Glossary

predicate – a Boolean-valued expression, typically indicating whether some particular property holds; see also predicate function. Lambdas (575)

predicate function – one that returns a Boolean value; see also predicate functor. Local Types '11 (86)

predicate functor – a callable object that returns a Boolean value; see also predicate. Lambdas (575)

primary-class-template declaration – a declaration that introduces a class template into the current scope (and, hence, is neither a specialization nor a partial specialization). Variadic Templates (881)

primary declaration – short for primary-class-template declaration. Variadic Templates (881)

private inheritance – derivation (e.g., using the **`private`** keyword) from a base class such that inheritance does not afford programmatic access to the base class to any further derived classes or ~~pubic~~ clients. `final` (1029)

procedural – implies, for a given imperative programming paradigm or language, that the basic building blocks are functions that operate on raw data rather than, say, objects that encapsulate raw data along with relevant functionality; see also object orientation. Generalized PODs '11 (448)

proctor – an object of a proctor class. `noexcept` Operator (646), `noexcept` Specifier (1139)

proctor class – one that, like a scoped guard, uses RAII to take temporary ownership of a resource and ensure that the held resource is released when the flow of control exits scope unexpectedly, e.g., via a thrown exception. Unlike a scoped guard, however, a proctor class necessarily has an explicit *release* operation that allows ownership of the resource to be adopted by another (longer-lived) entity before the proctor is destroyed. Proctor objects are used for writing exception-safe code in an exception-agnostic programming style. `noexcept` Operator (646)

production build – one that employs a compilation mode that prioritizes the performance required of a production system, perhaps sacrificing niceties such as defensive checks or debugging information. Generalized PODs '11 (469)

programmatically accessible – implies, for a given (logical) entity (e.g., within some other entity), that it can be manipulated, accessed, or detected *programmatically* (i.e., using the C++ language) by clients. `noexcept` Specifier (1085)

protocol – a class whose (user-declared) members — apart from an empty destructor, possibly defined out of line in a source (`.cpp`) file — consist of only pure virtual functions and that does not inherit (either directly or indirectly) from any other class that is not itself a protocol. Generalized PODs '11 (440), Inheriting Ctors (540), `final` (1018)

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)