, 22, 24–5, 28–9, 80–2, 96, 114–15, 122–3
- intension of a concept, defining attribute view 8–9
- interest rates 41–5, 234–42, 268–303, 315–26, 333–43, 365–6, 380–7
- `interface` 396–8, 402–6
- interface inheritance, concepts 113, 396–8
- interfaces 113, 178, 187–202, 232–3, 255–63, 373–406
- interoperability characteristics, software quality 12–13
- interpolation 203, 282, 310–14, 328, 359, 367–70
  - concepts 310–14, 359
  - methods 310–13, 367–70
- interviews, programmers 363
- `iostream` 18–19, 24–5, 28–9, 71–2, 145–9, 159–63, 357–8
- ISO 9126 quality characteristics 11–12, 14, 341
- ISO standards 11–12, 14, 169, 341
- iterative schemes, numerical analysis 105–9
- iterators
  - see also* pointers
  - concepts 169–85, 188–9, 195–202, 208–15, 230–3, 239–42
  - examples 179–83, 188–9, 195–202
  - maps 188–9
  - property sets 208–13
  - types 177–8
- `IUnknown` interface 401–6
- Jackson, Michael 60
- Japan 345
- Java 9
- JR 245, 250–1, 253–5, 257–60
- judo 345
- jump probabilities 153–4, 320–1
- K & R reference, C language 353
- key-value pairs, maps 41, 169–70, 176, 187–202, 204–22
- lattice structures
  - see also* binomial method; trinomial method
  - advanced data structures 318–21
  - comparisons with other methods 340–1
  - concepts 203, 215–21, 245–63, 315–26, 349
  - examples 217–21
  - generic structures 322–6
  - issues 215–16
- law of large numbers 331–2, 343
- layering, software 348–9
- lazy initialisation 375–6
- learnability characteristics, software quality 13
- least squares interpolation, concepts 310–14, 367–70
- legacy applications 28–9
- libraries
  - see also* Standard Template Library
  - C 366–71
  - concepts 8, 27–9, 35–8, 40–1, 71–2, 75, 147–8, 187–202, 203–22, 363–71, 373–89
  - efficiency issues 203
  - Fortran 11, 203
  - string stream library 27–9
- LIFO structures 173
- lightweight payoff classes 119–23
- `limits.h` library 367
- linear algorithms, concepts 171–2, 203
- linear interpolation 203, 282, 310–14
- linear regression 310–14, 363, 367–70
- lines 227–33, 237–42
- linker errors 26
- Linux 45
- `list` 170, 201, 236–42
- lists 41, 169, 172–3, 201, 206, 236–42
  - concepts 169, 172–3
  - nodes 172–3
  - types 172–3
- local directories 19–22
- logarithmic algorithms, concepts 171–2
- logic 8–9
  - errors 26–7
- logical units 170–83
- lognormal processes 333
- longs 28–9, 300–3
- loop optimization, concepts 60

