

Index

\ (backslash), **FindFirstFile/FindNextFile** API, 188–189
/ (slash), **FindFirstFile/FindNextFile** API, 188–189

A

`absolutePath` flag, 124–131, 158–161
abstraction
 container type, 233–234
 file systems, 99–102
 leaky, 40–41, 64
 platform-dependent variables. *See* Environmental mapping.
 string handling, 98–99
access shims, 52–56
`accumulate()` method, 349, 507–509
ACE (Adaptive Communications Environment), 414–420
ACE Reactor framework, 414–420
`Ace_Message_Block` communications
 block transfer validity, 421–423
 description, 420
 iterator worker methods, 422
 message_queue_sequence class, 421–423
 performance, 420–421, 426–427
 shared_handle worker methods, 423
 standard library, specializing, 424–426
ACESTL subproject, 415–416, xxxv
adapted iterators traits, 533–535, 540. *See also* Iterator adaptors.
`adapted_iterator_traits` class
 `const_pointer` member, 538
 `const_reference` member, 538
 description, 533–535
 `difference_type` member, 536–537
 `effective_const_pointer` member, 539
 `effective_const_reference` member, 539
 `effective_pointer` member, 539
 `effective_reference` member, 539
 `iterator_category` member, 536

`pointer` member, 537
 `reference` member, 537–538
 using, 540
 `value_type` member, 536
Adaptive Communications Environment (ACE), 414–420
adaptor class templates
 description, 77–78
 IIA (inferred interface adaptation)
 applying, 85–86
 description, 80–81
 immutable collections, 79–80
 interface-incomplete types, 78–79
 SFINAE, 82–83
 type detection, 82–83
 type fixing, 83–85
 type selection, 81–82
adaptors
 Class Adaptors
 constructors, 236
 definition, 3, 227
 reverse iterator type, defining, 11
 templates, 4
 definition, 3
 Instance Adaptors
 class template, 244–245
 constructors, 236
 definition, 3, 227
 input/output stream adaptation, 12
 stream buffer adaptation, 12
 templates, 4
`advance()` method, 378–379
algorithms, 3, 12–13. *See also* Function objects; Iterators.
allocators, 3, 13
`allocator_selector` template, 75
APIs
 callback enumeration APIs, 17
 element access, 15–16
 element-at-a-time, 16, 19
 elements-en-bloc, 16
 file system enumeration
 UNIX. *See* **Glob** API;
 Opendir/readdir API.
 Windows. *See* **FindFirstFile/FindNextFile** API; **WinInet** API.
 FTP server directories. *See* **WinInet** API.
 modules. *See* **Process State (PSAPI)** API.
 processes. *See* **Process State (PSAPI)** API.

Registry, 444–446
 Scatter/Gather I/O, 409–414
 for windows. *See* **USER** API.
argument-dependent return-type variance (ARV). *See* ARV (argument-dependent return-type variance).
arrangement-immutable collections, 17
arrangement-mutable collections, 17
array adaptor classes, 234–235
array containers. *See* **CArray** container family.
array dimensions, DRY SPOT principle, 35
ARV (argument-dependent return-type variance)
 dual-semantic subscripting, 430–431
 generalized compatibility, 431–433
 overloadtype, selecting, 433–434
 return type, selecting, 433–434
 Ruby code, 428–430
 signed subscript indexes, 434
`assert()` macro, 44
`assign()` method, 257–258
associative containers, 4
attribute shims, 49
`auto_buffer` class template, 93–97
B
backslash (\), **FindFirstFile/FindNextFile** API, 188–189
base types, 69–70, 311, 549
`base_type_traits` class, 69–70
`basic_file_path_buffer` class
 template, 103–107, 131, 160, 174–176
`basic_findfile_sequence` class
 attribute methods, 174
 class interface, 172–175
 code listing, 166–167
 constructors/destructor, 173–174
 constructors/destructors, 197
 implementation methods, 174
 invariant methods, 174
 iteration methods, 174
 member types, 172–173
 member variables, 175
 state methods, 174
`basic_findfile_sequence_const_iterator` class, 178–180

- basic_findfile_sequence_value_type class, 191–193
 - basic_ftplib_sequence class, 199–200
 - basic_key_reg_sequence class, 451–463
 - basic_key_reg_sequence_iterator class, 456–459
 - basic_string class template, 4
 - begin() method, 94–95, 129, 147–148, 177, 246–247
 - bidirectional iteration
 - description, 8–9
 - environment mapping, 279–282
 - semantics, 547–548
 - Z-plane travel, 304–306, 309–314
 - blanks in strings, preserving, 342–343
 - Boost libraries, 331, xxxvi
 - by-value temporary element references, 28–33

 - C**
 - C++ libraries, signs of success
 - discoverability, xxvii–xxviii
 - efficiency, xxvii
 - expressiveness, xxviii–xxx
 - flexibility, xxx–xxxii
 - modularity, xxxi
 - portability, xxxii
 - robustness, xxx
 - transparency, xxvii–xxviii
 - C++ references, 21–22. *See also* Element reference categories.
 - callback enumeration APIs, 17
 - capacity() method, 252–253
 - capacity of containers, 252–253
 - CArray class template, 232–233
 - CArray container family
 - abstract state manipulation, 235
 - abstracting container type, 233–234
 - array adaptor classes, 234–235
 - assignment, 235–236, 261–264
 - base and derived classes, 241
 - begin() method, 246–247
 - capacity, 252–253
 - capacity() method, 252–253
 - CArray class template, 232–233
 - CArray_adaptor_base class template
 - comparison methods, 253–256
 - constructors, 245
 - CRTP, 241
 - element access methods, 245
 - interface, 237–239
 - iteration, 246–248
 - reverse iteration, 248
 - size() methods, 248–250
 - CArray_cadaptor class template
 - constructors, 242–243
 - description, 239–240
 - parameters, 240–241
 - CArray_iadaptor class template, 244–245
 - CArray_traits class, 233–234
 - cdecl calling convention, xl, 109–110
 - character set delimiters, 344–345
 - child_window_sequence class, 301–302
 - CArray_iadaptor class
 - template, 244–245, 254, 261–264
 - CArray_traits class, 233–234
 - class adaptation, 239–240
 - Class Adaptors, 227, 236
 - collection-interface composition, 236–237
 - comparison, 253–256
 - construction, 242–243
 - container modifiers
 - assign() method, 257–258
 - clear() method, 258–259
 - erase() method, 259–260
 - insert() method, 260–261
 - pop_back() method, 258–259
 - push_back() method, 256–257
 - copy-and-swap idiom, 235–236
 - CRTP (Curiously Recurring Template Pattern), 241
 - description, 232–233
 - design considerations, 232–235
 - element access methods, 245
 - end() method, 246–247
 - inheritance, 240–241
 - Instance Adaptors, 227, 236, 244–245
 - iteration, 246–248
 - memory allocation, 245–246, 250–252
 - method names, resolving, 243–244
 - operator [] (subscript operator), 243–244
 - rbegin() method, 247–248
 - rend() method, 247–248
 - reserve() method, 252–253
 - size, 248–250
 - swap() method, 262–264
 - template declaration, 240–241
 - uses for, 227–230
 - vector class template, emulating, 230–232
- CArray_adaptor_base class template
 - comparison methods, 253–256
 - constructors, 245
 - CRTP, 241
 - element access methods, 245
 - interface, 237–239
 - iteration, 246–248
 - reverse iteration, 248
 - size() methods, 248–250
- CArray_cadaptor class template
 - constructors, 242–243
 - description, 239–240
 - parameters, 240–241
- CArray_iadaptor class template, 244–245
- CArray_traits class, 233–234
- cdecl calling convention, xl, 109–110
- character set delimiters, 344–345
- child_window_sequence class, 301–302
- Class Adaptors
 - constructors, 236
 - definition, 3, 227
 - reverse iterator type, defining, 11
 - templates, 4
- class invariants, 42–43, 127–128, 174, 177–178, 376, 379–380
- class templates
 - adaptor
 - description, 77–78
 - IIA (inferred interface adaptation), 80–81, 85–86
 - immutable collections, 79–80
 - interface-incomplete types, 78–79
 - SFINAE, 82–83
 - type detection, 82–83
 - type fixing, 83–85
 - type selection, 81–82
 - auto_buffer, 93–97
 - basic_file_path_buffer, 103–107
 - basic_string, 4
 - CArray, 232–233
 - CArray_adaptor_base
 - comparison methods, 253–256
 - constructors, 245
 - CRTP, 241
 - element access methods, 245
 - interface, 237–239
 - iteration, 246–248
 - reverse iteration, 248
 - size() methods, 248–250
 - CArray_cadaptor, 239–243
 - CArray_iadaptor, 244–245
 - collection_sequence, 398–403
 - deque, 4
 - filesystem_traits, 97–103
 - find_next_token(), 330
 - list, 4
 - listbox_operation_traits, 440–444
 - priority_queue, 4
 - queue, 4
 - scatter_slice_sequence, 411–414
 - scoped_handle, 107–109
 - stack, 4
 - string_tokeniser, 331–333, 436
 - string_tokeniser_comparator, 338–340
 - string_tokeniser_type_traits, 337–338
 - StringTokenizer, 327–328
 - tokenizer, 331
 - vector, 4, 230–232
- classes
 - adaptation, 239–240
 - adapted_iterator_traits, 533–540
 - array adaptor, 234–235
 - base_type_traits, 69–70

