

Index

Operators are listed under *operator*. That is, `operator<<` is listed under `operator<<`, not under `<<`, etc.

Example classes, structs, and class or struct templates are indexed under *example classes/templates*. Example function and function templates are indexed under *example functions/templates*.

Before A

.NET 7, 81, 135, 145, 194
 see also C#
=, in initialization vs. assignment 6
1066 150
2nd edition of this book
 compared to 3rd edition xv-xvi, 277-279
 see also inside back cover
3rd edition of this book
 compared to 2nd edition xv-xvi, 277-279
 see also inside back cover
80-20 rule 139, 168

A

Abrahams, David xvii, xviii, xix
abstract classes 43
accessibility
 control over data members' 95
 name, multiple inheritance and 193
accessing names, in templated
 bases 207-212
addresses
 inline functions 136
 objects 118
aggregation, see composition
Alexandrescu, Andrei xix
aliasing 54
alignment 249-250
allocators, in the STL 240

alternatives to virtual functions 169-177
ambiguity
 multiple inheritance and 192
 nested dependent names and types 205
Arbiter, Petronius vii
argument-dependent lookup 110
arithmetic, mixed-mode 103, 222-226
array layout, vs. object layout 73
array new 254-255
array, invalid index and 7
ASPECT_RATIO 13
assignment
 see also operator=
 chaining assignments 52
 copy-and-swap and 56
 generalized 220
 to self, operator= and 53-57
 vs. initialization 6, 27-29, 114
assignment operator, copy 5
auto_ptr, see std::auto_ptr
automatically generated functions 34-37
 copy constructor and copy assignment
 operator 221
 disallowing 37-39
avoiding code duplication 50, 60

B

Bai, Yun xix
Barry, Dave, allusion to 229
Bartolucci, Guido xix

- base classes
 - copying 59
 - duplication of data in 193
 - lookup in, this-> and 210, 214
 - names hidden in derived classes 263
 - polymorphic 44
 - polymorphic, destructors and 40–44
 - templated 207–212
 - virtual 193
 - basic guarantee, the 128
 - Battle of Hastings 150
 - Berck, Benjamin xix
 - bidirectional iterators 227
 - bidirectional_iterator_tag 228
 - binary upgradeability, inlining and 138
 - binding
 - dynamic, see dynamic binding
 - static, see static binding
 - birds and penguins 151–153
 - bitwise const member functions 21–22
 - books
 - C++ Programming Language, The* xvii
 - C++ Templates* xviii
 - Design Patterns* xvii
 - Effective STL* 273, 275–276
 - Exceptional C++* xvii
 - Exceptional C++ Style* xvii, xviii
 - More Effective C++* 273, 273–274
 - More Exceptional C++* xvii
 - Satyricon* vii
 - Some Must Watch While Some Must Sleep* 150
 - Boost 10, 269–272
 - containers 271
 - Conversion library 270
 - correctness and testing support 272
 - data structures 272
 - function objects and higher-order programming utilities 271
 - functionality not provided 272
 - generic programming support 271
 - Graph library 270
 - inter-language support 272
 - Lambda library 271
 - math and numerics utilities 271
 - memory management utilities 272
 - MPL library 270, 271
 - noncopyable base class 39
 - Pool library 250, 251
 - scoped_array 65, 216, 272
 - shared_array 65
 - shared_ptr implementation, costs 83
 - smart pointers 65, 272
 - web page xvii
 - string and text utilities 271
 - template metaprogramming support 271
 - TR1 and 9–10, 268, 269
 - typelist support 271
 - web site 10, 269, 272
 - boost, as synonym for std::tr1 268
 - Bosch, Derek xviii
 - breakpoints, and inlining 139
 - Buffy the Vampire Slayer* xx
 - bugs, reporting xvi
 - built-in types 26–27
 - efficiency and passing 89
 - incompatibilities with 80
- ## C
- C standard library and C++ standard library 264
 - C# 43, 76, 97, 100, 116, 118, 190
 - see also .NET
 - C++ Programming Language, The* xvii
 - C++ standard library 263–269
 - <iosfwd> and 144
 - array replacements and 75
 - C standard library and 264
 - C89 standard library and 264
 - header organization of 101
 - list template 186
 - logic_error and 113
 - set template 185
 - vector template 75
 - C++ Templates* xviii
 - C++, as language federation 11–13
 - C++0x 264
 - C++-style casts 117
 - C, as sublanguage of C++ 12
 - C99 standard library, TR1 and 267
 - caching
 - const and 22
 - mutable and 22
 - Cai, Steve xix
 - calling swap 110
 - calls to base classes, casting and 119
 - Cargill, Tom xviii
 - Carrara, Enrico xix
 - Carroll, Glenn xviii
 - casting 116–123
 - see also const_cast, static_cast, dynamic_cast, and reinterpret_cast
 - base class calls and 119
 - constness away 24–25
 - encapsulation and 123
 - grep and 117
 - syntactic forms 116–117
 - type systems and 116
 - undefined behavior and 119
 - chaining assignments 52