- LU decomposition, triangular matrices 305–14, 349
- machine-readable code, compilation process 15
- macro languages 354–5
- magic numbers 81–2, 269–70, 364
- main 18–19
- maintainability characteristics, software quality 7, 12–13, 225, 250, 341
- maintenance, software lifecycle 225
- malloc 356, 358
- many-to-many relationships 176, 229–33
- map 187–202, 204–22
- mappings
  - concepts 94–6, 104–9, 225, 231–42
  - function categories 94–6, 104–9
- maps
  - concepts 41, 169–70, 176, 187–202, 204–22
  - definition 187
  - iterators 188–9
  - QF applications 190–1
  - record erasure 189
- mathematical expectation 330–1
- math.h library 363, 366–7
- matrices 58, 60, 65–6, 75–8, 176, 187–202, 203, 270–82, 299–303, 305–14, 349, 373–89
  - see also* containers
  - Excel 373–89
  - map QF applications 190–1
  - multiplication 77–8
  - optimization 60
  - triangular matrices 305–14, 349
  - tridiagonal matrices 190–1, 305–14
  - visualisation packages in Excel 373–89
- matrix 65, 75–8
- maturity characteristics, software quality 12–13
- Max 10–11, 16–19
- mean
  - law of large numbers 331–2, 343
  - reversion 318–19, 334–6
- mechanisms, software layering 348–9
- member data
  - see also* attributes
  - concepts 16–19, 31–4, 55–7, 365–6
  - static member data 55–7
- member functions
  - see also* functions; operations
  - concepts 10–11, 19–22, 31–47, 52–4, 68, 87–8, 204–5
  - constructors 32–4, 54–7, 75–8, 81–92, 115–19, 124–30, 154–5, 159–60, 176–85, 204–7, 227–33, 239–42, 306–9
  - destructors 32–4, 41, 57, 87, 114–15, 154–5, 159–60, 204–7, 239–42
  - dot notation 38–40
  - examples 10–11, 31–47
  - modifiers 32–3
  - non-member functions 57, 58, 69–71, 96, 158, 160–3
  - operator overloading 63–78, 87–8, 205, 346–9
  - property pattern 204–5
  - read-only (const) member functions 52–4
  - scope resolution operator (::) 35, 41, 101–2, 116–18, 393–4
  - selectors 32–3, 232–3
  - static member functions 55–7
- memory management
  - see also* heap; stack
  - allocation/deallocation essentials 65, 79–92, 113–15, 245–63, 346–9, 353, 356–9
  - automatic memory 55–7, 79–92, 364
  - concepts 26–7, 79–92, 179–83, 346–9, 364
  - errors 26–7, 80
  - information-hiding principle 89, 101, 193–4, 358
  - usage methods 55–7, 79–92, 179–83
- merging algorithms 177, 181–3
- mesh data, Finite Difference Method 270–82, 297–303
- message-passing metaphor 38–40
- metadata 133–4
- MFC 347
- Microsoft 134–5, 391, 396–7
- mid-point rule 358–9
- Milstein method 336, 343
- Min 16–19
- modeling, options 9–10, 31–47
- modf 367
- modified CRR 245, 250–1, 257–60
- modifiers 32–3, 189, 199–200, 258–9
- modify 189, 199–200
- modular programming styles 8, 11, 16, 41–5
- monolithic code 250
- Monte Carlo simulation
  - application examples 327–32, 341–3
  - C++ 337–40, 347–9
  - comparisons with other methods 340–1
  - concept map 224, 347–9
  - concepts 224, 327–43, 347–9
  - critique 327, 337
  - examples 327–32, 338–42
  - historical background 327
  - pricing 224, 333–43, 347–9
  - QF 224, 333–43
  - SDEs 333–6
  - software architecture 337–40
  - test programs 338–40
- multi-asset options 128–30, 283–303, 321–2
  - see also* basket options
  - types 128–30, 283–4
- multi-dimensional arrays, C 363–71