- basic_findfile_sequence
 - attribute methods, 174
 - class interface, 172–175
 - code listing, 166–167
 - constructors/destructors, 173–174, 197
 - implementation methods, 174
 - invariant methods, 174
 - iteration methods, 174
 - member types, 172–173
 - member variables, 175
 - state methods, 174
- basic_findfile_sequence_
 - sequence_
 - const_iterator, 178–180
- basic_findfile_sequence_value_type, 191–193
- basic_ftpdir_sequence, 199–200
- basic_key_reg_sequence, 451–463
- basic_key_reg_sequence_iterator, 456–459
- CArray_traits, 233–234
- child_window_sequence, 301–302
- cloneable_cloning_policy, 384–385
- combobox_sequence, 440–444
- const_iterator (readdir). *See also* Readdir_sequence class.
 - absolutePath flag, 158–161
 - definition, 156
 - environment mapping, 289–290
 - fullPath flag, 158–161
 - iterator types, 157–158
 - operator * (dereference operator), 160
 - prepare_directory() method, 160–161
 - version 1, 148–153
 - version 2, 154–156
- const_iterator (stlsoft::string_tokenizer), 334–336
- converting. *See* Adaptor class templates.
- cstring_concatenator_iterator, 522–525, 527–532
- enumeration_context
 - advance() method, 378–379
 - class invariant, 379–380
 - construction methods, 375–378
 - definition, 373–374
 - description, 372
 - is_valid() method, 379–380
 - iterator methods, 378–379
 - uses for, 372–373
- enumerator_sequence
 - breaking value semantics, 367
 - clear() method, 361–363
 - clear_elements() method, 362–363
 - cloning policy defaults, 385–390
 - cloning_policy_type
 - member type, 360
 - const-incorrect iterator methods, 366–367
 - construction methods, 364–366
 - copy() method, 361–363
 - empty() method, 390
 - enumeration_context
 - class, 362–363
 - init() method, 361–363
 - init_elements() method, 362–363
 - interface_type
 - member type, 360
 - iteration methods, 366
 - member variables, 364
 - pointer
 - member type, 360, 392–393
 - public interface, 359–360
 - quanta parameter, 360, 365
 - reference member type, 360
 - validate_quanta() method, 365–366
 - value policies, 361–363
 - value_policy_adaptor
 - class, 363
 - value_type member type, 360
- environment_variable_traits, 268–271
- external_iterator
 - invalidation, 307–308
- filter_iterator, 545–548
- findfile_sequence, 198
- fixed array classes, 24
- forward_cloning_policy, 383–384
- glob_sequence. *See also* Glob_API.
 - absolute path processing, 130–131
 - code decomposition, 135
 - code listing, 116–117
 - constructors, 126–127, 137–139
 - copying entries, 132
 - declaration of, 122–123
 - description, 121
 - destructors, 126–127
 - dots directories, eliding, 132–133
 - element access, 128
 - element-interface coherence, 436
 - filtering files and directories, 133–134
 - flags, 124–126, 131–132
 - init_glob_() worker function, 129–134
 - iteration methods, 128–129
 - member constants, 124–126
 - member types, 123
 - member variables, 123–124
 - public interface, 121–123
 - size methods, 128
 - sorting elements, 134
 - input_cloning_policy, 381–383
- invariants, 42–43
- is_fundamental_type, 73–74
- is_integral_type, 71–73
- is_same_type, 74
- is_signed_type, 73
- iterator, 368–372
- listbox_const_iterator, 440–444
- listbox_sequence, 440–444
- message_queue_sequence, 415–416, 421–423
- MFC (Microsoft Foundation Classes) library, adapting to STL. *See* CArray container family.
- monitored_shared_handle, 459–461
- new_enum_by_dispid_policy, 405–406
- pid_sequence, 202–203
- process_module_sequence, 206
- readdir_sequence. *See also* Const_iterator class; **Opendir/readdir** API.
 - attribute methods, 148
 - constants, 146
 - construction, 146–147
 - copy semantics, 154–156
 - description, 144–146
 - element-interface coherence, 436
 - filtering files and directories, 159
 - iteration, 148–153
 - member types, 146
 - operator -> (member selection operator), 158
 - operator ++ (preincrement operator), 156–157
 - paths, 158–161
 - shared_handle class, 155, 157
 - shared_handle, 155, 157, 418–420
 - sign_traits, 70–71
 - snapshot, 295–296
 - transform_iterator
 - arithmetic operators, 490
 - comparison operators, 490
 - construction, 489
 - decrement operators, 489–490
 - defining, 487–489, 491–494, 494–496
 - increment operators, 489–490
 - pointer arithmetic methods, 489–490
 - value_policy_adaptor, 363
 - window_peer_sequence, 301, 303–304, 306, 321–322
- wrapper, 65
- zorder_iterator
 - bidirectional iteration, 305–306