- Chang, Brandon xix
- Clamage, Steve xviii
- class definitions
 - artificial client dependencies, eliminating 143
 - class declarations vs. 143
 - object sizes and 141
- class design, see type design
- class names, explicitly specifying 162
- class, vs. typename 203
- classes
 - see also class definitions, interfaces
 - abstract 43, 162
 - base
 - see also base classes
 - duplication of data in 193
 - polymorphic 44
 - templated 207–212
 - virtual 193
 - defining 4
 - derived
 - see also inheritance
 - virtual base initialization of 194
 - Handle 144–145
 - Interface 145–147
 - meaning of no virtual functions 41
 - RAII, see RAII
 - specification, see interfaces
 - traits 226–232
- client 7
- clustering objects 251
- code
 - bloat 24, 135, 230
 - avoiding, in templates 212–217
 - copy assignment operator 60
 - duplication, see duplication
 - exception-safe 127–134
 - factoring out of templates 212–217
 - incorrect, efficiency and 90
 - reuse 195
 - sharing, see duplication, avoiding
- Cohen, Jake xix
- Comeau, Greg xviii
 - URL for his C/C++ FAQ xviii
- common features and inheritance 164
- commonality and variability analysis 212
- compatibility, vptrs and 42
- compatible types, accepting 218–222
- compilation dependencies 140–148
 - minimizing 140–148, 190
 - pointers, references, and objects and 143
- compiler warnings 262–263
 - calls to virtuals and 50
 - inlining and 136
 - partial copies and 58
- compiler-generated functions 34–37
 - disallowing 37–39
 - functions compilers may generate 221
- compilers
 - parsing nested dependent names 204
 - programs executing within, see template metaprogramming
 - register usage and 89
 - reordering operations 76
 - typename and 207
 - when errors are diagnosed 212
- compile-time polymorphism 201
- composition 184–186
 - meanings of 184
 - replacing private inheritance with 189
 - synonyms for 184
 - vs. private inheritance 188
- conceptual constness, see const, logical
- consistency with the built-in types 19, 86
- const 13, 17–26
 - bitwise 21–22
 - caching and 22
 - casting away 24–25
 - function declarations and 18
 - logical 22–23
 - member functions 19–25
 - duplication and 23–25
 - members, initialization of 29
 - overloading on 19–20
 - pass by reference and 86–90
 - passing `std::auto_ptr` and 220
 - pointers 17
 - return value 18
 - uses 17
 - vs. `#define` 13–14
- `const_cast` 25, 117
 - see also casting
- `const_iterator`, vs. iterators 18
- constants, see const
- constraints on interfaces, from inheritance 85
- constructors 84
 - copy 5
 - default 4
 - empty, illusion of 137
 - explicit 5, 85, 104
 - implicitly generated 34
 - inlining and 137–138
 - operator new and 137
 - possible implementation in derived classes 138
 - relationship to new 73
 - static functions and 52
 - virtual 146, 147
 - virtual functions and 48–52
 - with vs. without arguments 114
- containers, in Boost 271

containment, see composition
 continue, delete and 62
 control over data members'
 accessibility 95
 convenience functions 100
 Conversion library, in Boost 270
 conversions, type, see type conversions
 copies, partial 58
 copy assignment operator 5
 code in copy constructor and 60
 derived classes and 60
 copy constructors
 default definition 35
 derived classes and 60
 generalized 219
 how used 5
 implicitly generated 34
 pass-by-value and 6
 copy-and-swap 131
 assignment and 56
 exception-safe code and 132
 copying
 base class parts 59
 behavior, resource management
 and 66–69
 functions, the 57
 objects 57–60
 correctness
 designing interfaces for 78–83
 testing and, Boost support 272
 corresponding forms of new and
 delete 73–75
 corrupt data structures, exception-safe
 code and 127
 cows, coming home 139
 crimes against English 39, 204
 cross-DLL problem 82
 CRTP 246
 C-style casts 116
 ctor 8
 curiously recurring template pattern 246

D

dangling handles 126
 Dashtinezhad, Sasan xix
 data members
 adding, copying functions and 58
 control over accessibility 95
 protected 97
 static, initialization of 242
 why private 94–98
 data structures
 exception-safe code and 127
 in Boost 272
 Davis, Tony xviii

deadly MI diamond 193
 debuggers
 #define and 13
 inline functions and 139
 declarations 3
 inline functions 135
 replacing definitions 143
 static const integral members 14
 default constructors 4
 construction with arguments vs. 114
 implicitly generated 34
 default implementations
 for virtual functions, danger of 163–167
 of copy constructor 35
 of operator= 35
 default initialization, unintended 59
 default parameters 180–183
 impact if changed 183
 static binding of 182
 #define
 debuggers and 13
 disadvantages of 13, 16
 vs. const 13–14
 vs. inline functions 16–17
 definitions 4
 classes 4
 deliberate omission of 38
 functions 4
 implicitly generated functions 35
 objects 4
 pure virtual functions 162, 166–167
 replacing with declarations 143
 static class members 242
 static const integral members 14
 templates 4
 variable, postponing 113–116
 delete
 see also operator delete
 forms of 73–75
 operator delete and 73
 relationship to destructors 73
 usage problem scenarios 62
 delete [], std::auto_ptr and tr1::shared_ptr
 and 65
 deleters
 std::auto_ptr and 68
 tr1::shared_ptr and 68, 81–83
 Delphi 97
 Dement, William 150
 dependencies, compilation 140–148
 dependent names 204
 dereferencing a null pointer, undefined
 behavior of 6
 derived classes
 copy assignment operators and 60
 copy constructors and 60
 hiding names in base classes 263

- implementing constructors in 138
- virtual base initialization and 194
- design
 - contradiction in 179
 - of interfaces 78–83
 - of types 78–86
- Design Patterns* xvii
- design patterns
 - curiously recurring template (CRTP) 246
 - encapsulation and 173
 - generating from templates 237
 - Singleton 31
 - Strategy 171–177
 - Template Method 170
 - TMP and 237
- destructors 84
 - exceptions and 44–48
 - inlining and 137–138
 - pure virtual 43
 - relationship to delete 73
 - resource managing objects and 63
 - static functions and 52
 - virtual
 - operator delete and 255
 - polymorphic base classes and 40–44
 - virtual functions and 48–52
- Dewhurst, Steve xvii
- dimensional unit correctness, TMP and 236
- DLLs, delete and 82
- dtor 8
- Dulimov, Peter xix
- duplication
 - avoiding 23–25, 29, 50, 60, 164, 183, 212–217
 - base class data and 193
 - init function and 60
- dynamic binding
 - definition of 181
 - of virtual functions 179
- dynamic type, definition of 181
- dynamic_cast 50, 117, 120–123
 - see also casting
 - efficiency of 120

E

- early binding 180
- easy to use correctly and hard to use incorrectly 78–83
- EBO, see empty base optimization
- Effective C++*, compared to *More Effective C++* and *Effective STL* 273
- Effective STL* 273, 275–276
 - compared to *Effective C++* 273