- multi-factor option payoffs 113–31, 203, 234–42, 283–303, 337, 343, 347–9
  - see also* correlation options
- multidimensional binomial method 321–2
- multimaps, concepts 169, 176, 230–42
- multiple inheritance
  - see also* inheritance. . .
  - COM 391–9
  - concepts 104–6, 127, 391–9
  - interfaces 396–8
  - problems 127
- multisets 169, 175
- mutating algorithms 177–83
  - see also* algorithms
- names, functions 16–17
- namespace 18, 19, 21, 24–5, 28–9, 42–5, 56, 71–2, 93, 96–9, 145–9
- namespaces 18, 19, 21, 24–5, 28–9, 42–5, 56, 71–2, 93, 96–9, 104–9, 145–9, 158–65
  - advantages 43
  - aliases 98–9
  - application areas 104–9
  - concepts 42–5, 96–9, 104–9, 158–65
  - definition 43, 96–7
  - examples 42–5, 96–9, 104–9
  - function pointers 98–9
  - naming considerations 97–8
  - template functions 158–65
- navigation, data structures 177–83, 208–13
- Neville’s algorithm 310
- new (allocate memory) 65, 79, 82–92, 113–15, 346, 356, 358
- Newton-Raphson method 93, 105–9
- nodes, lists 172–3
- non-const 188–9
- non-member functions
  - see also* global functions
  - concepts 57, 58, 69–71, 96, 158, 160–3
- non-mutating algorithms 177–83
  - see also* algorithms
- nonlinear equations, solutions 104–11
- Notification pattern 237–42
  - see also* behavioural patterns
- Numeric Linear Algebra 203
- numerical analysis
  - concepts 305–14, 328, 347–9, 359
  - definition 305
  - QF applications 305–14, 347–9
- numerical integration 328–30, 341–2, 358–9
- NumericMatrix 75–8
- Object Linking and Embedding (OLE) 391
- Object Management Group 225
- object-oriented paradigm (OO)
  - see also* encapsulation; information-hiding. . . ; inheritance. . . ; polymorphism
  - bottom-up approach 13
  - concepts 7–10, 31–47, 49–62, 127, 154, 346–9, 353–61, 391–9
  - critique 8, 154
  - message-passing metaphor 38–40
  - programming languages 9
  - theory of concepts 8–9
- objects
  - anonymous objects 59, 69–70
  - arrays 79–92
  - casting concepts 137–52, 346–9
  - class determination 133–52
  - concepts 8–11, 25–30, 32–4, 59, 79–92, 115, 133–52, 346–9
  - ‘deep’ copies 33–4, 70–1, 205
  - dot notation 38–40
  - examples 9–10
  - lifecycles 32–3, 54–5
  - robustness issues 49–62, 346–9
  - run-time behaviour 133–52
  - self-aware objects 133–52, 204, 212–13
  - single objects 79–92
  - tangible aspects 32–3
- OLE *see* Object Linking and Embedding
- one-dimensional arrays, C 363–71
- one-factor options 113–31, 203, 234–42, 245–63, 265–82, 285–6, 337, 347–9, 384–7
- one-to-many relationships 229–33
- OO *see* object-oriented paradigm
- operability characteristics, software quality 13
- operating system (OS) 174, 347–8
- operations
  - see also* member functions
  - concepts 8–9, 31–2
- operator 68–72, 194–6, 204–5
- operator overloading
  - advantages 65–8
  - concepts 63–78, 87–8, 205, 346–9
  - important applications 65
  - possibilities 63–5
  - simple I/O operations 71–2
  - steps 68–72
  - usage rule-of-thumb 65
- operators
  - see also* entries at start of index
  - list 63–4
  - operator overloading 63–78, 87–8, 346–9
  - types 63–4, 82–92
- optimization 49, 58–60
  - see also* performance issues
  - anonymous objects 59, 69–70
  - inline 58, 61
  - Jackson’s rules 60
  - loop optimization 60

- options 8, 9–10, 31–47, 108–9, 175, 209–10, 213–21, 234–42, 245–63, 265–303, 333–43, 380–7, 403–6
  - see also* American...; call...; European...; pricing; put...
  - Black-Scholes options pricing model 31–3, 37, 246, 259–60, 265–82, 283–303, 309, 366, 381–7
  - correlation options 127–30, 283–303
  - The Greeks 260–3, 278, 282, 308, 373, 381–7
  - modeling 9–10, 31–47
  - Monte Carlo simulation 224, 333–43
  - payoff functions 113–31, 245–63
- Ornstein-Uhlenbeck model 318–19
- Orthogonal Base Class scenario 391–3
- OS *see* operating system
- ostream 160–3
- out-performance options 128–30, 284–303
- outputs
  - application activities 347
  - exported data 373–89
- overflow/underflow errors 26–7, 80, 140–9
- overview, C++ 345–9
- OWL 347
  