- classes (*cont.*)
 - definition, 302–303
 - double dereference, 311–313, 321
 - element-interface coherence, 436
 - reverse iteration, 315
 - traits, 315–317
 - `zorder_iterator_`
 - `forward_traits`, 315–316
 - `zorder_iterator_`
 - `reverse_traits`, 316–317
- cleanup, 43, 107–109, 129–132, 189
- `clear()` method, 258–259, 361–363
- `clear_elements()` method, 362–363
- `Clone()` method, 356–357
- `cloneable_cloning_policy` class, 384–385
- cloning
 - enumerator instances, 356–357
 - policy defaults, 385–390
- `cloning_policy_type` member type, 360
- closed namespaces, *DRY SPOT* principle, 38–39
- collections. *See also* Containers.
 - adapting. *See* `CArray` container family.
 - arrangement-immutable, 17
 - arrangement-mutable, 17
 - callback enumeration APIs, 17
 - contiguous, 17
 - definition, 15
 - element access APIs, 15–16
 - element-at-a-time API, 16, 19
 - element-immutable, 17
 - element-mutable, 17
 - elements-en-bloc API, 16
 - iterators, 91–92
 - mutability, 17–18
 - notional, 15
 - STL, 16
 - terminology, 15–16
- `collection_sequence` class
 - template
 - constants, 399
 - construction methods, 399–400
 - `enum_sequence_type` member type, 399
 - iteration methods, 400–401
 - member types, 399
 - public interface, 398
 - `size()` method, 402–403
- COM collections, adapting
 - code listing, 394–397
 - `collection_sequence` class
 - template
 - constants, 399
 - construction methods, 399–400
 - `enum_sequence_type` member type, 399
 - iteration methods, 400–401
 - member types, 399
- public interface, 398
 - `size()` method, 402–403
- enumerator acquisition policies, 403–406
 - `_NewEnum` member, 403–406
 - `new_enum_by_dispid_policy` class, 405–406
 - `new_enum_method_policy` method, 404–405
 - `new_enum_property_policy` property, 404
- COM enumeration
 - cloning enumerator instances, 356–357
 - code listings, 354–355, 358–359
 - constants, defining, 360
 - different type values, 357–358
 - enumeration context, 372–373C
 - `enumeration_context` class
 - `advance()` method, 378–379
 - class invariant, 379–380
 - construction methods, 375–378
 - definition, 373–374
 - description, 372
 - `is_valid()` method, 379–380
 - iterator methods, 378–379
 - uses for, 372–373
 - `enumerator_sequence` class
 - breaking value semantics, 367
 - `clear()` method, 361–363
 - `clear_elements()` method, 362–363
 - cloning policy defaults, 385–390
 - `cloning_policy_type` member type, 360
 - `const-incorrect` iterator methods, 366–367
 - construction methods, 364–366
 - `copy()` method, 361–363
 - `empty()` method, 390
 - `enumeration_context` class, 372–373
 - `init()` method, 361–363
 - `init_elements()` method, 362–363
 - `interface_type` member type, 360
 - iteration methods, 366
 - member variables, 364
 - pointer member type, 360, 392–393
 - public interface, 359–360
 - quanta parameter, 360, 365
 - reference member type, 360
 - `validate_quanta()` method, 365–366
 - value policies, 361–363
 - `value_policy_adaptor` class, 363
 - `value_type` member type, 360
- `IEnumFileEntry` interface
 - `Clone()` method, 356–357
 - definition, 355–356
 - `Next()` method, 356
 - `Reset()` method, 356
 - `Skip()` method, 356
- IEnumXXX** protocol, 356–358
- iterator class
 - construction methods, 369–370
 - definition, 368–369
 - `equal()` method, 371–372
 - iteration methods, 370–371
 - iterator comparison, 371–372
 - `make_clone()` method, 369–370
- iterator cloning policies
 - choosing a default, 385–391
 - `cloneable_cloning_policy` class, 384–385
 - description, 381
 - forward iteration, 381–385
 - `forward_cloning_policy` class, 383–384
 - `input_cloning_policy` class, 381–383
 - interface, 381
 - member types, defining, 360
 - resetting enumeration point, 356
 - retrieving elements, 356, 359–360
 - skipping elements, 356
 - uses for, 353–355
- ComboBox controls, 437–444
- `comboBox_sequence` class, 440–444
- comparison operators, 253–256
- composite shims, 52–56
- composite string delimiter, 342
- COMSTL** subproject
 - `cloneable_cloning_policy` class, 384–385
 - `collection_sequence` class
 - template
 - constants, 399
 - construction methods, 399–400
 - `enum_sequence_type` member type, 399
 - iteration methods, 400–401
 - member types, 399
 - public interface, 398
 - `size()` method, 402–403
 - description, xxxiv–xxxv
 - `enumerator_sequence` class
 - template, 359–360
 - `forward_cloning_policy` class, 383–384
 - `input_cloning_policy` class, 381–383
- concatenating
 - string objects, 527–532
 - strings, 519–525
- conformance
 - Duck Rule, 60–61
 - Duck Typing, 60–61
 - explicit semantics, 62–63
 - Goose Rule, 61, 64
 - intersecting, 64

- leaky abstractions, 64
- member type tagging, 63
- named, 57–59, 62–63
- shims, 63
- structural, 59–61, 62–63
- const members, selecting, 517–518
- const-incorrect iterator methods, 366–367
- const_iterator class
 - (readdir)
 - absolutePath flag, 158–161
 - definition, 156
 - environment mapping, 289–290
 - fullPath flag, 158–161
 - iterator types, 157–158
 - operator * (dereference operator), 160
 - prepare_directory() method, 160–161
 - version 1, 148–153
 - version 2, 154–156
- const_iterator class (stlsoft::string tokenizer), 334–336
- const_pointer member, 538
- constraints, 45–47
- const_reference member, 538
- constructor-bound range, 220–223
- containers. *See also* Collections.
 - associative, 4
 - basic_string class template, 4
 - contiguity of storage, 4–5
 - contiguous, 16
 - definition, 16
 - deque class template, 4
 - element references, 22–23
 - list class template, 4
 - map, 4
 - modifying
 - assign() method, 257–258
 - clear() method, 258–259
 - erase() method, 259–260
 - insert() method, 260–261
 - pop_back() method, 258–259
 - push_back() method, 256–257
 - multimap, 4
 - multiset, 4
 - mutability, 17–18
 - priority_queue class template, 4
 - queue class template, 4
 - sequence, 4
 - set, 4
 - stack class template, 4
 - standard, 16
 - standard-conformant, 16
 - STL, 16
 - swapping internal states, 5
 - vector class template, 4
- contiguity of storage, 4–5
- contiguous collections, 17
- contiguous containers, 16
- contiguous iteration, 19–20, 278–279
- contract programming
 - class invariants, 42–43
 - contract violations, 42–43
 - definition, 42
 - detection, 43–44
 - enforcement mechanisms, 43–44
 - enforcement types, 42–43
 - Fibonacci sequence, 216
 - preconditions, 42–43
 - post conditions, 42–43
 - reporting, 43–44
 - response, 43–44
- contract violations, 42–43
- control shims, 48
- conversion shims, 49–52
- converting classes and interfaces. *See* Adaptor class templates.
- copy() method, 361–363
- copy-and-swap idiom, 235–236
- copying *versus* referencing, 343–344
- create_shared_handle() method, 455, 461
- creator functions, 12, 35–36, 110, 486, 501, 503–505, 510–518, 524–525, 528–529, 542, 544, 552–554
- CRTP (Curiously Recurring Template Pattern), 241
- cstring_concatenator() function, 524–525
- cstring_concatenator_iterator class, 522–525, 527–532
- curious untemporary references, 32–33. *See also* Element reference categories.
- Curiously Recurring Template Pattern (CRTP), 241
- D**
- deque class template, 4
- Dereference Proxy** pattern, 477–480
- dialecticism, xxxii–xxxiii
- difference_type member, 536–537
- dimensionof operator, 35
- directories
 - dot, eliding from search, 118, 120, 132–133
 - enumeration, UNIX APIs. *See* **Glob** API; **Opendir/readdir** API.
 - filtering, 133–134
 - full path, determining, 176–178
 - limiting total size, 171
 - trailing separators, 176–178
 - validating, 176–178
- discoverability, C++, xxvii–xxviii
- discoverability failure, 218
- distance() algorithm, 12
- dl_call() functions, 109–110
- Don't Repeat Yourself (DRY), 34
- double dereference, 311–313, 321
- drives, 171
- DRY SPOT principle
 - array dimensions, 35
 - closed namespaces, 38–39
 - constants, 34
 - creator functions, 35–36
 - definition, 34
 - function return types, 37–38
 - parent classes, 36–37
- Duck Rule, 60–61. *See also* Conformance.
- Duck Typing, 60–61. *See also* Conformance.
- E**
- effective_const_pointer member, 539
- effective_const_reference member, 539
- effective_pointer member, 539
- effective_reference member, 539
- efficiency
 - C++, xxvii
 - versus* usability, 196–200
- element access APIs, 15–16
- element reference categories
 - by-value temporary, 28–33
 - C++ references, 21–22
 - compile-time detection, 29–30
 - container elements, 22–23
 - definition, 21
 - fixed, 24
 - fixed array classes, 24
 - invalidatable, 24–26
 - operator -> (member selection operator), 31
 - permanent, 23
 - referents, 21–22
 - transient, 26–27
 - undefined iterator behavior, 30–31
 - untemporary references, 32–33
 - void, 29, 31
- element reference category, 18–19. *See also* Iterator category.
- element-at-a-time API, 16, 19
- element-immutable collections, 17
- element-interface coherence, 435–437
- element-mutable collections, 17
- elements
 - accessing. *See* Iterators.
 - key lookup, 4
 - ranges, 12–13
 - referencing. *See* Element reference categories.
- elements-en-bloc API, 16
- empty() method
 - as adjective, 61. *See* conformance
 - CArray_adaptor_base class, 249
- container requirements, 16