- contents of 275–276
- efficiency
 - assignment vs. construction and destruction 94
 - default parameter binding 182
 - dynamic_cast 120
 - Handle classes 147
 - incorrect code and 90, 94
 - init. with vs. without args 114
 - Interface classes 147
 - macros vs. inline functions 16
 - member init. vs. assignment 28
 - minimizing compilation
 - dependencies 147
 - operator new/operator delete and 248
 - pass-by-reference and 87
 - pass-by-value and 86–87
 - passing built-in types and 89
 - runtime vs. compile-time tests 230
 - template metaprogramming and 233
 - template vs. function parameters 216
 - unused objects 113
 - virtual functions 168
- Eiffel 100
- embedding, see composition
- empty base optimization (EBO) 190–191
- encapsulation 95, 99
 - casts and 123
 - design patterns and 173
 - handles and 124
 - measuring 99
 - protected members and 97
 - RAII classes and 72
- enum hack 15–16, 236
- errata list, for this book xvi
- errors
 - detected during linking 39, 44
 - runtime 152
- evaluation order, of parameters 76
- example classes/templates
 - A 4
 - ABEntry 27
 - AccessLevels 95
 - Address 184
 - Airplane 164, 165, 166
 - Airport 164
 - AtomicClock 40
 - AWOV 43
 - B 4, 178, 262
 - Base 54, 118, 137, 157, 158, 159, 160, 254, 255, 259
 - BelowBottom 219
 - bidirectional_iterator_tag 228
 - Bird 151, 152, 153
 - Bitmap 54
 - BorrowableItem 192
 - Bottom 218
 - BuyTransaction 49, 51

- C 5
- Circle 181
- CompanyA 208
- CompanyB 208
- CompanyZ 209
- CostEstimate 15
- CPerson 198
- CTextBlock 21, 22, 23
- Customer 57, 58
- D 178, 262
- DatabaseID 197
- Date 58, 79
- Day 79
- DBConn 45, 47
- DBConnection 45
- deque 229
- deque::iterator 229
- Derived 54, 118, 137, 157, 158, 159, 160, 206, 254, 260
- Directory 31
- ElectronicGadget 192
- Ellipse 161
- Empty 34, 190
- EvilBadGuy 172, 174
- EyeCandyCharacter 175
- Factorial 235
- Factorial<0> 235
- File 193, 194
- FileSystem 30
- FlyingBird 152
- Font 71
- forward_iterator_tag 228
- GameCharacter 169, 170, 172, 173, 176
- GameLevel 174
- GamePlayer 14, 15
- GraphNode 4
- GUIObject 126
- HealthCalcFunc 176
- HealthCalculator 174
- HoldsAnInt 190, 191
- HomeForSale 37, 38, 39
- input_iterator_tag 228
- input_iterator_tag<lter*> 230
- InputFile 193, 194
- Investment 61, 70
- IOFile 193, 194
- IPerson 195, 197
- iterator_traits 229
 - see also std::iterator_traits
- list 229
- list::iterator 229
- Lock 66, 67, 68
- LoggingMsgSender 208, 210, 211
- Middle 218
- ModelA 164, 165, 167
- ModelB 164, 165, 167
- ModelC 164, 166, 167
- Month 79, 80
- MP3Player 192
- MsgInfo 208
- MsgSender 208
- MsgSender<CompanyZ> 209
- NamedObject 35, 36
- NewHandlerHolder 243
- NewHandlerSupport 245
- output_iterator_tag 228
- OutputFile 193, 194
- Penguin 151, 152, 153
- Person 86, 135, 140, 141, 142, 145, 146, 150, 184, 187
- PersonInfo 195, 197
- PhoneNumber 27, 184
- PMImpl 131
- Point 26, 41, 123
- PrettyMenu 127, 130, 131
- PriorityCustomer 58
- random_access_iterator_tag 228
- Rational 90, 102, 103, 105, 222, 223, 224, 225, 226
- RealPerson 147
- Rectangle 124, 125, 154, 161, 181, 183
- RectData 124
- SellTransaction 49
- Set 185
- Shape 161, 162, 163, 167, 180, 182, 183
- SmartPtr 218, 219, 220
- SpecialString 42
- SpecialWindow 119, 120, 121, 122
- SpeedDataCollection 96
- Square 154
- SquareMatrix 213, 214, 215, 216
- SquareMatrixBase 214, 215
- StandardNewDeleteForms 260
- Student 86, 150, 187
- TextBlock 20, 23, 24
- TimeKeeper 40, 41
- Timer 188
- Top 218
- Transaction 48, 50, 51
- Uncopyable 39
- WaterClock 40
- WebBrowser 98, 100, 101
- Widget 4, 5, 44, 52, 53, 54, 56, 107, 108, 109, 118, 189, 199, 201, 242, 245, 246, 257, 258, 261
- Widget::WidgetTimer 189
- WidgetImpl 106, 108
- Window 88, 119, 121, 122
- WindowWithScrollBars 88
- WristWatch 40
- X 242
- Y 242
- Year 79
- example functions/templates
 - ABEntry::ABEntry 27, 28
 - AccessLevels::getReadOnly 95
 - AccessLevels::getReadWrite 95
 - AccessLevels::setReadOnly 95

- AccessLevels::setWriteOnly 95
- advance 228, 230, 232, 233, 234
- Airplane::defaultFly 165
- Airplane::fly 164, 165, 166, 167
- askUserForDatabaseID 195
- AWOV::AWOV 43
- B::mf 178
- Base::operator delete 255
- Base::operator new 254
- Bird::fly 151
- BorrowableItem::checkOut 192
- boundingBox 126
- BuyTransaction::BuyTransaction 51
- BuyTransaction::createLogString 51
- calcHealth 174
- callWithMax 16
- changeFontSize 71
- Circle::draw 181
- clearAppointments 143, 144
- clearBrowser 98
- CPerson::birthDate 198
- CPerson::CPerson 198
- CPerson::name 198
- CPerson::valueDelimClose 198
- CPerson::valueDelimOpen 198
- createInvestment 62, 70, 81, 82, 83
- CTextBlock::length 22, 23
- CTextBlock::operator[] 21
- Customer::Customer 58
- Customer::operator= 58
- D::mf 178
- Date::Date 79
- Day::Day 79
- daysHeld 69
- DBConn::~DBConn 45, 46, 47
- DBConn::close 47
- defaultHealthCalc 172, 173
- Derived::Derived 138, 206
- Derived::mf1 160
- Derived::mf4 157
- Directory::Directory 31, 32
- doAdvance 231
- doMultiply 226
- doProcessing 200, 202
- doSomething 5, 44, 54, 110
- doSomeWork 118
- eat 151, 187
- ElectronicGadget::checkOut 192
- Empty::~Empty 34
- Empty::Empty 34
- Empty::operator= 34
- encryptPassword 114, 115
- error 152
- EvilBadGuy::EvilBadGuy 172
- f 62, 63, 64
- FlyingBird::fly 152
- Font::~Font 71
- Font::Font 71
- Font::get 71
- Font::operator FontHandle 71
- GameCharacter::doHealthValue 170
- GameCharacter::GameCharacter 172, 174, 176
- GameCharacter::healthValue 169, 170, 172, 174, 176
- GameLevel::health 174
- getFont 70
- hasAcceptableQuality 6
- HealthCalcFunc::calc 176
- HealthCalculator::operator() 174
- lock 66
- Lock::~Lock 66
- Lock::Lock 66, 68
- logCall 57
- LoggingMsgSender::sendClear 208, 210
- LogginMsgSender::sendClear 210, 211
- loseHealthQuickly 172
- loseHealthSlowly 172
- main 141, 142, 236, 241
- makeBigger 154
- makePerson 195
- max 135
- ModelA::fly 165, 167
- ModelB::fly 165, 167
- ModelC::fly 166, 167
- Month::Dec 80
- Month::Feb 80
- Month::Jan 80
- Month::Month 79, 80
- MsgSender::sendClear 208
- MsgSender::sendSecret 208
- MsgSender<CompanyZ>::sendSecret 209
- NewHandlerHolder::~NewHandlerHolder 243
- NewHandlerHolder::NewHandlerHolder 243
- NewHandlerSupport::operator new 245
- NewHandlerSupport::set_new_handler 245
- numDigits 4
- operator delete 255
- operator new 249, 252
- operator* 91, 92, 94, 105, 222, 224, 225, 226
- operator== 93
- outOfMem 240
- Penguin::fly 152
- Person::age 135
- Person::create 146, 147
- Person::name 145
- Person::Person 145
- PersonInfo::theName 196
- PersonInfo::valueDelimClose 196
- PersonInfo::valueDelimOpen 196
- PrettyMenu::changeBackground 127, 128, 130, 131
- print 20
- print2nd 204, 205
- printNameAndDisplay 88, 89
- priority 75
- PriorityCustomer::operator= 59

