

Index

- constexpr** variables (cont.)
 - potential pitfalls, 314
 - use cases, 307–314
 - alternative to enumerated compile-time integral constants, 307–310
 - diagnosing undefined behavior at compile time, 312–314
 - nonintegral symbolic numeric constants, 310–311
 - storing **constexpr** data structures, 311–312
- constexpr** keyword, 75n5, 304n1, 316n8
- const**-qualified member functions, 300
- constraining
 - deduced parameters, 970–973
 - multiple arguments, 983–984
- constructors. *See also* copy constructors; move constructors
 - boilerplate code repetition, avoiding, 323–325
 - code duplication, avoiding, 48–50
 - delegating
 - description of, 46–48
 - potential pitfalls, 50–51
 - use cases, 48–50
 - deleted in aggregates, 247
 - explicit, passing multiple arguments, 250–252
 - inheriting
 - annoyances, 549–552
 - description of, 535–539
 - potential pitfalls, 546–549
 - use cases, 539–545
 - restrictions on, 269–276
 - for `std::initializer_list`, inadvertently calling, 242–244
 - as trivial, 437
 - user-declared, 274n7, 1087
 - value initializing arguments, avoiding the most vexing parse, 237–238
- containers
 - initialization, 561–562
 - iterating all elements, 684–685
 - nested, 22
- contextual keywords, 1007. *See also* keywords
 - override**
 - description of, 104–105
 - further reading for, 107
 - potential pitfalls, 106
 - use cases, 105–106
 - potential pitfalls, 1023
- contextually convertible to **bool**, 63–65, 1129
- continuous refactoring, 147
- contract guarantees
 - nofail functions, 1117–1122
 - overly strong, 1112–1116
- contract violations, 485
- contracts, 467n26, 485
 - constexpr** functions as part of, 261–262
 - new operator, 616
 - overly restrictive, 480–482
 - rvalue references, 714
- control constructs
 - emulating, 599–600
 - in lambda expressions, 600–601
- controlling constant expressions, 285
- conventional string literals, 113
- conversion operators
 - explicit
 - description of, 61–63
 - potential pitfalls, 66–67
 - use cases, 63–65
 - as placeholders, 1193–1194
- converting constructors, 61
- cooked UDL operators, 841, 843–845, 870
- cookies, 669–675
- copy assignable, 485–486
- copy assignment, 485, 758
- copy assignment operator
 - deleted functions, 54
 - rvalue references, 714
 - user-provided, 759
 - vertical encoding, 451
- copy constructible, 455
- copy construction, 489–492
- copy constructors
 - declaring special member functions, 34
 - deleted functions, 54
 - hijacking with perfect-forwarding constructor, 395–397
 - literal types and, 281
 - rvalue references, 714
 - RVO and NRVO requirements, 804–805
 - as trivial, 437
 - user-provided, 758–759
 - vertical encoding, 450
- copy elision, 390
- copy initialization, 215–216
 - in aggregate initialization, 221
 - in generic code, 239
 - for nonstatic data members, 318
 - scalar type, 235–236
 - unions, 506
- copy list initialization, 226–228
 - direct list initialization, compared, 231–232
 - in factory functions, 240
 - in generic code, 239
 - in member initializer lists, 249–250