- empty() method (*cont.*)
 - enumerator_sequence class, 390
 - Fibonacci_sequence class, 221, 224
 - glob_sequence class, 128
 - reasons for not providing in COM enumerator adaptations, 390
 - pid_sequence class, 203
 - provided without corresponding size(), 147–148, 174
 - stack adaptor, 4
 - string_tokeniser class, 332
- encapsulation, 89–92
- end() method, 147–148, 246–247
- end() sentinels gotcha, 309–311
- enforcement mechanisms, 43–44
- enforcement types, 42–43
- enumerating
 - file systems
 - efficiency *versus* usability, 196–200
 - UNIX. *See* **Glob** API; **Opendir/readdir** API.
 - Windows. *See* COM collections; COM enumeration; **FindFirstFile/FindNextFile** API; **WinInet** API.
 - FTP server directories. *See* **WinInet** API.
 - processes and modules. *See* **Process State (PSAPI)** API.
 - registry keys and values
 - basic_key_reg_sequence class, 451–463
 - basic_key_reg_sequence_iterator class, 456–459
 - create_shared_handle() method, 455, 461
 - description, 446–448
 - Façade** classes, 448–451
 - monitored_shared_handle class, 459–461
 - Registry** API, 444–446
 - Registry** library, 448–449
- enumeration context, 372–373
- enumeration_context class
 - advance() method, 378–379
 - class invariant, 379–380
 - construction methods, 375–378
 - definition, 373–374
 - description, 372
 - is_valid() method, 379–380
 - iterator methods, 378–379
 - uses for, 372–373
- enumerator acquisition policies, 403–406
- enumerators, COM. *See* COM enumeration.
- enumerator_sequence class
 - breaking value semantics, 367
 - clear() method, 361–363
 - clear_elements() method, 362–363
 - cloning policy defaults, 385–390
 - cloning_policy_type member type, 360
 - const-incorrect iterator methods, 366–367
 - construction methods, 364–366
 - copy() method, 361–363
 - empty() method, 390
 - enumeration_context class, 362–363
 - init() method, 361–363
 - init_elements() method, 362–363
 - interface_type member type, 360
 - iteration methods, 366
 - member variables, 364
 - pointer member type, 360, 392–393
 - public interface, 359–360
 - quanta parameter, 360, 365
 - reference member type, 360
 - validate_quanta() method, 365–366
 - value policies, 361–363
 - value_policy_adaptor class, 363
 - value_type member type, 360
- EnumProcesses() function, 201–211
- EnumProcessModules() function, 201, 206, 210
- enum_sequence_type member type, 399
- environ variable, 267–268
- eNViromental eXpander (nvx), 266
- environmental mapping
 - abstracting platform-dependent variables, 268–271
 - element-interface coherence, 436
 - environ variable, 267–268
 - environment_variable_traits class, 268–271
 - getenv() function, 267–268
 - heterogeneous reference categories, 297
 - interface planning, 271
 - iteration
 - bidirectional iterator, 279–282
 - class interface, 286–287
 - const_iterator class, 289–290
 - contiguous iterator, 278–279
 - erase() method, 293–294
 - exception safety, 290–292
 - insert() method, 290–292
 - snapshot, 282–286, 288–289
 - snapshot class, 295–296
 - lookup
 - lookup() method, 295–296
 - map interface, 271–277
 - by name, 271–277
 - nvx (eNViromental eXpander), 266
 - operator [] (subscript operator), 295–296
 - nvx (eNViromental eXpander), 266
 - putenv() function, 267–268
 - setenv() function, 267–268
 - size() method, 297
 - subscript by index, 297
 - unsetenv() function, 267–268
 - uses for, 266–267
 - variables
 - by-name access, 267–268
 - deleting, 277–278
 - erasing, 293–294
 - getting, 267–268
 - inserting, 277–278, 290–292
 - linear access, 297
 - lookup, 266
 - putting, 267–268
 - setting, 267–268
 - updating, 277–278
 - wildcards, 266
 - environment_variable_traits class, 268–271
 - equal() method, 89–92, 150, 191, 222, 253–255, 310, 371–372, 489–490, 511
 - erase() method, 259–260, 293–294
 - erasing environmental variables, 293–294
 - error handling
 - enumerating file systems, 118
 - nonexistent window, 307–308
 - read errors, enumerating file systems, 118
 - traits, 103
 - escape characters, enumerating file systems, 118
 - exception handling
 - CArray_adaptor_base, 249–250, 256–257, 261, 263
 - enumerating windows, 301
 - enumerator_sequence, 375, 377–378
 - environment_variable_traits, 270
 - external iterator invalidation, 460–462
 - Fibonacci sequence, 217–218
 - FindFirstFile/FindNextFile** API, 176–178
 - exception safety, enumerating file systems, 121
 - explicit semantic conformance, 62–63
 - expressiveness, C++, xxviii–xxx
 - external iterator invalidation
 - ComboBox controls, 437–444
 - combobox_sequence class, 440–444

- element-interface coherence, 435–437
- enumerating Registry keys and values
 - `basic_key_reg_sequence` class, 451–463
 - `basic_key_reg_sequence_iterator` class, 456–459
 - `create_shared_handle()` function, 461
 - `create_shared_handle()` method, 455
 - description, 446–448
 - Facade** classes, 448–451
 - `monitored_shared_handle` class, 459–461
 - Registry** API, 444–446
 - Registry** library, 448–449
- extrathread action, 437
- intrathread API side effects, 437
- intrathread application side effects, 437
- `ListBox` controls, 437–444
- `listbox_const_iterator` class, 440–444
- `listbox_operation_traits` class template, 440–444
- `listbox_sequence` class, 440–444
- retrieval races, 438–440
- Z-plane, 302, 307–308
- `external_iterator_invalidation` class, 307–308
- externally initialized RAII, 66
- extrathread action, 437
- F**
- fastcall** calling convention, xl, 109–110
- Fibonacci sequence
 - constructor-bound range, 220–223
 - contract programming, 216
 - definition, 212
 - discoverability failure, 218
 - element-interface coherence, 436
 - finite bounds, defining, 218
 - Golden Ratio, 212
 - infinite sequence interface, 215
 - iterators, 219
 - overflow error, 217–218
 - `overflow_error()` method, 217–218
 - Principle of Least Surprise*, 219
 - as an STL sequence, 212–215
 - throwing errors, 217–218
 - true typedefs, 223–225
 - value types, 216–217
- file systems
 - abstraction, 99–102
 - enumerating
 - See* COM collections
 - See* COM enumeration
 - See* **FindFirstFile/FindNextFile** API
 - See* **Glob** API
 - See* **Opendir/readdir** API
 - maximum path length, 103–107
 - files
 - dot, eliding from search, 118, 120
 - filtering, 133–134
 - finding first, 181–184
 - `filesystem_traits` class template, 97–103
 - filtered iteration, 542–554
 - filtering
 - elements, 190–191
 - files and directories, 159
 - `filter_iterator` class, 545–548
 - `findfile_sequence` class, 198
 - `find_first_file()` method
 - FindFirstFile/FindNextFile** API, 181–184
 - WinInet** API, 198
 - FindFirstFile/FindNextFile** API
 - \`\` (backslash), handling, 188–189
 - /`/` (slash), handling, 188–189
 - `basic_findfile_sequence` class
 - attribute methods, 174
 - class interface, 172–175
 - code listing, 166–167
 - constructors/destructor, 173–174
 - implementation methods, 174
 - invariant methods, 174
 - iteration methods, 174
 - member types, 172–173
 - member variables, 175
 - state methods, 174
 - `basic_findfile_sequence_const_iterator` class, 178–180
 - `basic_findfile_sequence_value_type` class, 191–193
 - `begin()` method, 177
 - code listing, 164–166
 - construction, 175, 180–181
 - constructor templates, 194
 - copy assignment operator, 181
 - description, 167–169
 - directories
 - full path, determining, 176–178
 - limiting total size, 171
 - trailing separators, 176–178
 - validating, 176–178
 - drives, 171
 - exception handling, 176–178
 - files, finding first, 181–184
 - filtering elements, 190–191
 - `find_first_file()` method, 181–184
 - infinite recursion, 171
 - invariant method, 177–178
 - iteration class, 178–180
 - iteration methods, 175–176
 - longhand version, 169–170
 - versus* **opendir/readdir** API, 172
 - operator `++` (preincrement operator), 184–191
 - reparse points, 171
 - retrieving entries, 189–190
 - search patterns, 187
 - sequence design, 171–172
 - shared context, creating, 189–190
 - shims, 193–194
 - shorthand version, 170–171
 - `skipReparseDirs` flag, 171
 - uses for, 164–167
 - `validate_directory()` method, 176–178
 - wildcards, 172
 - `find_next_token()` function template, 330
 - fixed array classes, 24
 - fixed element references, 24
 - fixing types. *See* Types, fixing.
 - flags
 - `absolutePath`, 122, 124–125, 130–131, 145, 158–161
 - `bracePatterns`, 122, 124–125
 - `breakOnError`, 122, 124–125
 - directories, 122, 124–126, 131–133, 141, 145–146, 148, 152–153, 159, 162, 167, 170, 173, 182–183, 190
 - `elideIdle`, 209–210
 - `elideSystem`, 209–210
 - `expandTilde`, 122, 124–125, 132
 - `FILE_ATTRIBUTE_DIRECTORY`,
 - `FILE_ATTRIBUTE_REPARSE_POINT`,
 - files, 116, 122, 124–126, 131–135, 145–146, 148, 152–154, 159, 166, 170, 173–174, 182, 190
 - `fullPath`, 141, 145–146, 153, 158–162
 - `GLOB_ALTDIRFUNC`, 118
 - `GLOB_APPEND`, 118–119, 125,
 - `GLOB_BRACE`, 118, 124–125
 - `GLOB_DOOFFS`, 118, 125
 - `GLOB_ERR`, 118, 124
 - `GLOB_LIMIT`, 118–119
 - `GLOB_MARK`, 115, 118–120, 124, 131, 136
 - `GLOB_NOCHECK`, 118
 - `GLOB_NOESCAPE`, 118, 124
 - `GLOB_NOMAGIC`, 118
 - `GLOB_NOSORT`, 118, 124, 131
 - `GLOB_ONLYDIR`, 118, 124, 131–132
 - `GLOB_PERIOD`, 118, 124–125
 - `GLOB_TILDE`, 118, 124–125, 132
 - `GLOB_TILDE_CHECK`, 118