- PriorityCustomer::PriorityCustomer 59
 - processWidget 75
 - RealPerson::~~RealPerson 147
 - RealPerson::RealPerson 147
 - Rectangle::doDraw 183
 - Rectangle::draw 181, 183
 - Rectangle::lowerRight 124, 125
 - Rectangle::upperLeft 124, 125
 - releaseFont 70
 - Set::insert 186
 - Set::member 186
 - Set::remove 186
 - Set::size 186
 - Shape::doDraw 183
 - Shape::draw 161, 162, 180, 182, 183
 - Shape::error 161, 163
 - Shape::objectId 161, 167
 - SmartPtr::get 220
 - SmartPtr::SmartPtr 220
 - someFunc 132, 156
 - SpecialWindow::blink 122
 - SpecialWindow::onResize 119, 120
 - SquareMatrix::invert 214
 - SquareMatrix::setDataPtr 215
 - SquareMatrix::SquareMatrix 215, 216
 - StandardNewDeleteForms::operator delete 260, 261
 - StandardNewDeleteForms::operator new 260, 261
 - std::swap 109
 - std::swap<Widget> 107, 108
 - study 151, 187
 - swap 106, 109
 - tempDir 32
 - TextBlock::operator[] 20, 23, 24
 - tfs 32
 - Timer::onTick 188
 - Transaction::init 50
 - Transaction::Transaction 49, 50, 51
 - Uncopyable::operator= 39
 - Uncopyable::Uncopyable 39
 - unlock 66
 - validateStudent 87
 - Widget::onTick 189
 - Widget::operator new 244
 - Widget::operator+= 53
 - Widget::operator= 53, 54, 55, 56, 107
 - Widget::set_new_handler 243
 - Widget::swap 108
 - Window::blink 122
 - Window::onResize 119
 - workWithIterator 206, 207
 - Year::Year 79
 - exception specifications 85
 - Exceptional C++* xvii
 - Exceptional C++ Style* xvii, xviii
 - exceptions 113
 - delete and 62
 - destructors and 44–48
 - member swap and 112
 - standard hierarchy for 264
 - swallowing 46
 - unused objects and 114
 - exception-safe code 127–134
 - copy-and-swap and 132
 - legacy code and 133
 - pimpl idiom and 131
 - side effects and 132
 - exception-safety guarantees 128–129
 - explicit calls to base class functions 211
 - explicit constructors 5, 85, 104
 - generalized copy construction and 219
 - explicit inline request 135
 - explicit specification, of class names 162
 - explicit type conversions vs. implicit 70–72
 - expression templates 237
 - expressions, implicit interfaces and 201
- ## F
- factoring code, out of templates 212–217
 - factory function 40, 62, 69, 81, 146, 195
 - Fallenstedt, Martin xix
 - federation, of languages, C++ as 11–13
 - Feher, Attila F. xix
 - final classes, in Java 43
 - final methods, in Java 190
 - fixed-size static buffers, problems of 196
 - forms of new and delete 73–75
 - FORTRAN 42
 - forward iterators 227
 - forward_iterator_tag 228
 - forwarding functions 144, 160
 - French, Donald xx
 - friend functions 38, 85, 105, 135, 173, 223–225
 - vs. member functions 98–102
 - friendship
 - in real life 105
 - without needing special access rights 225
 - Fruchterman, Thomas xix
 - FUDGE_FACTOR 15
 - Fuller, John xx
 - function declarations, const in 18
 - function objects
 - definition of 6
 - higher-order programming utilities and, in Boost 271
 - functions
 - convenience 100
 - copying 57

- defining 4
- deliberately not defining 38
- factory, see factory function
- forwarding 144, 160
- implicitly generated 34–37, 221
 - disallowing 37–39
- inline, declaring 135
- member
 - templated 218–222
 - vs. non-member 104–105
- non-member
 - templates and 222–226
 - type conversions and 102–105, 222–226
- non-member non-friend, vs member 98–102
- non-virtual, meaning 168
- return values, modifying 21
- signatures, explicit interfaces and 201
- static
 - ctors and dtors and 52
- virtual, see virtual functions
- function-style casts 116

G

- Gamma, Erich xvii
- Geller, Alan xix
- generalized assignment 220
- generalized copy constructors 219
- generative programming 237
- generic programming support, in Boost 271
- get, smart pointers and 70
- goddess, see Urbano, Nancy L.
- goto, delete and 62
- Graph library, in Boost 270
- grep, casts and 117
- guarantees, exception safety 128–129
- Gutnik, Gene xix

H

- Handle classes 144–145
- handles 125
 - dangling 126
 - encapsulation and 124
 - operator[] and 126
 - returning 123–126
- has-a relationship 184
- hash tables, in TR1 266
- Hastings, Battle of 150
- Haugland, Solveig xx
- head scratching, avoiding 95
- header files, see headers