- Padé approximation *see* Cayley map
- pair 188–9
- parabolic differential equations 283–4, 384–7
- paradigms, concepts 7–11, 41–5, 58, 130, 346–9, 353
- parameters, concepts 49–62, 232–3
- parametrised classes, concepts 232–3
- parsers 65–6
- partial differential equations (PDEs) 127–30, 262–3, 265–82, 283–303, 312, 328, 381–7
  - see also* Black-Scholes options pricing model
  - basket options 285–303
  - C++ classes 292–7
- Pascal Calling Convention 396–7
- path generation 336
- payoff classes/functions
  - advanced inheritance 113–31
  - lightweight payoff classes 119–23
  - super lightweight payoff classes 121–3
- PDEs *see* partial differential equations
- performance issues
  - see also* efficiency...; optimization
  - comparisons of numerical methods 340–1
  - tips 58–60, 262–3, 340–1
- philosophy 8–9
- Planck's constant 279
- POC *see* Proof-of-Concept
- pointers 79, 82–96, 117–23, 136–52, 353, 355–7, 363–71, 399
  - C 353, 355–6, 363–71
  - casting concepts 137–52, 346–9
  - concepts 82–92, 93–6, 136–52, 353, 355–6, 399
  - dereferencing 82–92, 356–7
  - function pointers 93–6, 98–9, 104–9, 117–23, 357, 358–9, 363–71
  - Virtual Function Tables 399
- poles, rational interpolation 311
- polylines 227–33, 237–42
- polymorphism
  - see also* dynamic casting; virtual
  - concepts 96, 102–4, 106–9, 113–15, 134–52, 209, 228–42
- polynomials, concepts 93, 310–14
- portability characteristics, software quality 12–13
- portfolios 137, 209–13
  - information access 137
- postconditions, exception-handling mechanism 141–5
- powers of numbers 43–4
- pragmatic exception mechanisms, concepts 145–9
- preconditions, exception-handling mechanism 141–5
- Predictor-Corrector method 336
- preprocessor, C 354–5
- present value 42–7
- pricing 8, 9–10, 31–47, 108–9, 213–21, 226–7, 234–42, 245–63, 265–82, 283–326, 366, 380–7, 403–6
  - basket options 283–303, 321–2
  - binomial method 31, 153, 158, 201, 203, 215–16, 219–21, 245–63, 321–2, 340–1, 347–9
  - Black-Scholes options pricing model 31–3, 37, 246, 259–60, 265–82, 283–303, 309, 366, 381–7
  - Monte Carlo simulation 224, 333–43, 347–9
  - trinomial method 153, 158, 201, 215–19, 245, 263, 297–8, 315–26, 340–1, 347–9
- principle of substitutability, concepts 116–17
- print 21, 78, 81–3, 100–3, 124–7, 188–91, 197–200, 209–10
- priority queues 174
- private 33–4, 40, 52–4, 56, 67–70, 74, 76–7, 88–9, 104, 107–9, 114–15, 120–3, 125–30, 154–5, 194–202, 206–7, 214–15, 227–33, 255–62, 284–303
- procedural programming styles 11, 41, 58
- processing, application activities 347
- programmers
  - see also* software
  - C basics 353–61
  - interviews 363
  - productivity issues 203
  - skills' development 345–9, 359, 363–71
- programming paradigms, concepts 7–11, 41–5, 58, 130, 346–9, 353

- Proof-of-Concept (POC) 288–92
- property pattern 203–22
  - concepts 203–5
  - examples 204–5
  - functions 206–7
- property sets 203, 205–22
  - additions/removals 207–8
  - concepts 205–22
  - data modeling 213–15
  - heterogeneous data types 209–13
  - iterators 208–13
  - navigation 208–13
  - QF data modeling 213–15
- Prototype pattern
  - see also* creational patterns
  - concepts 234–42, 287–303
- prototypical objects 9, 204–5
- pseudo random numbers 336
- `public` 21–2, 33–4, 40–1, 50, 52–4, 56, 81–2, 88–9, 99–111, 114–15, 121–30
- Pull operations, stack 173–4
- Push operations, stack 173–4
- put options 9, 31, 32–8, 108, 118–21, 234–42, 246–63, 283–303
- QF *see* quantitative finance
- QMC *see* Quasi-Monte Carlo
- quadratic algorithms, concepts 171–2
- quantitative finance (QF) 11–14, 41–5, 60, 93–111, 113–31, 178–83, 187–202, 203–22, 223–42, 245–343, 345–9, 376–89, 403–6
- C++ relationship 11–14, 203, 223–5, 345–9
- concepts 11–14, 41–5, 60, 178, 187–202, 213–15, 305–26, 345–9, 403–6
- data types 213–15
- design patterns 223–4, 233–42, 245–303, 384–7
- Excel 376–89
- maps 190–1
- modular programming styles 41–5
- Monte Carlo simulation 224, 333–43
- numerical analysis 305–14, 328, 347–9, 359
- property sets 213–15
- STL 178, 187–202, 346–9
- tridiagonal matrices 305–14
- Quasi-Monte Carlo (QMC) 330, 336
- quasi-random sequences 330
- queues, concepts 169, 173–4
- quotient options 128–30, 284–303
- rainbow options 128–30, 284–303
- random access iterators 177–8
- random numbers 327–43
- random walks 318–19
- range errors, arrays 80–1, 140–52, 200
- rational interpolation, concepts 203, 310–14, 326
- Rayleigh-Ritz-Galerkin method 312
- read-only (const) member functions 52–4
- real-valued functions, concepts 94–6, 104–9
- record structures, COBOL 365
- recoverability characteristics, software quality 12–13
- rectangles, creation 123–7, 285–303
- recursive functions 42–5, 104, 215, 236, 310–11
- recursive/nested property sets 215
- reference calls *see* call by reference
- references, C 355–6, 406
- reflection, concepts 133–52, 204–5, 212–15
- reflexive friend relationships 74
- `reinterpret_cast` 139
- relational databases 365
- relationships, classes 225–33
- reliability characteristics, software quality 12–13, 223–5, 250
- remote directories 19–22
- `remove` 177, 180–1, 192–6, 207–8, 227–33, 239–42
- `replace` 177, 180
- replaceability characteristics, software quality 13
- requirements, software lifecycle 7, 11–14, 224–5
- resource efficiency 13
- `return` 22–9, 36–8, 43–5, 52–4, 58–9, 67–8, 156, 162–3, 193–4, 208, 229
- return types, functions 16–17
- `reverse` 177
- `rho` 129–30, 287–303, 381–7
- Richardson extrapolation 275, 336
- risk management 8, 9–10
- risk-free interest rates 9–10, 31–47, 246, 283–303, 315–26, 333–43
- risk-neutrality 315–18
- robust classes
  - see also* classes
  - creation 49–62, 346–9
- `rotate` 180
- RTTI *see* run-time type information
- run-time behaviour 133–52, 346–9
- run-time errors 26–7, 133–5, 140–9
- run-time type information (RTTI)
  - see also* `typeid`; `type_info`
  - concepts 134–7, 212–15
- scalar-valued functions, concepts 94–6
- Schaum vectors 67
- Schrödinger equation 279
- scope resolution operator (`::`), concepts 35, 41, 101–2, 116–18, 393–4
- SDEs *see* stochastic differential equations
- `search` 177
- Secant method 105–9
- second-order parabolic differential equations 283–4

