Glossary

special member function – one of the six member functions that the compiler knows how to generate: default constructor, copy constructor, move constructor, copy-assignment operator, move-assignment operator, and destructor. Defaulted Functions (33), Deleted Functions (53), `constexpr` Functions (266), Generalized PODs '11 (413), `initializer_list` (553), *Rvalue* References (710), `noexcept` Specifier (1086), `union` '11 (1174)

specialization – the class, function, or class member that results when supplying a template argument for every (nonoptional) template parameter of a template. The use of a specialization will trigger template instantiation of the *primary template* (or one of its partial specializations) unless there exists an explicit specialization for that template. Variadic Templates (884)

stack frame – a contiguous segment of memory within a running program that is used to hold local variables — i.e., objects having automatic storage duration. `noexcept` Specifier (1101)

stack memory – a synonym for automatically allocated memory.

stack unwinding – the process by which the runtime library destroys automatic variables when traversing the ~~call~~ stack from the point an exception is thrown to the point it is caught. Note that, if a thrown exception encounters a function having a **`noexcept`** specification, it is implementation defined whether stack unwinding occurs before the runtime library calls `std::terminate`; see Section 3.1."**`noexcept`** Specifier" on page 1085. `noexcept` Operator (621), `noexcept` Specifier (1135)

standard conversion – an *implicit conversion*, having built-in meaning, from one type to another, such as array to pointer decay or **`short`** to **`int`** promotion. Unlike a user-defined conversion, a standard conversion does not involve a converting constructor or conversion operator. Zero or more *implicit conversions* might be used when an expression is converted to a destination type, such as when direct initializing a variable, or when a function parameter is being initialized from a function argument. The Standard defines a *partial ordering* of conversions that is used, during overload resolution, to choose certain conversions over others, with sequences consisting entirely of standard conversions always preferred over those containing a user-defined conversion. Deleted Functions (56), `enum class` (334), Generalized PODs '11 (509), User-Defined Literals (835)

standard-layout – implies, for a given class type, that its physical layout (a.k.a. footprint in memory) has properties that can be relied upon portably (e.g., that the address of the first data member coincides with the address of the object overall), as well as being suitable for integration with other languages, C in particular; see also standard-layout class and standard-layout type. Generalized PODs '11 (420)

standard-layout class – a **`class`**, **`struct`**, or **`union`** having all nonstatic data members at the same access level, no nonstatic data members of reference type or non-standard-layout type, no base classes that are not standard-layout classes, and no **`virtual`** functions or **`virtual`** base classes; furthermore, if such a class is not a **`union`**, no two subobjects — i.e., base classes, nonstatic data members, and subobjects thereof — of the same type will both have offset 0 within an object of the outermost type. Generalized PODs '11 (422)

standard-layout class type – a synonym for standard-layout class. Generalized PODs '11 (420)

standard-layout type – one that is a (possibly cv-qualified) scalar or standard-layout class type, or an array of such a type. `alignas` (178), `alignof` (186), Generalized PODs '11 (401)

statement – a unit of programming larger than an expression that typically governs the scope of temporaries created therein; unlike expressions, statements cannot be nested. An expression