- headers
 - for declarations vs. for definitions 144
 - inline functions and 135
 - namespaces and 100
 - of C++ standard library 101
 - templates and 136
 - usage, in this book 3
- hello world, template metaprogramming and 235
- Helm, Richard xvii
- Henney, Kevlin xix
- Hicks, Cory xix
- hiding names, see name hiding
- higher-order programming and function object utilities, in Boost 271
- highlighting, in this book 5

I

- identity test 55
- if...else for types 230
- #ifdef 17
- #ifndef 17
- implementation-dependent behavior, warnings and 263
- implementations
 - decoupling from interfaces 165
 - default, danger of 163–167
 - inheritance of 161–169
 - of derived class constructors and destructors 137
 - of Interface classes 147
 - references 89
 - std::max 135
 - std::swap 106
- implicit inline request 135
- implicit interfaces 199–203
- implicit type conversions vs. explicit 70–72
- implicitly generated functions 34–37, 221
 - disallowing 37–39
- #include directives 17
 - compilation dependencies and 140
- incompatibilities, with built-in types 80
- incorrect code and efficiency 90
- infinite loop, in operator new 253
- inheritance
 - accidental 165–166
 - combining with templates 243–245
 - common features and 164
 - intuition and 151–155
 - mathematics and 155
 - mixin-style 244
 - name hiding and 156–161
 - of implementation 161–169
 - of interface 161–169

- of interface vs. implementation 161–169
 - operator new and 253–254
 - penguins and birds and 151–153
 - private 187–192
 - protected 151
 - public 150–155
 - rectangles and squares and 153–155
 - redefining non-virtual functions and 178–180
 - scopes and 156
 - sharing features and 164
 - inheritance, multiple 192–198
 - ambiguity and 192
 - combining public and private 197
 - deadly diamond 193
 - inheritance, private 214
 - combining with public 197
 - eliminating 189
 - for redefining virtual functions 197
 - meaning 187
 - vs. composition 188
 - inheritance, public
 - combining with private 197
 - is-a relationship and 150–155
 - meaning of 150
 - name hiding and 159
 - virtual inheritance and 194
 - inheritance, virtual 194
 - init function 60
 - initialization 4, 26–27
 - assignment vs. 6
 - built-in types 26–27
 - const members 29
 - const static members 14
 - default, unintended 59
 - in-class, of static const integral members 14
 - local static objects 31
 - non-local static objects 30
 - objects 26–33
 - reference members 29
 - static members 242
 - virtual base classes and 194
 - vs. assignment 27–29, 114
 - with vs. without arguments 114
 - initialization order
 - class members 29
 - importance of 31
 - non-local statics 29–33
 - inline functions
 - see also inlining
 - address of 136
 - as request to compiler 135
 - debuggers and 139
 - declaring 135
 - headers and 135
 - optimizing compilers and 134
 - recursion and 136
 - vs. #define 16–17
 - vs. macros, efficiency and 16
 - inlining 134–139
 - constructors/destructors and 137–138
 - dynamic linking and 139
 - Handle classes and 148
 - inheritance and 137–138
 - Interface classes and 148
 - library design and 138
 - recompiling and 139
 - relinking and 139
 - suggested strategy for 139
 - templates and 136
 - time of 135
 - virtual functions and 136
 - input iterators 227
 - input_iterator_tag 228
 - input_iterator_tag<lter*> 230
 - insomnia 150
 - instructions, reordering by compilers 76
 - integral types 14
 - Interface classes 145–147
 - interfaces
 - decoupling from implementations 165
 - definition of 7
 - design considerations 78–86
 - explicit, signatures and 201
 - implicit 199–203
 - expressions and 201
 - inheritance of 161–169
 - new types and 79–80
 - separating from implementations 140
 - template parameters and 199–203
 - undeclared 85
 - inter-language support, in Boost 272
 - internationalization, library support for 264
 - invalid array index, undefined behavior and 7
 - invariants
 - NVI and 171
 - over specialization 168
 - <iosfwd> 144
 - is-a relationship 150–155
 - is-implemented-in-terms-of 184–186, 187
 - istream_iterators 227
 - iterator categories 227–228
 - iterator_category 229
 - iterators as handles 125
 - iterators, vs. const_iterators 18
- J**
- Jagdhari, Emily xix
 - Janert, Philipp xix
 - Java 7, 43, 76, 81, 100, 116, 118, 142, 145, 190, 194

Johnson, Ralph xvii
 Johnson, Tim xviii, xix
 Josuttis, Nicolai M. xviii

K

Kaelbling, Mike xviii
 Kakulapati, Gunavardhan xix
 Kalenkovich, Eugene xix
 Kennedy, Glenn xix
 Kernighan, Brian xviii, xix
 Kimura, Junichi xviii
 Kirman, Jak xviii
 Kirmse, Andrew xix
 Knox, Timothy xviii, xix
 Koenig lookup 110
 Kourounis, Drosos xix
 Kreuzer, Gerhard xix

L

Laeuchli, Jesse xix
 Lambda library, in Boost 271
 Langer, Angelika xix
 languages, other, compatibility with 42
 Lanzetta, Michael xix
 late binding 180
 layering, see composition
 layouts, objects vs. arrays 73
 Lea, Doug xviii
 leaks, exception-safe code and 127
 Leary-Coutu, Chanda xx
 Lee, Sam xix
 legacy code, exception-safety and 133
 Lejter, Moises xviii, xx
 lemur, ring-tailed 196
 Lewandowski, Scott xviii
 lhs, as parameter name 8
 Li, Greg xix
 link-time errors 39, 44
 link-time inlining 135
 list 186
 local static objects
 definition of 30
 initialization of 31
 locales 264
 locks, RAI and 66–68
 logic_error class 113
 logically const member functions 22–23