- security issues 12–13, 49–62
- selectors 32–3, 232–3
- self-aware objects, concepts 133–52, 204, 212–13
- Self language 9
- Semi-Orthogonal Base Class scenario 392–3
- sequential iterators 177–8
- server code, client-server programming 140–9
- set 58, 187–202, 206–7
- Set and Get functions 205
- sets
  - concepts 169, 174–6, 187–202, 203, 205–22
  - property sets 203, 205–22
  - STL 192–6
  - user-friendly sets 191–6
  - wrapper classes 194–6, 206–7, 210–13, 238–42
- shapes 123–7, 190–1, 223–42
- short rate trees, trinomial method 318–21
- signatures, functions 16, 18
- Simula 7
- sine function 73–4
- Singleton pattern 57, 147–8, 234–42, 375–6
  - see also* creational patterns
- singly linked lists 172–3
- skills' development 345–9, 359, 363–71
- skip lists 172–3
- smoothing functions 359
- Sobol numbers 330, 337, 340
- software
  - see also* applications; source code
  - debug facilities 80, 353, 357–8
  - development phases 223–5
  - documentation issues 225–33, 245–6
  - ISO 9126 quality characteristics 11–12, 14, 341
  - layering 348–9
  - lifecycle 223–5
  - monolithic code 250
  - Monte Carlo simulation 337–40
  - quality issues 11–14, 223–5, 250, 341
  - requirements 7, 11–14, 224–5, 249–50, 341, 348–9
  - understandability/maintainability needs 7, 12–13, 225, 250
  - 'uses' hierarchy 348
- solid circles 237–8
- sorting algorithms 177, 180–3
- source code
  - see also* errors; software
  - compilation process 15, 18, 20–2, 25–6, 58, 104, 134, 139, 154, 159, 354
  - debug facilities 80, 353, 357–8
  - files 16–22, 31–4, 162–3
  - mechanics 15–30, 31–47
  - monolithic code 250
- sparse matrices 190–1
- specialisation/generalisation relationships *see* generalisation/specialisation relationships
- specific data types 32–3
- splitting methods 283
- spread options 128–30, 284–303
- spreads 117–19, 128–30
- spreadsheets 187–9, 190–202
- sprintf 29
- square brackets 83–8
- square root processes, SDEs 334–6
- squares, creation 123–7
- stability characteristics, software quality 13
- stack 51, 56–7, 79–92, 169, 173–4, 236–42, 346–9, 364, 396–7
  - see also* automatic memory
  - concepts 51, 56–7, 79–92, 173–4, 346–9, 364, 396–7
  - LIFO structures 173
  - Push/Pull operations 173–4
- Standard Template Library (STL)
  - see also* std. . .
  - concepts 40–1, 44–5, 75–8, 147–8, 151, 158, 169–85, 187–202, 238–42, 346–9, 354, 363
  - first example 178–83
  - iterators 169–85
  - overview 169–85
  - QF interfaces 178, 187–202, 346–9
  - sets 192–6
  - simpler interfaces 187–202
  - syntax 170
  - uses 169–85, 187–202, 238–42, 363
- static 56–7, 137–9, 403
- static arrays 80–6, 364
  - see also* arrays
- static casting, concepts 133, 137–52
- static member data/functions, concepts 55–7
- static memory, concepts 55–7
- static\_cast 137–9
- std, namespace 18, 19, 21, 24–5, 28–9, 42–5, 71–2
- stdcall 396–404
- std::string 29
- Steffensen iteration 105–9
- STL *see* Standard Template Library
- stochastic differential equations (SDEs) 96–8, 246–63, 315–18, 321–2, 327–43
  - concepts 333–6
  - finite difference schemes 333–6
  - types 334–6
- stochastic volatility 9–10
- stock prices 32–47
- straddles 118–19
- strangles 118–19
- Strategy pattern, concepts 94, 120–1, 250–62
- stratifications 340
- strike prices 9–10, 31–47, 108, 234–42, 248–63, 265–82, 287–303, 333–43, 365–6, 380–7
- string 354

