

Index

- for **nonstatic data members**, 318
 - `std::initializer_list`, 555
 - copy operations, 522, 627
 - deleted functions, 53
 - move operations as optimization of, 741–767
 - rvalue references, 715
 - sink arguments, 782–783
 - some equivalent to moves, 788
 - copy semantics, 54, 627, 742, 852
 - copy/direct, 215
 - copy/swap idiom, 636, 1097
 - core constant expression, 960n1
 - core language specification, 482
 - core traits, 482
 - .cpp files, 41n2
 - critical section, 71
 - C-style ellipsis, 952
 - C-style functions, 158
 - curiously recurring template pattern (CRTP), 1042–1054
 - compile-time polymorphism, 1046–1050
 - compile-time visitation, 1050–1054
 - refactoring, 1042–1044
 - synthesizing equality, 1045–1046
 - currying, 597–598
 - cv-qualifiers, 1153–1154, 1157–1158, 1207–1208
 - forwarding references, 379
 - as literal types, 280
 - rvalue references, 724
 - as standard-layout types, 417
 - as trivial types, 425
 - cyclic physical dependency, 374
 - cyclically dependent, 75
- D**
- `d_engaged` flag, 1072
 - dangling references, 566–567, 607–608, 1171, 1212
 - data dependency, 999
 - data dependency chains, 998–999, 1002
 - data members
 - const**
 - difficulty of synthesizing, 993–994
 - `memcpy` usage on, 489–493
 - returning as *rvalues* pessimizes performance, 786–787
 - reordering, 178n10
 - strengthening alignment, 168, 170–171
 - data races, 68–69
 - data structures, **constexpr**, 311–312
 - data tables, compile-time evaluation, 291–295
 - death tests, 656
 - debug build, 468
 - debugging lambda expressions, 611
 - decay (of a type), 815
 - decimal floating-point (DFP), 862
 - declarations, 121, 315, 879
 - friend**
 - curiously recurring template pattern (CRTP) use cases, 1042–1054
 - description of, 1031–1033
 - further reading for, 1042
 - potential pitfalls, 1041
 - use cases, 1033–1041
 - prior to C++11, 815–818
 - user-provided destructors, 1105
 - declarator operators, 889
 - declared interface, 987
 - declared type (of an object), 25
 - declaring
 - deleted functions, 58–59
 - function pointers, 127–128
 - decltype**. *See also* **auto** variables; **decltype(auto)** placeholder; **rvalue references**
 - annoyances, 31
 - description of
 - use with entities, 25
 - use with expressions, 25–26
 - potential pitfalls, 30
 - use cases
 - avoidance of explicit typenames, 26–27
 - creation of auxiliary variable of generic type, 28
 - explicit expression of type-consistency, 27–28, 28n1
 - validation of generic expressions, 28–30
 - decltype(auto)** placeholder
 - annoyances, 1213
 - description of, 1205–1210
 - in new expressions, 1210
 - potential pitfalls, 1212–1213
 - specification, 1206–1208
 - syntactic restrictions, 1208–1209
 - use cases, 1210–1212
 - `declval` function, 31
 - deduced parameters, constraints on, 970–973
 - deduced return types, 593–594, 1146
 - annoyances, 1201–1203
 - description of, 1182–1194
 - for lambda expressions, 1189–1190, 1197–1198
 - potential pitfalls, 1200
 - use cases, 1194–1200
 - compiler-applied rules, 1197
 - complicated return types, 1194–1196
 - delaying return-type deduction, 1199–1120
 - perfect returning wrapped functions, 1198
 - returning lambda expressions, 1197–1198