Glossary

protocol hierarchy – an inheritance hierarchy consisting exclusively of protocols, whereby higher-level protocols serve only to widen and augment the functionality available to public clients; see **lakos96**, Appendix A, "The Protocol Hierarchy Design Pattern," pp. 737–768. `final` (1020)

prvalue – short for *pure rvalue*; an expression of this value category — such as a function that returns *by value* or a numeric literal — has a value but no inherent identity; see also *lvalue* and *xvalue*. `decltype` (25), `nullptr` (99), `auto` Variables (206), `constexpr` Functions (282), `enum class` (346), Generalized PODs '11 (513), Range `for` (692), *Rvalue* References (711), `decltype(auto)` (1206)

publicly accessible – implies, for a given member of a user-defined type, that its access level is *public*. Generalized PODs '11 (489)

pun – the act of type punning, i.e., accessing an object from other than a compatible type.

pure abstract interface – one that is a protocol, i.e., providing no implementation of any kind. Inheriting Ctors (540)

pure function – one that (1) produces no side effects and (2) returns a consistent value for a given set of arguments. Such functions can, for example, have their results *memoized* (reused for specific input arguments without having to reevaluate the function) with no observable change in the behavior of the program. Attribute Syntax (16)

pure virtual function – one that is a **`virtual`** function declared with a *pure* specifier, `=0`. Such a function need be defined only if it is called with (or as if with) its qualified id; see also protocol. `final` (1008)

qNaN – short for quiet NaN.

QoI – short for quality of implementation.

qualified id – an unqualified id following a (possibly empty) sequence of namespace or class specifiers, each separated by `::` — e.g., `B::x`, **`decltype`**`(a)::d_b`, or `std::vector<int>`.

qualified name – one that contains the scope-resolution operator (`::`) — e.g., `::foo` (global scope) or `bar::baz` (`bar` scope). `inline namespace` (1060)

qualifier – (classically) short for cv-qualifier, one of **`const`** or **`volatile`**; see Section 3.1."Ref-Qualifiers" on page 1153. Variadic Templates (889)

quality of implementation (QoI) – a characterization with respect to a range of (e.g., compiler or library) implementation behaviors that are allowed by the Standard along with the possibly subjective measure of how well those implementations meet the intent of the Standard and the needs of the users. `constexpr` Functions (277), Generalized PODs '11 (529)

quiet NaN (qNaN) – a NaN value that is generated by certain computations (e.g., 0.0/0.0) and that, when used as an operand, yields a NaN result; e.g., $1.0 + NaN$ yields NaN. Contrast with signaling NaN. Generalized PODs '11 (531)

RAII – short for resource acquisition is initialization. Deleted Functions (54), Forwarding References (388), `noexcept` Operator (644), *Rvalue* References (769), `noexcept` Specifier (1126)

range – any object (e.g., an `std::vector`) that exposes a sequence of elements suitable for iteration using a range-based **`for`** loop. Note that C++20 elaborates on this notion via the addition of the *ranges* library. `decltype` (29), Variadic Templates (877)

range-based `for` loop – a variety of **`for`** loop, introduced in C++11 (see Section 2.1."Range **`for`**" on page 679), that provides a simpler, more abstract syntax — **`for`** `(` *declaration* `:` *range-expression* `)` — used to iterate through the elements of a range. Range `for` (679)