M

mailing list for Scott Meyers xvi

maintenance
 common base classes and 164
 delete and 62
 managing resources, see resource management
 Manis, Vincent xix
 Marin, Alex xix
 math and numerics utilities, in Boost 271
 mathematical functions, in TR1 267
 mathematics, inheritance and 155
 matrix operations, optimizing 237
 Matthews, Leon xix
 max, std, implementation of 135
 Meadowbrooke, Chrysta xix
 meaning
 of classes without virtual functions 41
 of composition 184
 of non-virtual functions 168
 of pass-by-value 6
 of private inheritance 187
 of public inheritance 150
 of pure virtual functions 162
 of references 91
 of simple virtual functions 163
 measuring encapsulation 99
 Meehan, Jim xix
 member data, see data members
 member function templates 218–222
 member functions
 bitwise const 21–22
 common design errors 168–169
 const 19–25
 duplication and 23–25
 encapsulation and 99
 implicitly generated 34–37, 221
 disallowing 37–39
 logically const 22–23
 private 38
 protected 166
 vs. non-member functions 104–105
 vs. non-member non-friends 98–102
 member initialization
 for const static integral members 14
 lists 28–29
 vs. assignment 28–29
 order 29
 memory allocation
 arrays and 254–255
 error handling for 240–246
 memory leaks, new expressions and 256
 memory management
 functions, replacing 247–252
 multithreading and 239, 253
 utilities, in Boost 272
 metaprogramming, see template metaprogramming

Meyers, Scott
 mailing list for xvi
 web site for xvi
 mf, as identifier 9
 Michaels, Laura xviii
 Mickelsen, Denise xx
 minimizing compilation
 dependencies 140–148, 190
 Mittal, Nishant xix
 mixed-mode arithmetic 103, 104, 222–226
 mixin-style inheritance 244
 modeling is-implemented-in-terms-
 of 184–186
 modifying function return values 21
 Monty Python, allusion to 91
 Moore, Vanessa xx
More Effective C++ 273, 273–274
 compared to *Effective C++* 273
 contents of 273–274
More Exceptional C++ xvii
 Moroff, Hal xix
 MPL library, in Boost 270, 271
 multiparadigm programming language,
 C++ as 11
 multiple inheritance, see inheritance
 multithreading
 memory management routines
 and 239, 253
 non-const static objects and 32
 treatment in this book 9
 mutable 22–23
 mutexes, RAII and 66–68

N

Nagler, Eric xix
 Nahil, Julie xx
 name hiding
 inheritance and 156–161
 operators new/delete and 259–261
 using declarations and 159
 name lookup
 this-> and 210, 214
 using declarations and 211
 name shadowing, see name hiding
 names
 accessing in templatized bases 207–212
 available in both C and C++ 3
 dependent 204
 hidden by derived classes 263
 nested, dependent 204
 non-dependent 204
 namespaces 110
 headers and 100
 namespace pollution in a class 166
 Nancy, see Urbano, Nancy L.

Nauroth, Chris xix
 nested dependent names 204
 nested dependent type names, typename
 and 205
 new
 see also operator new
 expressions, memory leaks and 256
 forms of 73–75
 operator new and 73
 relationship to constructors 73
 smart pointers and 75–77
 new types, interface design and 79–80
 new-handler 240–247
 definition of 240
 deinstalling 241
 identifying 253
 new-handling functions, behavior of 241
 new-style casts 117
 noncopyable base class, in Boost 39
 non-dependent names 204
 non-local static objects, initialization
 of 30
 non-member functions
 member functions vs. 104–105
 templates and 222–226
 type conversions and 102–105, 222–226
 non-member non-friend functions 98–102
 non-type parameters 213
 non-virtual
 functions 178–180
 static binding of 178
 interface idiom, see NVI
 nothrow guarantee, the 129
 nothrow new 246
 null pointer
 deleting 255
 dereferencing 6
 set_new_handler and 241
 NVI 170–171, 183

O

object-oriented C++, as sublanguage of
 C++ 12
 object-oriented principles, encapsulation
 and 99
 objects
 alignment of 249–250
 clustering 251
 compilation dependencies and 143
 copying all parts 57–60
 defining 4
 definitions, postponing 113–116
 handles to internals of 123–126
 initialization, with vs. without
 arguments 114
 layout vs. array layout 73

- multiple addresses for 118
 - partial copies of 58
 - placing in shared memory 251
 - resource management and 61–66
 - returning, vs. references 90–94
 - size, pass-by-value and 89
 - sizes, determining 141
 - vs. variables 3
 - Oldham, Jeffrey D. xix
 - old-style casts 117
 - operations, reordering by compilers 76
 - operator delete 84
 - see also delete
 - behavior of 255
 - efficiency of 248
 - name hiding and 259–261
 - non-member, pseudocode for 255
 - placement 256–261
 - replacing 247–252
 - standard forms of 260
 - virtual destructors and 255
 - operator delete[] 84, 255
 - operator new 84
 - see also new
 - arrays and 254–255
 - bad_alloc and 246, 252
 - behavior of 252–255
 - efficiency of 248
 - infinite loop within 253
 - inheritance and 253–254
 - member, and “wrongly sized” requests 254
 - name hiding and 259–261
 - new-handling functions and 241
 - non-member, pseudocode for 252
 - out-of-memory conditions and 240–241, 252–253
 - placement 256–261
 - replacing 247–252
 - returning 0 and 246
 - standard forms of 260
 - std::bad_alloc and 246, 252
 - operator new[] 84, 254–255
 - operator() (function call operator) 6
 - operator=
 - const members and 36–37
 - default implementation 35
 - implicit generation 34
 - reference members and 36–37
 - return value of 52–53
 - self-assignment and 53–57
 - when not implicitly generated 36–37
 - operator[] 126
 - overloading on const 19–20
 - return type of 21
 - optimization
 - by compilers 94
 - during compilation 134
 - inline functions and 134
 - order
 - initialization of non-local statics 29–33
 - member initialization 29
 - ostream_iterators 227
 - other languages, compatibility with 42
 - output iterators 227
 - output_iterator_tag 228
 - overloading
 - as if...else for types 230
 - on const 19–20
 - std::swap 109
 - overrides of virtuals, preventing 189
 - ownership transfer 68
- ## P
- Pal, Balog xix
 - parameters
 - see also pass-by-value, pass-by-reference, passing small objects
 - default 180–183
 - evaluation order 76
 - non-type, for templates 213
 - type conversions and, see type conversions
 - Pareto Principle, see 80-20 rule
 - parsing problems, nested dependent names and 204
 - partial copies 58
 - partial specialization
 - function templates 109
 - std::swap 108
 - parts, of objects, copying all 57–60
 - pass-by-reference, efficiency and 87
 - pass-by-reference-to-const, vs pass-by-value 86–90
 - pass-by-value
 - copy constructor and 6
 - efficiency of 86–87
 - meaning of 6
 - object size and 89
 - vs. pass-by-reference-to-const 86–90
 - patterns
 - see design patterns
 - Pedersen, Roger E. xix
 - penguins and birds 151–153
 - performance, see efficiency
 - Persephone ix, xx, 36
 - pessimization 93
 - physical constness, see const, bitwise
 - pimpl idiom
 - definition of 106
 - exception-safe code and 131