- string stream library 27–9
- strings 19–22, 205, 209–13, 354
  - data conversions 27–9, 181–2
- stringstream 27–9
- strong law, law of large numbers 331–2
- Stroustrup, Bjarne 7–8, 15, 361
- struct 27, 190–1, 357–8, 363, 365–70, 396–7
- structural patterns
  - concepts 223, 236–42, 252–62, 267–82
  - types 236–7
- 'substrate' classes 203
- SUCCEEDED macro, IUnknown interface 402–6
- suitability characteristics, software quality 12–13, 341
- super lightweight payoff classes 121–3
- swap 177
- swap functions 158
- swaps 175, 209–10
- swing options 342
- symmetry issues, friend relationships 74
- syntax
  - C 346, 347–9, 353–61, 363–71
  - classes 40–1, 49–62, 154–65, 170
  - errors 25–6
  - template classes 154–65, 170
- system directories 19–22
- system patterns 213–15
  
- Tanh integration rule 358, 360
- technical analysis, futures markets 322–5
- template 23–5, 143–5, 154–65, 190–202, 204–8, 217, 220–1, 232–3
- template classes
  - complete example 159–63
  - concepts 22–5, 44–5, 75–8, 153–65, 190–202, 203–22, 267–82, 297–303, 322–6, 346–9
  - creation 154–65
  - reusability benefits 157–8
  - syntax 154–65, 170
  - test programs 162–3
  - usage examples 156–8, 190–202, 322–6
- template functions
  - concepts 10–11, 22–5, 27–9, 158–65, 179–83, 201–2
  - namespaces 158–65
- template mechanism 153–4, 267
- templates, concepts 10–11, 22–5, 27–9, 44–5, 81–2, 153–65, 190–202, 346–9
- temporal data types 203
- tensors
  - see also* arrays; matrices
  - visualisation packages in Excel 373–89
- terminate 150–2
- test programs
  - binomial method 259–60
  - Black-Scholes options pricing model 276–8
  - C 358–9, 367–70
  - classes 38–40, 162–3
  - files 19–22, 38–40
  - Monte Carlo simulation 338–40
  - template classes 162–3
- testability characteristics, software quality 13
- text editors 15
- theory of concepts 8–9
- thetas 282, 381–7
- three-dimensional surfaces 190–202
- throw 144–52
- throw process, exception-handling mechanism 141–52
- time 65–6
- time
  - binomial method 153, 246–63
  - efficiency 13
  - series 324
  - to expiration 9–10, 31–47, 234–42, 265–82, 287–303, 365–6, 380–7
- time.h library 367
- transaction costs 246
- transform 177, 182–3
- transitive issues, friend relationships 74
- TRG 253–4, 257–60
- triangles 237–8, 305–14, 349
- triangular matrices 305–14, 349
- tridiagonal matrices
  - C++ 306–9
  - definition 305
  - numerical analysis 305–14
  - solving 305–9
  - sparse arrays 190–1
- trigonometric functions 72–4, 366–7
- trinomial method 153, 158, 201, 215–19, 245, 263, 297–8, 315–26, 340–1, 347–9
  - see also* lattice structures
  - C++ 315–26, 347–9
  - comparisons with other methods 340–1
  - concepts 315–26, 340–1, 347–9
  - Monte Carlo simulation 340–1
  - short rate trees 318–21
- try 144–52, 378–9
- try, throw and catch, exception-handling mechanism 141–52, 302, 378–9
- two-dimensional shapes 123–7, 190–1, 223–42
- two-factor options 127–30, 283–303, 321–2, 343, 347–9
- typedef 44, 157–8, 190–1, 196–202, 208, 231–3, 357
- typeid 134–7
- type\_info 134–7
  
