Glossary

golden file – a file containing the expected output of a (regression) test program. The test program is run, creating an output file that is then compared to the golden file, and if the files match, the test passes. Raw String Literals (114)

grouping macro – one that expands all of its arguments using `__VA_ARGS__`; a grouping macro is useful for circumventing syntactic annoyances that occur when a conventional macro is supplied a multiparameter template and thus with macro arguments containing commas that are not themselves nested within parentheses; e.g., `SOME_MACRO(SomeTemplate<A, B, C>)` results in a syntax error. Generalized PODs '11 (520)

guaranteed copy elision – a form of copy elision that became mandatory in C++17: when an object is initialized with a *prvalue* of the same type (e.g., when returning from a function *by value*), no temporary object is created, and the destination object is constructed directly from the initializing expression, thereby eliminating any need for (accessible) copy operations. Braced Init (216), `noexcept` Operator (648), *Rvalue* References (791), Ref-Qualifiers (1163)

handle type – one that defines a (typically lightweight) *proxy* for a physically separate object or resource, often wrapping a lower-level API that interacts directly with a raw resource. *Rvalue* References (792)

hard UB – short for language undefined behavior (a.k.a. language UB). `noexcept` Specifier (1115)

has identity – states, for a given entity, that there is a way (e.g., by name or address) of identifying it (e.g., a *glvalue*) other than just reiterating its value (e.g., a *prvalue*). For example, a variable or data member thereof has identity, whereas a (nonstring) literal does not. *Rvalue* References (711)

header-only library – a library whose full implementation is contained in header files and all defined functions are **`template`** or **`inline`**, removing the need to link library-specific object files. `inline namespace` (1067)

heap memory – a synonym for dynamically allocated memory.

hidden-friend idiom – the design technique of declaring and defining a free function or free operator as a **`friend`** of a type within the scope of a class definition. A function implemented in this way is not visible to ordinary name lookup or even qualified lookup and will be found only through argument-dependent lookup — i.e., only when the type declaring the *hidden friend* is participating in overload resolution. Generalized PODs '11 (472)

hide – preventing access, by one entity, to another entity of the same name due to name lookup rules. For example, function-name hiding occurs when a member function in a derived class has the same name as one in the base class, but it is not overriding it due to a difference in the function signature or because the member function in the base class is not **`virtual`**; the *hidden* member function is accessible only via a pointer or reference to the base class. Another example occurs when a type `S` is hidden by a variable — e.g., **`struct`** `S { } S; S s;` (Error, `S` is not a type.) — i.e., one having the same name in the same scope. Inheriting Ctors (536), Lambda Captures (987)

hierarchical reuse – a central paradigm of effective large-scale software development in which reuse is not limited to client-facing components but instead extends downward recursively to apply to all of the parts comprised by every component; see also Software Capital. `final` (1012)

higher-order function – one that operates on other functions — i.e., takes a function as an argument or returns a function as its return value. Trailing Return (125)