Index

lambda expressions
  annoyances, 611–614
    capturing `*this` by copy, 611–612
    debugging, 611
    mixing immediate and deferred-execution code, 612–613
    trailing punctuation, 613–614
  configuring algorithms via, 86–87
  `decltype(auto)` placeholders and, 1206
  deduced return types for, 1189–1190, 1197–1198
  description of, 573–597
  further reading for, 614
  generic lambdas
    annoyances, 981–984
    description of, 968–975
    further reading for, 985
    potential pitfalls, 981
    use cases, 975–981
  local/unnamed types, 83–84
  parts of, 577–578
    closures, 578–581
    lambda body, 595–597
    lambda captures, 581–591
    lambda declarators, 591–595
    lambda introducers, 581–591
  potential pitfalls, 607–611
    dangling references, 607–608
    local variables in unevaluated contexts, 610–611
    mixing captured and noncaptured variables, 609
    overuse, 609
  use cases, 597–607
    emulating local functions, 598–599
    emulating user-defined control constructs, 599–600
    event-driven callbacks, 603–604
    interface adaptation, partial application, currying, 597–598
    recursion, 604–605
    stateless lambdas, 605–607
    with `std::function`, 601–603
    variables and control constructs in expressions, 600–601
lambda introducers, 581–591, 986
lambda-capture expressions. *See also* `auto` variables; braced initialization; forwarding references; lambda expressions; rvalue references
  annoyances, 993–994
    difficulty of synthesizing `const` data members, 993–994
    `std::function` supports only copyable callable objects, 994
    description of, 986–988
    further reading for, 995
    potential pitfalls, 992–993
    use cases, 988–992
      capturing modifiable copy of `const` variable, 990–992
      moving objects into closure, 988–989
      providing mutable state for closure, 989–990
lambda-capture list, 919–921
language undefined behavior, 1115
libraries
  Guidelines Support Library, 17
  Ranges Library, 391–393, 686n4, 687n5
  resilience to code changes, 203
library undefined behaviors, 1115
lifetime extensions, 1162, 1213
  prvalues, 720
  range-based `for` loops, 680, 691–696
  temporary objects, 819–820
limerick in C++ Language Standard, 1081–1082
linear search in variadic templates, 957
`linearInterpolation` function, 16–17
linkage, 83
link-safe ABI versioning, 1067–1071
link-time optimization, 1094, 1143
Liskov, Barbara, 1026, 1030
Liskov Substitution ~~Principal~~ (LSP), 1030
list initialization
  braced initialization and, 215, 233–234
  deducing, 210–211
list initialized literal types, 260
literal types, 278–284
  aggregate types as, 279–280
  array types as, 280
  compile-time constructible, 462–464
  in constant expressions, 260–261, 273, 277–278
  `constexpr` constructors and, 281
  cv-qualifiers as, 280
  identifying, 282–284
  pointers as, 281
  reference types as, 279
  scalar types as, 278
  `std::initializer_list`, 556
  `std::is_literal_type`, 283n14
  trivially destructible types as, 431
  user-defined, 280
  variable templates of, 302
  `void` return type as, 280
literals
  binary
    description of, 142–143
    further reading for, 146
    use cases, 144–146