- flags (*cont.*)
 - GW_CHILD, 300–302, 309, 315–316, 323
 - GW_HWNDFIRST, 300, 304, 306–307, 315, 317, 321
 - GW_HWNDLAST, 300, 302, 305, 311, 316
 - GW_HWNDNEXT, 300, 302–303, 307–308, 314, 316, 321
 - GW_HWNDPREV, 300, 302, 305, 314, 316, 321
 - GW_OWNER, 300
 - includeDots, 122, 125–126, 132, 145–146, 148, 152, 173, 182, 190, 200
 - markDirs, 122, 124–126, 131, 133–135, 138
 - matchPeriod, 122, 124–125
 - noEscape, 122, 124–125
 - noSort, 122, 124–125, 131, 134, 137–139
 - REG_NOTIFY_CHANGE_NAME, 447–448, 455, 459
 - skipReparseDirs, 167, 170, 171, 173–174, 183, 190
 - sort, 209–210
 - SYCLBOMF_FIXED_ARRAY, 410–411, 414
- flexibility, C++, xxx–xxxi
- formatting output, 467–468
- forward iteration, 8, 302–304, 545–547
- forward_cloning_policy class, 383–384
- friendship, 89–92
- fullPath flag, 158–161
- function objects, 3, 13. *See also* Algorithms.
- functions
 - cstring_concatenator(), 524–525
 - dynamic invocation, 109–110
 - EnumProcesses(), 201–211
 - EnumProcessModules(), 201, 206, 210
 - environ variable, 267–268
 - getenv(), 267–268
 - GetModuleFileNameEx(), 210–211
 - init_glob() worker, 129–134
 - putenv(), 267–268
 - return types, DRY SPOT principle, 37–38
 - setenv(), 267–268
 - strtok(), 325–327
 - strtok_r(), 330
 - transform_filter(), 552–553
 - unsetenv(), 267–268
- functors. *See* Function objects.
- G**
 - getenv() function, 267–268
 - GetModuleFileNameEx() function, 210–211
 - get_module_path() method, 207
 - get_scatter_slice_ptr shim, 411–414
 - get_scatter_slice_size shim, 411–414
 - get_scatter_slice_size_member_ptr shim, 411–414
 - getting environmental variables, 267–268
 - glob** API. *See also* Glob_sequence class.
 - code decomposition, 119–121
 - code listing, 115–116
 - enumerating file systems
 - { } (braces), allowing in syntax, 118
 - absolute paths, 120, 130–131
 - appending a slash to returned directories, 118
 - appending results, 118
 - copying entries, 132
 - dot files and directories, eliding from search, 118, 120
 - dots directories, eliding, 132–133
 - element access, 128
 - escape characters, 118
 - exception safety, 121
 - filtering files and directories, 133–134
 - flags, 124–126, 131–132
 - flaws, 120–121
 - leaving vacant pointers, 118
 - limiting number of entries, 118
 - no matches found, 118
 - performance, 117
 - read errors, 118
 - search functions, 118
 - searching directories only, 118
 - sorting elements, 134
 - sorting path names, 118
 - tilde expansion, 118
 - trailing slashes, 120
 - wildcard searches, 118
 - error returns, 118
 - flags, 118
 - types and functions, 117
 - uses for, 115–117
 - GLOB_ALTDIRFUNC flag, 118
 - GLOB_APPEND flag, 118
 - GLOB_BRACE flag, 118, 124–126
 - GLOB_DOOFFS flag, 118
 - GLOB_ERR flag, 118, 124–126
 - GLOB_LIMIT flag, 118
 - GLOB_MARK flag, 118, 124–126
 - GLOB_NOCHECK flag, 118
 - GLOB_NOESCAPE flag, 118, 124–126
 - GLOB_NOMAGIC flag, 118
 - GLOB_NOSORT flag, 118, 124–126
 - GLOB_ONLYDIR flag, 118
 - GLOB_PERIOD flag, 118, 124–126
 - glob_sequence class. *See also* **Glob** API.
 - absolute path processing, 130–131
 - code decomposition, 135
 - code listing, 116–117
 - constructors, 126–127, 137–139
 - copying entries, 132
 - declaration of, 122–123
 - description, 121
 - destructors, 126–127
 - dots directories, eliding, 132–133
 - element access, 128
 - element-interface coherence, 436
 - filtering files and directories, 133–134
 - flags, 124–126, 131–132
 - init_glob() worker function, 129–134
 - iteration methods, 128–129
 - member constants, 124–126
 - member types, 123
 - member variables, 123–124
 - public interface, 121–123
 - size methods, 128
 - sorting elements, 134
 - GLOB_TILDE flag, 118, 124–126
 - GLOB_TILDE_CHECK flag, 118
 - Golden Ratio, 212
 - Goose Rule, 61, 64. *See also* Conformance.
- H**
 - Henney, Kevlin, 87–88
 - Henney's Hypothesis*, 87–88, 324, 348
 - heterogeneous iterators, 498–500
 - heterogeneous reference categories, 297
- I**
 - Idle process, 208–210
 - IEnumFileEntry interface, 355–357
 - IEnumXXX** protocol, 356–358
 - IIA (inferred interface adaptation), 80–81, 85–86
 - ILockBytes interface, 410–411
 - immutable collections, 17, 79–80
 - indenting output, 467–468
 - independence *versus* state sharing, 19
 - inferred interface adaptation, 68
 - inferred interface adaptation (IIA), 80–81, 85–86
 - infinite sequence interface, 215
 - inheritance, 240–241
 - init() method, 361–363
 - init_elements() method, 362–363
 - init_glob() worker function, 129–134
 - input iterators, 6–7
 - input_cloning_policy class, 381–383
 - input/output. *See* I/O.
 - input/output stream adaptation, 12

