Index

abstract, 34
  calculus, 129, 130, 132, 137, 140, 141
class, 94
data type, 34
data type calculus, 58, 59, 62, 63, 66, 69, 70, 72,
  73, 75–78, 80, 99, 130, 132, 141
interface, 79
abstract calculus, 202, 236, 293, 310
abstract factory, 228, 234
abstract factory pattern, 99
abstract syntax tree, 253
actual argument, 54
adapter pattern, 99
ADT
calculus, 111
ADT calculus, 141
aggregation, 48, 57, 62, 78, 365–367
allocatable, 35, 38, 41, 62
allocation
  sourced, 155
ANSI C, 254
application, 96
application programming interface
  (API), 295
architecture, 7
argument
  actual, 41, 54, 79, 145, 202, 268, 269
dummy, 41, 50, 79, 111, 135, 268
  passed-object dummy, 38, 51, 54, 135
assertion, 232, 233, 241, 242, 247
assertions, 232
assignment
  defined, 55
intrinsic, 55, 58, 70, 78
association, 362
attribute, 34
auto-parallelization, 293
auto-vectorization, 293
Babel, 254
back substitution, 337, 338, 341, 343
backward difference, 349
backward Euler, 59
beginner’s mind, 87
binding
  deferred, 95
bisection method, 71, 72
BLACS, 108
BLAS, 141
body count, 262
Boolean, 35
BOOST, 116
Builder, 227
Burgers equation, 202, 204, 310
butterfly effect, 101
bzip2, 69

C++
  0x standard, 99
class, 36
data members, 36
exception handling, 137
function prototype, 39
namespace, 36
Standard Template Library, 23, 36, 129
  virtual member function, 36
call tree, 234
central difference, 349
chain of responsibility pattern, 99
chaos, 101
character string
  deferred-length, 38
  fixed-width, 38
Chasm, 252–254
class
  child, 57, 78
  interface, 95
coarray, 309
cohesion, 69–71, 109
Common Component Architecture (CCA), 5, 23
compiler
  Cray Fortran, 31
  GNU Fortran, 31
Intel, 31
Index

compiler (cont.)
Numerical Algorithms Group, 31
XL Fortran, 31
complexity, 67, 69, 79
algorithmic, 67, 69
computational, 69
Kolmogorov, 67, 76
theory, 71
component, 23
-based, 24
-based software engineering, 23
derived type, 29
composition, 48, 57, 365–367
concrete class, 94
concurrency, 55
coupled, 32, 34, 38, 41–43, 49, 50, 53, 55
generic name, 37
context, 370
copy
deep, 70
shallow, 70
coupling, 69–71, 81
Cray, 31
dead-code removal, 28, 246
decorator pattern, 99
deferred binding, 79, 95
defined assignment, 55
dependency, 362
depth count, 111
derived type
components, 29
design by contract, 24
design patterns, 5
creational, 203
diagonally dominant, 339
distributed programming, 99
dynamic dispatching, 54
derived type components, 29
design by contract, 24
design patterns, 5
creational, 203
diagonally dominant, 339
distributed programming, 99
dynamic dispatching, 54
dead-code removal, 28, 246
decorator pattern, 99
deferred binding, 79, 95
defined assignment, 55
dependency, 362
depth count, 111
derived type
components, 29
design by contract, 24
design patterns, 5
creational, 203
diagonally dominant, 339
distributed programming, 99
dynamic dispatching, 54
cascade, 31, 36, 39, 50, 54
class, 284, 43, 54, 55
except handling, 136
expression templates, 77
extended type, 22
extensibility, 45, 54, 55
Extensible Markup Language (XML), 252
facade pattern, 108
factory method, 203, 204, 234
fault, 231
final subroutine, 109–111
finite differences
central, 105
Pade, 105
Runge-Kutta, 105
flattening, 255, 257
fork-join, 295
formal
methods, 231, 232
specification, 231
verification, 231
formal constraints, 24
formal methods, 24
Fortran
2008 standard, 39, 99, 155
forward difference, 348
forward elimination, 337
forward Euler, 58
forward reference, 145
framework, 96, 331
function
intrinsic, 22, 41, 53
function evaluation, 131
Gang of Four (GoF), 93–96, 99, 127, 141–144, 164,
202–204, 226–228, 240
garbage collection, 117
Gaussian elimination, 337–342
generalization, 358, 362, 363
generic
class, 37
interface, 41
name, 41
programming, 35
type, 32
graphical user interface, 98
GUI, 98
heat equation, 345
hierarchy
class, 54, 57
inheritance, 54
high-performance computing (HPC), 4, 107
IBM, 31
IEEE, 256
implicit Euler, 351
information
entropy, 74, 75
theory, 69, 73
information hiding, 21, 31, 36, 38, 39, 54, 57
inheritance, 22, 31, 48, 57, 62, 75, 78, 363, 364
initialization, 255, 257
inline, 253
instantiate, 34
interface, 19, 22, 32, 39, 41–43, 50, 53, 57, 58, 61,
67, 73, 75, 76, 78, 79, 361, 363, 370, 371
abstract, 79, 95
blocks, 39
class, 94, 95
explicit, 39
graphical user, 98
implicit, 39
logical, 255
physical, 255
specifications, 39
interface block, 262
interface body, 262
intermediate object representation (IOR), 254
intrinsic assignment, 55, 58, 70, 78
intrinsic function, 22, 41, 53
intrinsic operation, 62
intrinsic type, 35
invariant, 240
iterator pattern, 127
Java, 38, 98, 99, 108, 254
Virtual Machine, 98