- ULONG, IUnknown interface 402–6
- UML *see* Unified Modeling Language
- unary operators, concepts 64–5, 182

- underlying assets 31, 32–47, 153, 158, 201, 203, 215–21, 245–63, 283–303, 315–26, 333–43, 365–6, 380–7
  - binomial method 31, 153, 158, 201, 203, 215–16, 219–21, 245–63
  - trinomial method 153, 158, 201, 215–19, 245, 263, 297–8, 315–26
- understandability characteristics, software quality 7, 12–13
- unexpected 150
- Unified Modeling Language (UML) 119–20, 124–5, 197, 224–32, 245–6, 250–1, 265–7, 286–7, 292–3
  - see also* design patterns; documentation issues
- union 357–8, 366
- unique 177
- up-and-out calls 338–9
- upcasting, concepts 138–9
- usability characteristics, software quality 12–13
- user-defined types 23, 28–9, 79–92, 194–6, 208–13
  - concepts 79–92, 194–6
  - STL sets 194–6
- user-friendly sets, concepts 191–6
- ‘uses’ hierarchy 348
- using 97
- using declarations/directives, concepts 97–9, 354
  
- valarray 178–83, 363
- value calls *see* call by value
- variable name, data 16–19
- variance 331–43
- Variational pattern 237–42
  - see also* behavioural patterns
- Vasicek model 318–19
- vector 75–8, 94–6, 178–83, 201, 306–14, 363
- vector functions, concepts 94–6
- vector spaces 76–8, 94–6
- vectors 41, 58, 60, 65, 67–8, 75–8, 94–6, 140, 169, 178–83, 201, 203–22, 226–7, 232–3, 258–63, 270–82, 306–14, 349, 363–71, 373–89
  - see also* arrays; containers
  - Excel 373–89
  - multiplication 77–8
  - operator overloading 67–8
  - optimization 60
  - STL 169
    - visualisation packages in Excel 373–89
- vegas 381–7
- virtual 34, 41, 76, 88–9, 102–4, 106–9, 113–31, 134–7, 142–9, 204–5, 228–42, 289–303
  - see also* polymorphism
  - concepts 102–4, 113–16, 134–7
  - function tables 399
- Virtual Function Tables 399
- Visitor pattern 205, 214–15, 240–1
- Visual Basic 147, 337, 373, 401
- Visual Studio 20
- visualisation packages 149, 373–89
  - basic driver functionality 373–6
  - concepts 373–89
  - functionality 373–6
  - option values 380–7
  - sensitivities (the Greeks) 380–7
- void 21, 34, 36–8, 52–5, 81–2, 95–6, 100–9, 116–19, 124–7, 134–7, 142–9, 160–3, 188–9, 193–200, 207–10, 227–33, 239–42, 271–82, 298–303, 357–8
- volatility 9–10, 31–47, 93, 104–9, 187–8, 190–202, 234–42, 246–63, 268–303, 315–26, 365–6, 380–7
  - see also* implied volatility
  - Black-Scholes options pricing model 268–303
  - calculations 104–9
  - fixed-income applications 104–9
  - map applications 190–1
  - surface modeling applications 187–8, 190–202
  
- weak law, law of large numbers 331–2
- Weibull distribution 388
- while loop 188–9
- Whole-Part pattern 213–15, 227–33, 238–42, 337
  - see also* aggregation relationships
- Wilmott, P. 269
- Windows 45, 347–8
- WinForms 347
- ‘work arrays’ 306
- wrapper classes 194–6, 206–7, 210–13, 238–42
  
- XML *see* Extensible Markup Language
  
- yield curves 203
  
- zero-divides errors 140–5, 148