

## Index

- potential pitfalls, 647–650
    - direct usage, 647–649
    - function bodies, lack of consideration, 649–650
  - use cases, 634–647
    - appending elements to `std::vector`, 634–639
    - enforcing **noexcept** contract using **static\_assert**, 639–640
    - `std::move_if_noexcept`, 640–644
    - `std::vector::push_back(T&&)`, 644–647
  - noexcept** swap, defining, 1097–1099
  - nofail functions, 1116–1123
  - nofail guarantee, 1117, 1122–1123
  - noncaptured variables, mixing with captured, 609
  - noncopyable types, making movable, 788–791
  - nondefining declarations, 729
  - nonintegral symbolic numeric constants, 310–311
  - nonprimitive functionality, 67
  - nonrecursive **constexpr** algorithms, 961–962
  - nonreporting contracts, 1120–1122
  - nonreporting functions, 1119, 1122
  - nonstatic data members**
    - auto** not allowed, 212
    - constexpr** variables, 305
    - initialization, 318, 322–323
  - nonthrowing move operations, 1094–1097
  - non-trivial, 1011
  - non-trivial constructors, union membership and, 1174
  - non-trivial destructors, 1101–1104, 1118, 1136
  - non-trivial special member functions, union type and, 1174–1181
  - non-trivial types, vertical encoding for, 448–452
  - non-trivially destructible, 1102–1109, 1137
  - non-type parameters, 902
  - non-type template parameter packs, 901–903
  - non-type template parameters, 901–903, 903n7
  - nonvirtual functions, hiding, 1026–1027
  - [[noreturn]] attribute, 13. *See also* attribute support
    - description of, 95
    - further reading for, 98
    - potential pitfalls, 97–98
      - inadvertently break working programs, 97
      - misuse on function pointers, 98
    - use cases, 95–97
      - compiler diagnostics, 95–96
      - runtime performance, 96–97
  - normative wording, 808
  - NRVO. *See* named return-value optimization (NRVO)
  - null address, 99–102
  - NULL macro, 100
  - null pointer value, 743
  - null statements, 268
  - null terminated strings, 743
  - null-pointer-literal. *See* **nullptr** keyword
  - nullptr** keyword
    - description of, 99–100
    - further reading for, 103
    - use cases, 100–103
      - overload resolution, 101–102
      - overloading literal null pointer, 102–103
      - type safety, 100–101
  - numeric literals
    - digit separators (‘) in
      - description of, 152–153
      - further reading for, 154
      - loss of precision in floating-point literals, 154–156
      - use cases, 153
    - user-defined, 858–862
- ### O
- .o files
    - extern template**, effect on, 359–365
    - reducing code bloat, 365–369
  - object factories, 929–930
  - object files
    - extern template**, effect on, 359–365
    - reducing code bloat, 365–369
  - object invariants, 539, 742
  - object orientation, 1015
  - object representation
    - POD types, 405
    - reinterpret\_cast** keyword, 510, 515–516
  - object-oriented design, vertical encoding comparison, 440–441
  - object-oriented programming, 1015
  - objects
    - creating, 516n42
    - iterating over fixed number, 565–566
    - moving into closure, 988–989
    - reducing code size, 1101–1111, 1143–1144
    - resource-owning, passing around, 771–775
    - `std::initializer_list<E>` initialization, 559
    - strengthening alignment, 169–170
  - obsolete entities, [[deprecated]] attribute for
    - description of, 147–148
    - potential pitfalls, 150
    - use cases, 148–150
  - ODR-used, 581–582, 590, 988n2, 1081
  - offsetof macro
    - aggressive usage, 520–521
    - navigating compound objects, 456–460
    - POD type usage, 410–412
    - support for, 423–425