Glossary

plain old data (POD) – a now-deprecated term in the C++ Standard (replaced by the union of standard-layout type and trivial) used to describe C++ types that were C compatible, having the same layout and behavior in both languages.

platonic value – one whose *unique* meaning is understood outside of the current running process. Multiple variables in separate programs, processes, or databases — having different representations (e.g., `5u`, `5.0`, `'V'`, or `"five"`) — might each identify such a value, but that value (i.e., *the* integer 5) is itself unique. *Rvalue* References (742)

PMR – short for polymorphic memory resource.

POD – short for Plain Old Data, such as a C++03 POD type or C++11 POD type. Generalized PODs '11 (401), **`union`** '11 (1174)

point of instantiation – the source-code location where template arguments are supplied (either directly or via template argument deduction) to template parameters to form a template instantiation.

pointer semantics – a *proxy* or handle type with *in-core* (a.k.a. in-process) value semantics that behaves similarly to a built-in pointer in that a value of the type provides access — typically via the dereference operators, `*` or `->` — to some resource (e.g., a separately allocated object, as is the case for `std::shared_ptr`) or else has a *null* value. If an object of pointer-semantic type is copied, the original and the *copy* will refer to the *same* resource; any modifications made to the resource through one will be reflected in accesses through the other. Two pointer-semantic objects will not have the same value unless they refer to the same resource or both are *null*. Note that pointer-semantic objects are more independent of their referenced entity than are reference-semantic objects in that the former can be modified (e.g., assigned) independently, have a separate notion of equality, and be *null*, whereas the latter approximate fixed aliases to their referenced entity, analogous to the difference between built-in pointers and references; in particular, assigning from a pointer-semantic object, unlike a reference-semantic one, does not imply copying its referenced resource.

pointer to member – a type (or a value of that type) that is able to identify (by its value) a specific non**`static`** member of a specific class type, such as an **`int`** data member of **`class`** `X`, or a possibly **`virtual`**, non**`static`** member function having a specific signature and return value. A class member cannot be accessed using the value of a pointer-to-member type alone, but instead it must be combined with the address of a live object of the specified (or derived) type. **`explicit`** Operators (64), Generalized PODs '11 (456)

polymorphic class – one having a **`virtual`** function or a **`virtual`** base class. **`noexcept`** Operator (617)

polymorphic memory resource (PMR) – (1) a class derived from the standard abstract base class `std::pmr::memory_resource`, used to customize memory allocation and deallocation when using classes that obtain memory from an `std::pmr::polymorphic_allocator` object; (2) colloquially, the facilities in the C++ Standard Library within the `std::pmr` namespace, including `memory_resource`, `polymorphic_allocator`, `monotonic_buffer_resource`, and `unsynchronized_pool_resource`. **`alignof`** (190), Default Member Init (328)

polymorphic type – one that, in C++, is implemented as a polymorphic class. **`noexcept`** Operator (616), **`final`** (1011)

positive semidefinite – implies, for a given matrix, that it is both Hermitian and all of its eigenvalues are non-negative; see **vandenbos17**. **`noexcept`** Operator (655)

POSIX epoch – 00:00:00 UTC on January 1, 1970, the reference time point against which POSIX time representations are typically based. **`constexpr`** Functions (291)