- insert () method, 260–261, 290–292
 - inserting environmental variables, 277–278, 290–292
 - Instance Adaptors
 - class template, 244–245
 - constructors, 236
 - definition, 3, 227
 - input/output stream adaptation, 12
 - stream buffer adaptation, 12
 - templates, 4
 - int type, 71–73
 - integral types, 70–71
 - intent, shims, 48
 - interface-incomplete types, 78–79
 - interface_type member type, 360
 - internally initialized RAII, 65–66
 - intersecting conformance, 64
 - intrathread API side effects, 437
 - intrathread application side effects, 437
 - invalidatable element references, 24–26
 - invalidating iterators. *See* External iterator invalidation.
 - I/O
 - formatting output, 467–468. *See also* `Ostream_iterator` class.
 - gathering. *See* Scatter/Gather I/O.
 - input/output stream iterators, 12
 - IOStreams, 3, 14, 330
 - ISequentialStream interface, 409
 - is_fundamental_type class, 73–74
 - is_integral_type class, 71–73
 - is_same_type class, 74
 - is_signed_type class, 73
 - IStream interface, 409
 - istreambuf_iterator, 12
 - istream_iterator, 12
 - is_valid() method, 379–380
 - iteration
 - bidirectional, 279–282, 304–306, 309–314, 547–548
 - contiguous, 278–279
 - double dereference, 311–313, 321
 - filtered, 542–554
 - filter_iterator class, 545–548
 - forward, 302–304, 545–547
 - iterator category, constraining, 549–550
 - methods, 370–371
 - random access, 548–549
 - reversed semantics, 320–321
 - reversible cloneable, 313–314
 - window_peer_sequence class, 303–304, 306, 321–322
 - iterator adaptors. *See also* Member selector iterator.
 - adapted iterators traits, 533–540
 - creator functions, 486, 503–505
 - defining, 485–487
 - member_selector_iterator class, 509–511
 - transform iterator
 - composite transformations, 496–501
 - description, 481–482
 - DRY SPOT violations, 497
 - heterogeneous iterators, 498–500
 - nontemporary function objects, 497–498
 - transform () algorithm, 483–484
 - typedefs, 497–498
 - uses for, 482–485
 - using, 484–485
 - transform_iterator class
 - arithmetic operators, 490
 - comparison operators, 490
 - construction, 489
 - decrement operators, 489–490
 - defining, 487–489, 491–494, 494–496
 - increment operators, 489–490
 - pointer arithmetic methods, 489–490
 - value type, 486–487
 - iterator category. *See also* Element reference category.
 - adaptable member types, 157–158
 - constraining, 549–550
 - iterator class
 - construction methods, 369–370
 - definition, 368–369
 - equal () method, 371–372
 - iteration methods, 370–371
 - iterator comparison, 371–372
 - make_clone () method, 369–370
 - iterator worker methods, 422
 - iterator_category member, 536
 - iterators. *See also* Algorithms.
 - adapting. *See* Adapted iterators; Iterator adaptors.
 - bidirectional, 8–9
 - categories. *See* Refinements.
 - cloning policies
 - choosing a default, 385–391
 - cloneable_cloning_policy class, 384–385
 - description, 381
 - forward iteration, 381–385
 - forward_cloning_policy class, 383–384
 - input_cloning_policy class, 381–383
 - interface, 381
 - comparison, 371–372
 - compile-time characteristics, 18
 - contiguous, 19–20
 - definition, 3
 - element reference category, 18–19
 - forward, 8
 - input, 6–7
 - input/output stream, 12
 - invalidation. *See* External iterator invalidation.
 - member selection operator, 9–11
 - mutability, 18
 - nonmutating single-pass, 19–20
 - output, 7–8. *See also* `Ostream_iterator` class.
 - predefined adaptors, 11–12
 - random access, 9, 19–20
 - refinements, 5–6, 19–20. *See also specific refinements.*
 - state sharing *versus* independence, 19
 - STL extensions, 18–20
 - traversal, 18
 - undefined behavior, 30–31
- ## K
- Karlsson, Björn, 434
- ## L
- Law of Leaky Abstractions*, 40–41, 279
 - Law of the Big Two*, 147
 - leaky abstractions, 40–41, 64, 100, 125, 136, 149, 255, 277, 279, 308, 385, 401, 406, 448, 478, 480, 544
 - libraries. *See also* C++ libraries.
 - COMSTL**, xxxiv
 - Boost**, 331, xxxvi
 - MFC, adapting to STL. *See* `CArray` container family.
 - Pantheios**, 506–507, xxxvi
 - PlatformSTL**, xxxv
 - recls**, xxxvi
 - Registry**, 448–449
 - STL (Standard Template Library)
 - concept overview, 3–4. *See also specific concepts.*
 - definition, 3
 - extensions, 15–20
 - standard library overlap, 14
 - STLSoft**, xxiv
 - UNIXSTL**, xxxiv
 - WinSTL**, xxxiv
 - list class template, 4
 - ListBox controls, 437–444
 - listbox_const_iterator class, 440–444
 - listbox_operation_traits class template, 440–444
 - listbox_sequence class, 440–444
 - logging facilities, 53–55, 506–507, 519–520. *See also Pantheios.*
 - logical shims, 48
 - long int type, 71–73

looking up variables. *See* Environmental mapping.

lookup() method, 295–296

M

make_clone() method, 369–370

map container, 4

map interface, 271–277

mapping the environment. *See* Environmental mapping.

MAX_PATH constant, 106–107

m_base member, 123–124

m_buffer member, 123–124

member selection operator, 9–11

member selector iterator. *See also* Iterator adaptors; String concatenation.

accumulate() method, 507–509

uses for, 506–509

member selector iterator, creator functions

const members, selecting, 517–518

mutating access

collection with class-type iterators, 517

const collection, 514–516

non-const array, 513

non-const collection, 514

nonmutating access

const array, 512–513

non-const array, 512

member type tagging, 63

memory

allocation, 245–246, 250–252. *See also* Allocators.

contiguity of storage, 4–5

stack, allocating, 93–97

message_queue_sequence class, 415–416, 421–423

method names, resolving, 243–244

methods

accumulate(), 507–509

advance(), 378–379

assign(), 257–258

begin(), 147–148, 177, 246–247

capacity(), 252–253

clear(), 258–259, 361–363

clear_elements(), 362–363

Clone(), 356–357

copy(), 361–363

create_shared_handle(), 455, 461

empty()

enumerator_sequence class, 390

opendir/readdir API, 147

glob_sequence class, 128

end(), 147–148, 246–247

equal(), 89–92, 371–372

erase(), 259–260, 293–294

find_first_file()

FindFirstFile/FindNextFile

API, 181–184

WinInet API, 198

get_module_path(), 207

init(), 361–363

init_elements(), 362–363

insert(), 260–261, 290–292

is_valid(), 379–380

lookup(), 295–296

make_clone(), 369–370

new_enum_method_policy(), 404–405

Next(), 356

overflow_error(), 217–218

pop_back(), 258–259

prepare_directory(), 160–161

push_back(), 256–257

rbegin(), 247–248

rend(), 247–248

reserve(), 252–253

Reset(), 356

size()

CArray_adaptor_base class, 248–250

collection sequence class, 402–403

environment mapping, 297

glob_sequence class, 128

opendir/readdir API, 147

Skip(), 356

swap(), 5, 97, 262–264

validate_directory_(), 176–178

validate_flags(), 125–126

validate_quanta(), 365–366

Meyers, Scott, 89

MFC Array containers. *See* CArray container family.

MFC (Microsoft Foundation Classes) library, adapting to STL. *See* CArray container family.

MFCSTL subproject, xxxv

m_flags member, 123–124

m_glob member, 123–124

Microsoft Foundation Classes (MFC) library, adapting to STL. *See* CArray container family.

m_numItems member, 123–124

modularity, C++, xxxi

modules, enumerating. *See* Process State (PSAPI) API.

monitored_shared_handle class, 459–461

multimap container, 4

multiset container, 4

mutability

collections, 17–18, 79–80

containers, 17–18

iterators, 18

RAII, 65

mutating access

collection with class-type iterators,

517

const collection, 514–516

non-const array, 513

non-const collection, 514

N

named conformance, 57–59, 62–63

_NewEnum member, 403–406

new_enum_by_dispid_policy class, 405–406

new_enum_method_policy() method, 404–405

new_enum_property_policy property, 404

Next() method, 356

nonmember friend function abuse, 89–92. *See also* equal()

nonmutating access, 512–513

nonmutating single-pass iterators, 19–20

notional collections, 15

nvx (eNvironmental eXpander), 266

O

objects

construction/destruction. *See*

Allocators.

generic operations. *See* Function objects.

opendir/readdir API. *See also*

Readdir_sequence class.