- placement delete, see operator delete
 - placement new, see operator new
 - Plato 87
 - pointer arithmetic and undefined behavior 119
 - pointers
 - see also smart pointers
 - as handles 125
 - bitwise const member functions and 21
 - compilation dependencies and 143
 - const 17
 - in headers 14
 - null, dereferencing 6
 - template parameters and 217
 - to single vs. multiple objects, and delete 73
 - polymorphic base classes, destructors and 40–44
 - polymorphism 199–201
 - compile-time 201
 - runtime 200
 - Pool library, in Boost 250, 251
 - postponing variable definitions 113–116
 - Prasertsith, Chuti xx
 - preconditions, NVI and 171
 - pregnancy, exception-safe code and 133
 - private data members, why 94–98
 - private inheritance, see inheritance
 - private member functions 38
 - private virtual functions 171
 - properties 97
 - protected
 - data members 97
 - inheritance, see inheritance
 - member functions 166
 - members, encapsulation of 97
 - public inheritance, see inheritance
 - pun, really bad 152
 - pure virtual destructors
 - defining 43
 - implementing 43
 - pure virtual functions 43
 - defining 162, 166–167
 - meaning 162
- R**
- Rabbani, Danny xix
 - Rabinowitz, Marty xx
 - RAII 66, 70, 243
 - classes 72
 - copying behavior and 66–69
 - encapsulation and 72
 - mutexes and 66–68
 - random access iterators 227
 - random number generation, in TR1 267
 - random_access_iterator_tag 228
 - RCSP, see smart pointers
 - reading uninitialized values 26
 - rectangles and squares 153–155
 - recursive functions, inlining and 136
 - redefining inherited non-virtual functions 178–180
 - Reed, Kathy xx
 - Reeves, Jack xix
 - references
 - as handles 125
 - compilation dependencies and 143
 - functions returning 31
 - implementation 89
 - meaning 91
 - members, initialization of 29
 - returning 90–94
 - to static object, as function return value 92–94
 - register usage, objects and 89
 - regular expressions, in TR1 266
 - reinterpret_cast 117, 249
 - see also casting
 - relationships
 - has-a 184
 - is-a 150–155
 - is-implemented-in-terms-of 184–186, 187
 - reordering operations, by compilers 76
 - replacing definitions with
 - declarations 143
 - replacing new/delete 247–252
 - replication, see duplication
 - reporting, bugs in this book xvi
 - Resource Acquisition Is Initialization, see RAI
 - resource leaks, exception-safe code and 127
 - resource management
 - see also RAI
 - copying behavior and 66–69
 - objects and 61–66
 - raw resource access and 69–73
 - resources, managing objects and 69–73
 - return by reference 90–94
 - return types
 - const 18
 - objects vs. references 90–94
 - of operator[] 21
 - return value of operator= 52–53
 - returning handles 123–126
 - reuse, see code reuse
 - revenge, compilers taking 58
 - rhs, as parameter name 8

Roze, Mike xix
 rule of 80-20 139, 168
 runtime
 errors 152
 inlining 135
 polymorphism 200

S

Saks, Dan xviii
 Santos, Eugene, Jr. xviii
 Satch 36
Satyricon vii
 Scherpelz, Jeff xix
 Schirripa, Steve xix
 Schober, Hendrik xviii, xix
 Schroeder, Sandra xx
`scoped_array` 65, 216, 272
 scopes, inheritance and 156
 sealed classes, in C# 43
 sealed methods, in C# 190
 second edition, see 2nd edition
 self-assignment, operator= and 53-57
 set 185
`set_new_handler`
 class-specific, implementing 243-245
 using 240-246
`set_unexpected` function 129
 shadowing, names, see name shadowing
 Shakespeare, William 156
 shared memory, placing objects in 251
`shared_array` 65
`shared_ptr` implementation in Boost,
 costs 83
 sharing code, see duplication, avoiding
 sharing common features 164
 Shewchuk, John xviii
 side effects, exception safety and 132
 signatures
 definition of 3
 explicit interfaces and 201
 simple virtual functions, meaning of 163
 Singh, Siddhartha xix
 Singleton pattern 31
`size_t` 3
`sizeof` 253, 254
 empty classes and 190
 freestanding classes and 254
 sizes
 of freestanding classes 254
 of objects 141
 sleeping pills 150
`slist` 227
 Smallberg, David xviii, xix
 Smalltalk 142
 smart pointers 63, 64, 70, 81, 121, 146, 237
 see also `std::auto_ptr` and `tr1::shared_ptr`
 get and 70
 in Boost 65, 272
 web page for xvii
 in TR1 265
 newed objects and 75-77
 type conversions and 218-220
 Socrates 87
Some Must Watch While Some Must Sleep 150
 Somers, Jeff xix
 specialization
 invariants over 168
 partial, of `std::swap` 108
 total, of `std::swap` 107, 108
 specification, see interfaces
 squares and rectangles 153-155
 standard exception hierarchy 264
 standard forms of operator new/delete 260
 standard library, see C++ standard
 library, C standard library
 standard template library, see STL
 Stasko, John xviii
 statements using new, smart pointers
 and 75-77
 static
 binding
 of default parameters 182
 of non-virtual functions 178
 objects, returning references to 92-94
 type, definition of 180
 static functions, ctors and dtors and 52
 static members
 const member functions and 21
 definition 242
 initialization 242
 static objects
 definition of 30
 multithreading and 32
`static_cast` 25, 82, 117, 119, 249
 see also casting
`std` namespace, specializing templates
 in 107
`std::auto_ptr` 63-65, 70
 conversion to `tr1::shared_ptr` and 220
 delete [] and 65
 pass by const and 220
`std::auto_ptr`, deleter support and 68
`std::char_traits` 232
`std::iterator_traits`, pointers and 230
`std::list` 186
`std::max`, implementation of 135
`std::numeric_limits` 232