kernel, 245
koan, 87

LAPACK, 141
legacy
interfaces, 57
legacy Fortran, 40, 41, 45
Lorenz equations, 101

magnetohydrodynamics (MHD), 100
Mathematical Acceleration Subsystem (MASS), 293
memory leak, 41
memory management, 231
message, 34, 36, 38, 53, 62, 74
Message Passing Interface (MPI), 297
method, 21, 23, 32, 34, 35, 41, 50, 53, 54
module, 62
molecular diffusion, 100
Morfeus, 330
Morpheus, 53
multicore, 294
multiphysics, 3, 4, 6, 100, 285
multiplicity, 49, 243, 367
multithreading, 99, 293–295
multithreads, 7

name mangling, 255
Newton’s method, 343

overloading, 256
operator, 22
operator, 22

package, 130
Pade, 226
parallel programming, 99
parent, 57, 75, 78
partial pivoting, 338, 339, 342, 343
passed-object dummy argument, 38, 51, 54, 135
pattern language, 86, 88, 141
PETSc, 141
phase space, 102
pointer
assignment, 112
polymorphism, 22, 31, 32, 45, 53, 54
dynamic, 38, 53, 54, 57
static, 53
postcondition, 240
precondition, 240
primitive type, 35
procedure
external, 39, 41
final, 42
module, 21, 39, 53
signature, 32, 79
type-bound, 36, 38, 39, 51, 53, 55, 75
procedure pointer, 215
programming
function, 26
functional, 20, 21
generic, 23
object-based, 21, 22, 24
object-oriented, 4, 6, 22
purely functional, 21
structured, 20
unstructured, 20, 21
protocol, 54
puppeteer, 169, 202, 227
Python, 254

quantum vortices, 100

realization, 95
refactor, 40, 59
reference counting, 117
reference-counted pointer (RCP), 117, 273
reuse, 22, 38, 48, 54, 57, 58
reverse engineering, 61, 62, 69
Runge-Kutta, 131
2nd order (RK2), 144
Runge-Kutta methods, 216

scalable abstract calculus, 285
ScalAPACK, 108, 141
scaling
strong, 292
weak, 292
Index

Scientific Interface Definition Language (SIDL), 254
scientific OOP (SOOP), 57, 62, 79, 80
scope, 62
private, 34, 39
public, 34
semidiscrete, 101, 347
sequence association, 268
shadow object, 255
shell, 245, 248, 250, 269, 270
shoshin, 87
side effects, 13, 21, 29, 238
signature, 32, 79
Simplified Wrapper and Interface Generator (SWIG), 252
single instruction multiple data, 293
skeleton, 253
software engineering
component-based (CBSE), 23
speedup, 291
state pattern, 99
stereotype, 34
STL, 23, 36
strategy, 202
strong scaling, 292
struct ID, 269, 270, 273
structure constructor, 37, 55
structured programming, 20, 24
stub, 253
superfluidity, 100
surrogate, 202
symmetric multiprocessing (SMP), 294
Taoism, 87
template, 23, 32
class, 35, 36, 117, 121
emulation, 23
expression, 77
metaprogramming, 116
template method, 228
tensor, 331
calculus, 309
test-driven development, 43
toolkit, 96
Trilinos, 108, 141, 169
truncation error, 348
type
child, 78
derived, 21
extended, 50, 53, 54, 79
extension, 49
finalization, 111
generic, 32, 35
guard, 136
intrinsic, 35, 59, 78
parameter (UML), 35
parameterized, 32
parameterized (UML), 35
parent, 57, 75, 78
primitive, 35
selection, 135
type-bound procedure, 21
typed allocation, 215
types
extended, 22
UML, 33, 232
adorned relationships, 205
attribute, 34–36
class diagram, 34–36, 49, 56, 357, 359, 366
generalization, 50, 363, 364
object diagram, 49, 357, 366
OCL, 232
operation, 32, 34–36, 43, 51, 53
package, 130
package diagram, 357
realization, 95
sequence diagram, 62, 66, 80, 357
use case, 33, 34, 36, 42, 357, 358
Unified Modeling Language (UML), 32
annotation, 240
weak scaling, 292
wrapping legacy Fortran, 40, 41, 45
Zen Buddhism, 87