code decomposition, 142–144

code listing, 140–141

enumerating directories, 140–142

features, 144–146

versus FindFirstFile/FindNextFile API, 172

snapshots, 161

storing elements, 161–162

types and functions, 142

uses for, 140–142

Open-RJ file reader, xxxvi

Open-RJ Ruby mapping, 428–434

operator -> (member selection operator), 158

operator [] (subscript operator), 243–244, 295–296

operator * (dereference operator), 160

operator ++ (preincrement operator), 156–157, 184–191

operator -> (member selection operator), 31

ostensible return type, 48

ostreambuf_iterator, 12

ostream_iterator class

compatibility, 474

Dereference Proxy pattern, 477–480

design principles, 474–475

formatting output, 467–468

indenting output, 467–468

safe semantics, 472–474

standard C++ library, 469–470

STLSoft library, 470–472

- stream insertion operators, 475–476
- string access shims, 472
- void difference type, 470
- output iterators, 7–8
- overflow error, 217–218
- overflow_error class, 217–218
- P**
- Pantheios** library, 53–55, 506–507, 519–520, xxxvi
- parent classes, DRY SPOT principle, 36–37
- PATH_MAX preprocessor symbol, 105–106, 159–160
- paths
 - absolute, 120, 130–131, 158–161
 - concatenation, 159
 - full, 158–161
 - names, sorting, 118
- performance
 - enumerating file systems, 117
 - gathering scattered I/O, 420–421, 426–427
 - string tokenization, 348–352
- permanent element references, 23
- pid_sequence class, 202–203
- PlatformSTL** subproject, 411–414, xxxv
- pointer member, 537
- pointer member type, 360, 392–393
- pop_back() method, 258–259
- portability, C++, xxxii
- post conditions, 42–43
- preconditions, 42–43
- predefined iterator adaptors, 11–12
- prepare_directory() method, 160–161
- primary shims, 49–52
- Principle of...*
 - Clarity*, xxv
 - Composition*, 211, xxv
 - Diversity*, 195, xxv
 - Economy*, 194–195, xxv
 - Extensibility*, xxv
 - Generation*, xxv
 - Least Surprise*, 385–390
 - Modularity*, 211, xxv
 - Most Surprise*, 194, 302, xxv
 - Optimization*, 211, xxv
 - Parsimony*, xxv
 - Robustness*, xxv
 - Separation*, xxv
 - Simplicity*, 211, xxvi
 - Transparency*, xxvi
- priority_queue class template, 4
- Process State (PSAPI) API**
 - collection characteristics, 202
 - constants, 209–210
 - constructors, 209–210
 - container methods, 202–203
 - EnumProcesses() function, 201–211
 - EnumProcessModules() function, 201, 206, 210
 - exception support, 205–206
 - feature methods, 209–210
 - GetModuleFileNameEx() function, 210–211
 - get_module_path() method, 207
 - Idle process, 208–210
 - module enumeration, 206–208
 - optional API headers, 210–211
 - pid_sequence class, 202–203
 - process enumeration, 207–208
 - process identifiers, acquiring, 204–205
 - process_module_sequence class, 206
 - runtime library invocation, 210–211
 - System pseudo process, 208–210
- processes, enumerating. *See* **Process State (PSAPI) API**.
- process_module_sequence class, 206
- push_back() method, 256–257
- putenv() function, 267–268
- putting environmental variables, 267–268
- Q**
- quanta parameter, 360, 365
- queue class template, 4
- R**
- RAII (Resource Acquisition Is Initialization), 65–66
- random access iteration, 548–549
- random access iterators, 9, 19–20
- ranges, 12–13. *See also* Iterators.
- rbegin() method, 247–248
- readdir API**. *See* **Opendir/readdir API**.
- readdir_sequence class. *See also* **Opendir/readdir API**.
- attribute methods, 148
- constants, 146
- const_iterator class
 - absolutePath flag, 158–161
 - definition, 156
 - fullPath flag, 158–161
 - iterator types, 157–158
 - operator * (dereference operator), 160
 - prepare_directory() method, 160–161
 - version 1, 148–153
 - version 2, 154–156
- construction, 146–147
- copy semantics, 154–156
- description, 144–146
- element-interface coherence, 436
- filtering files and directories, 159
- iteration, 148–153
- member types, 146
- operator -> (member selection operator), 158
- operator ++ (preincrement operator), 156–157
- paths
 - absolute, 158–161
 - concatenation, 159
 - full, 158–161
 - shared_handle class, 155, 157
- recls** library, xxxvi
- refactoring template inheritance, 346–347
- reference member type, 360, 537–538
- referencing elements. *See* Element reference categories.
- referencing *versus* copying, 343–344
- referents, 21–22
- refinements, 5–6, 19–20
- registry, enumerating keys and values
 - basic_key_reg_sequence class, 451–463
 - basic_key_reg_sequence_iterator class, 456–459
 - create_shared_handle() function, 461
 - create_shared_handle() method, 455
 - description, 446–448
 - Façade** classes, 448–451
 - monitored_shared_handle class, 459–461
 - Registry API**, 444–446
 - Registry** library, 448–449
- Registry API**, 444–446
- Registry** library, 448–449
- rend() method, 247–248
- repare points, 171
- reporting contract violations, 43–44
- reserve() method, 252–253
- Reset() method, 356
- resource acquisition, 65–66
- Resource Acquisition Is Initialization (RAII), 65–66
- response to contract violations, 43–44
- retrieval races, 438–440
- return-type variance. *See* ARV (argument-dependent return-type variance).
- reverse iterator type, defining, 11
- reverse_iterator, 11
- reversible cloneable iteration, 313–314
- robustness, C++, xxx
- Ruby, 428–434
- S**
- satisficing / satisfaction, xxxii–xxxiii
- Scatter/Gather I/O
 - ACE (Adaptive Communications Environment)**, 414–420
 - ACE Reactor** framework, 414–420
 - Ace_Message_Block communications

- Scatter/Gather I/O (*cont.*)
- block transfer validity, 421–423
 - description, 420
 - iterator worker methods, 422
 - message_queue_sequence class, 421–423
 - performance, 420–421, 426–427
 - shared_handle worker methods, 423
 - standard library, specializing, 424–426
- APIs, 409–414
- description, 407–408
- get_scatter_slice_ptr** shim, 411–414
- get_scatter_slice_size** shim, 411–414
- get_scatter_slice_size_member_ptr** shim, 411–414
- ILockBytes interface, 410–411
- ISequentialStream interface, 409
- IStream interface, 409
- linearizing with COM streams, 409–411
- message_queue_sequence class, 415–416, 421–423
- scatter_slice_sequence class template, 411–414
- shared_handle class, 418–420
- shims, 411–414
- stream-related abstraction, 409–411
- scatter_slice_sequence class template, 411–414
- scoped_handle class template, 107–109
- search functions, enumerating file systems, 118
- search patterns, **FindFirstFile/FindNextFile** API, 187
- searching. *See also* Enumerating, file systems.
- directories only, 118
 - FTP server directories. *See* **WinInet** API.
- sequence containers, 4
- set containers, 4
- setenv() function, 267–268
- setting environmental variables, 267–268
- SFINAE (Substitution Failure Is Not An Error), 82–83
- shared context, creating, 189–190
- shared_handle class, 155, 157, 418–420
- shared_handle worker methods, 423
- Shavit, Adi, 205, 211
- shims
- access, 52–56
 - attribute, 49
 - category, 48
 - composite, 52–56
 - conformance, 63
 - control, 48
 - conversion, 49–52
 - definition, 48
 - FindFirstFile/FindNextFile** API, 193–194
 - forms of, 48
 - get_scatter_slice_ptr**, 411–414
 - get_scatter_slice_size**, 411–414
 - get_scatter_slice_size_member_ptr**, 411–414
 - intent, 48
 - logical, 48
 - name, 48
 - ostensible return type, 48
 - primary, 49–52
 - Scatter/Gather I/O, 411–414
 - string access, 53–56
 - short int type, 71–73
 - signed char type, 71–73
 - signed integer types, 71–73
 - sign-opposite types, 70–71
 - sign_traits class, 70–71
 - Single Point of Truth (SPOT), 34. *See also* *DRY SPOT* principle
 - single-character token delimiter, 340–341
 - size() method
 - CArray_adaptor_base class, 248–250
 - collection_sequence class, 402–403
 - environment mapping, 297
 - glob_sequence class, 128
 - opendir/readdir** API, 147
 - Skip() method, 356
 - skipReparseDirs flag, 171
 - slash (/), **FindFirstFile/FindNextFile** API, 188–189
 - snapshot, 282–286, 288–289
 - snapshot class, 295–296
 - sorting
 - elements, 134
 - path names, 118
 - Spolsky, Joel, 40
 - SPOT (Single Point of Truth), 34. *See also* *DRY SPOT* principle
 - stack class template, 4
 - stack memory, allocating, 93–97
 - standard containers, 16
 - Standard Template Library (STL). *See* STL (Standard Template Library).
 - standard-conformant containers, 16
 - state sharing *versus* independence, 19
 - static assertions, 46–47
 - stdcall** calling convention, xl, 109–110
 - STL (Standard Template Library)
 - concept overview, 3–4. *See also* *specific concepts*.
 - definition, 3
 - extensions
 - collections, 15–17
 - containers, 15–17
 - iterators, 18–20
 - standard library overlap, 14
 - STL collections. *See* Collections.
 - STL containers. *See* Containers.
 - STLSoft** library, 24, xxiv
 - STLSoft** subprojects, xxxiv–xxxv
 - storage, contiguous, 4–5
 - stream buffer adaptation, 12
 - string access shims, 53–56
 - string concatenation, 519–525.
 - See also* Member selector iterator.
 - string handling, abstraction, 98–99
 - string objects, concatenating, 527–532
 - string tokenization
 - Boost libraries, 331
 - comparing tokens, 338–340
 - const_iterator, 334–336
 - definition, 325
 - description, 325–327
 - design problems, 345–348
 - element-interface coherence, 436
 - examples
 - character set delimiter, 344–345
 - composite string delimiter, 342
 - copying *versus* referencing, 343–344
 - preserving blanks, 342–343
 - single-character delimiter, 340–341
 - views, 343–344
 - finding the next token, 330
 - find_next_token() function template, 330
 - Henney's Hypothesis*, 324, 348
 - IOStreams library, 330
 - performance, 348–352
 - refactoring template inheritance, 346–347
 - selecting categories, 336–337
 - string_tokeniser class template, 331–333
 - string_tokeniser_comparator class template, 338–340
 - string_tokeniser_type_traits class template, 337–338
 - StringTokenizer class template, 327–328
 - strtok() function, 325–327
 - strtok_r() function, 330
 - tokenizer class template, 331
 - tokenizing a string, 325–327, 330
 - typedef templates, 347–348
 - type generator templates, 347–348
 - use cases, 329–330
 - string_tokeniser class template, 331–333, 436