- std::set 185
 - std::size_t 3
 - std::swap
 - see also swap
 - implementation of 106
 - overloading 109
 - partial specialization of 108
 - total specialization of 107, 108
 - std::tr1, see TR1
 - stepping through functions, inlining
 - and 139
 - STL
 - allocators 240
 - as sublanguage of C++ 12
 - containers, swap and 108
 - definition of 6
 - iterator categories in 227–228
 - Strategy pattern 171–177
 - string and text utilities, in Boost 271
 - strong guarantee, the 128
 - Stroustrup, Bjarne xvii, xviii
 - Stroustrup, Nicholas xix
 - Sutter, Herb xvii, xviii, xix
 - swallowing exceptions 46
 - swap 106–112
 - see also std::swap
 - calling 110
 - exceptions and 112
 - STL containers and 108
 - when to write 111
 - symbols, available in both C and C++ 3
- T**
- template C++, as sublanguage of C++ 12
 - template metaprogramming 233–238
 - efficiency and 233
 - hello world in 235
 - pattern implementations and 237
 - support in Boost 271
 - support in TR1 267
 - Template Method pattern 170
 - templates
 - code bloat, avoiding in 212–217
 - combining with inheritance 243–245
 - defining 4
 - errors, when detected 212
 - expression 237
 - headers and 136
 - in std, specializing 107
 - inlining and 136
 - instantiation of 222
 - member functions 218–222
 - names in base classes and 207–212
 - non-type parameters 213
 - parameters, omitting 224
 - pointer type parameters and 217
 - shorthand for 224
 - specializations 229, 235
 - partial 109, 230
 - total 107, 209
 - type conversions and 222–226
 - type deduction for 223
 - temporary objects, eliminated by
 - compilers 94
 - terminology, used in this book 3–8
 - testing and correctness, Boost support
 - for 272
 - text and string utilities, in Boost 271
 - third edition, see 3rd edition
 - this->, to force base class lookup 210, 214
 - threading, see multithreading
 - Tilly, Barbara xviii
 - TMP, see template metaprogramming
 - Tondo, Clovis xviii
 - Topic, Michael xix
 - total class template specialization 209
 - total specialization of std::swap 107, 108
 - total template specializations 107
 - TR1 9, 264–267
 - array component 267
 - bind component 266
 - Boost and 9–10, 268, 269
 - boost as synonym for std::tr1 268
 - C99 compatibility component 267
 - function component 265
 - hash tables component 266
 - math functions component 267
 - mem_fn component 267
 - random numbers component 267
 - reference_wrapper component 267
 - regular expression component 266
 - result_of component 267
 - smart pointers component 265
 - support for TMP 267
 - tuples component 266
 - type traits component 267
 - URL for information on 268
 - tr1::array 267
 - tr1::bind 175, 266
 - tr1::function 173–175, 265
 - tr1::mem_fn 267
 - tr1::reference_wrapper 267
 - tr1::result_of 267
 - tr1::shared_ptr 53, 64–65, 70, 75–77
 - construction from other smart pointers
 - and 220
 - cross-DLL problem and 82
 - delete [] and 65
 - deleter support in 68, 81–83
 - member template ctors in 220–221
 - tr1::tuple 266

tr1::unordered_map 43, 266
 tr1::unordered_multimap 266
 tr1::unordered_multiset 266
 tr1::unordered_set 266
 tr1::weak_ptr 265
 traits classes 226–232
 transfer, ownership 68
 translation unit, definition of 30
 Trux, Antoine xviii
 Tsao, Mike xix
 tuples, in TR1 266
 type conversions 85, 104
 explicit ctors and 5
 implicit 104
 implicit vs. explicit 70–72
 non-member functions and 102–105, 222–226
 private inheritance and 187
 smart pointers and 218–220
 templates and 222–226
 type deduction, for templates 223
 type design 78–86
 type traits, in TR1 267
 typedef, typename and 206–207
 typedefs, new/delete and 75
 typeid 50, 230, 234, 235
 typelists 271
 typename 203–207
 compiler variations and 207
 typedef and 206–207
 vs. class 203
 types
 built-in, initialization 26–27
 compatible, accepting all 218–222
 if...else for 230
 integral, definition of 14
 traits classes and 226–232

U

undeclared interface 85
 undefined behavior
 advance and 231
 array deletion and 73
 casting + pointer arithmetic and 119
 definition of 6
 destroyed objects and 91
 exceptions and 45
 initialization order and 30
 invalid array index and 7
 multiple deletes and 63, 247
 null pointers and 6
 object deletion and 41, 43, 74
 uninitialized values and 26
 undefined values of members before construction and after destruction 50

unexpected function 129
 uninitialized
 data members, virtual functions and 49
 values, reading 26
 unnecessary objects, avoiding 115
 unused objects
 cost of 113
 exceptions and 114
 Urbano, Nancy L. vii, xviii, xx
 see also goddess
 URLs
 Boost 10, 269, 272
 Boost smart pointers xvii
 Effective C++ errata list xvi
 Effective C++ TR1 Info. Page 268
 Greg Comeau's C/C++ FAQ xviii
 Scott Meyers' mailing list xvi
 Scott Meyers' web site xvi
 this book's errata list xvi
 usage statistics, memory management and 248
 using declarations
 name hiding and 159
 name lookup and 211

V

valarray 264
 value, pass by, see pass-by-value
 Van Wyk, Chris xviii, xix
 Vandevoorde, David xviii
 variable, vs. object 3
 variables definitions, postponing 113–116
 vector template 75
 Viciano, Paco xix
 virtual base classes 193
 virtual constructors 146, 147
 virtual destructors
 operator delete and 255
 polymorphic base classes and 40–44
 virtual functions
 alternatives to 169–177
 ctors/dtors and 48–52
 default implementations and 163–167
 default parameters and 180–183
 dynamic binding of 179
 efficiency and 168
 explicit base class qualification and 211
 implementation 42
 inlining and 136
 language interoperability and 42
 meaning of none in class 41
 preventing overrides 189
 private 171
 pure, see pure virtual functions
 simple, meaning of 163

- uninitialized data members and 49
- virtual inheritance, see inheritance
- virtual table 42
- virtual table pointer 42
- Vlissides, John xvii
- vptr 42
- vtbl 42

W

- Wait, John xx
- warnings, from compiler 262–263
 - calls to virtuals and 50
 - inlining and 136
 - partial copies and 58
- web sites, see URLs
- Widget class, as used in this book 8
- Wiegers, Karl xix
- Wilson, Matthew xix
- Wizard of Oz*, allusion to 154

X

- XP, allusion to 225
- XYZ Airlines 163

Z

- Zabluda, Oleg xviii
- Zolman, Leor xviii, xix