- string_tokeniser_
 - comparator class template, 338–340
 - string_tokeniser_type_
 - traits class template, 337–338
 - StringTokenizer class template, 327–328
 - Stroustrup, Bjarne, 45
 - strtok() function, 325–327
 - strtok_r() function, 330
 - structural conformance, 59–61, 62–63
 - subscript by index, 297
 - Substitution Failure Is Not An Error (SFINAE), 82–83
 - swap() method, 5, 97, 262–264
 - swapping internal states, 5
 - System pseudo process, 208–210
 - system state information, accessing. *See* **Process State (PSAPI) API**.
- T**
- templates. *See also* Traits.
 - Class Adaptors, 4
 - for classes. *See* Class templates.
 - instance adaptors, 4
 - usability, 87–88
 - tilde expansion, enumerating file systems, 118
 - tokenizer class template, 331
 - tokenizing strings. *See* String tokenization.
 - traits. *See also* Templates.
 - adapted iterator, 533–540
 - adapted_iterator_traits class
 - const_pointer member, 538
 - const_reference member, 538
 - description, 533–535
 - difference_type member, 536–537
 - effective_const_pointer member, 539
 - effective_const_reference member, 539
 - effective_pointer member, 539
 - effective reference member, 539
 - iterator_category member, 536
 - pointer member, 537
 - reference member, 537–538
 - using, 540
 - value_type member, 536
 - allocator_selector template, 75
 - base types, 69–70
 - base_type_traits class, 69–70
 - comparing types, 74
 - description, 67–69
 - error handling, 103
 - establishing member types, 97–98
 - inferred interface adaptation, 68
 - int type, 71–73
 - integral types, 70–71
 - is_fundamental_type class, 73–74
 - is_integral_type class, 71–73. *See also* Is_signed_type class.
 - is_same_type class, 74
 - is_signed_type class, 73. *See also* Is_integral_type class.
 - long int type, 71–73
 - return types, 103
 - short int type, 71–73
 - signed char type, 71–73
 - signed integer types, 71–73
 - sign-opposite types, 70–71
 - sign_traits class, 70–71
 - true typedefs, 75
 - type aliases, 75
 - type generators, 75
 - unsigned char type, 71–73
 - unsigned int type, 71–73
 - unsigned integer types, 71–73
 - unsigned long int type, 71–73
 - unsigned short int type, 71–73
- transform() algorithm, 13, 483–484
- transform iterator
 - composite transformations, 496–501
 - description, 481–482
 - DRY SPOT violations, 497
 - filtered iterators, 551–554
 - heterogeneous iterators, 498–500
 - nontemporary function objects, 497–498
 - transform() algorithm, 483–484
 - transform_filter() function, 552–553
 - typedefs, 497–498
 - uses for, 482–485
 - using, 484–485
- transform_filter() function, 552–553
- transforming types and values. *See* Transform iterator.
- transform_iterator class
 - arithmetic operators, 490
 - comparison operators, 490
 - construction, 489
 - decrement operators, 489–490
 - defining, 487–489, 491–494, 494–496
 - increment operators, 489–490
 - pointer arithmetic methods, 489–490
 - transient element references, 26–27
 - transparency, C++, xxvii–xxviii
 - traversal, iterators, 18
 - true typedefs, 75, 223–225
 - typedef templates, 347–348
 - type generator templates, 347–348
 - type system leverage, 45–46
 - types
 - adapting. *See* Adaptors.
 - aliases, 75
 - base, 69–70
 - changing interfaces. *See* Adaptors.
 - comparing, 74
 - conversion. *See* Shims.
 - describing. *See* Templates; Traits.
 - detection, 82–83
 - fixing, 83–85
 - integral, 70–71
 - return-type variance. *See* ARV (argument-dependent return-type variance).
 - selection, 81–82
 - signed integer, 71–73
 - sign-opposite, 70–71
 - true typedefs, 75
 - unsigned integer, 71–73
- U**
- UNIXSTL subproject, xxxiv–xxxv
 - unsetenv() function, 267–268
 - unsigned char type, 71–73
 - unsigned int type, 71–73
 - unsigned integer types, 71–73
 - unsigned long int type, 71–73
 - unsigned short int type, 71–73
 - untemporary references, 32–33
 - updating environmental variables, 277–278
 - usability *versus* efficiency, 196–200
 - USER API**
 - child_window sequence class, 301–302
 - description, 299–302
 - element-interface coherence, 436
 - enumerating top-level windows, 300–302
 - errors
 - double deference, 311–313
 - end() sentinels gotcha, 309–311
 - nonexistent window, 307–308
 - external Z-order change, 306–308
 - external_iterator_invalidation class, 307–308
 - iteration. *See also* Zorder_iterator class.
 - bidirectional, 304–306, 309–314
 - double dereference, 311–313, 321

USER API (*cont.*)

- forward, 302–304
- reversed semantics, 320–321
- reversible cloneable, 313–314
- window_peer_sequence class, 303–304, 306, 321–322
- window_peer_sequence class, 301, 303–304
- Windows API, 300
- windows peer sequences, 321–322
- zorder_iterator class
 - bidirectional iteration, 305–306
 - definition, 302–303
 - double dereference, 311–313, 321
 - reverse iteration, 315
 - traits, 315–317
- zorder_iterator_forward_traits class, 315–316
- zorder_iterator_reverse_traits class, 316–317
- zorder_iterator_tmpl**<> template, 317–320

V

- validate_directory() method, 145, 148, 174–178

- validate_flags() method, 125–126, 147, 174–175
- validate_quanta() method, 365–366
- value_policy_adaptor class, 363
- value_type member type, 360, 536
- variables, environmental. *See* Environmental mapping.
- vector class template, 4, 230–232
- views, 343–344
- void difference type, 470
- void element references, 29, 31

W

- wildcards
 - enumerating file systems, 118
 - environmental mapping, 266
 - FindFirstFile/FindNextFile** API, 172
- window_peer_sequence class, 301, 303–304, 306, 321–322
- windows, API for. *See* **USER** API.
- Windows registry. *See* Registry.
- WinInet** API
 - basic_findfile_sequence class, 197
 - basic_ftpdir_sequence class, 199–200

- code listing, 196
- description, 196
- findfile_sequence class, 198
- find_first_file() method, 198

WinSTL subproject, xxxiv–xxxv
wrapper classes, 65

Z

- zorder_iterator class
 - bidirectional iteration, 305–306
 - definition, 302–303
 - double dereference, 311–313, 321
 - element-interface coherence, 436
 - reverse iteration, 315
 - traits, 315–317
- zorder_iterator_forward_traits class, 315–316
- zorder_iterator_reverse_traits class, 316–317
- zorder_iterator_tmpl<> template, 317–320
- Z-plane, API for. *See* **USER** API.