Glossary

aggregate – an aggregate type or an object thereof. Aggregate Init '14 (138), Braced Init (230), Default Member Init (330), Generalized PODs '11 (402), *Rvalue* References (750), Variadic Templates (877), `noexcept` Specifier (1087)

aggregate class – one of aggregate type. Generalized PODs '11 (415)

aggregate initialization – the initialization of an aggregate from a braced-initializer list. Aggregate Init '14 (138), Braced Init (221), `constexpr` Functions (273), Default Member Init (330), Generalized PODs '11 (463), *Rvalue* References (752)

aggregate type – (1) a class type having no user-provided or explicit constructors, no base classes, no private or protected non**`static`** data members, and no virtual functions, or (2) any array type. As of C++14, aggregates can have default member initializers for non**`static`** data members. As of C++17, public non**`virtual`** base classes are allowed, but inherited constructors are not; as of C++20, all user-declared constructors become disallowed. `constexpr` Functions (279), Generalized PODs '11 (410), *Rvalue* References (742)

algebra – a set of operations, often involving just a single type, that can be applied to object values, along with any rules governing those operations and how they interrelate; see also value semantics. `constexpr` Functions '14 (961)

algorithm selection – the process by which an algorithm is chosen from among a portfolio of potentially applicable algorithms, based on readily observable, especially compile-time, features of the input data set (see **leyton-brown03**). Variadic Templates (947)

alias template – one that defines a family of type aliases parameterized by one or more template parameters. `using` Aliases (135), Variadic Templates (887)

aliasing – having pointers or references to distinct objects (possibly of distinct type) whose footprints overlap in the address space. `noexcept` Operator (638)

alignment (of an address) – the largest integral power of 2 that evenly divides the numerical value of a given address in the address space. `alignas` (168)

alignment requirement (of a type) – the smallest alignment at which an object of a given type can reside in the address space; see also natural alignment. `alignas` (168), `alignof` (184)

allocating object – one that might itself allocate and manage dynamically allocated memory outside of its own footprint using **`new`**, `malloc`, or some other allocation interface, such as `std::allocator` or, as of C++17, `std::pmr::polymorphic_allocator`. `noexcept` Operator (634)

allocator aware – implies, for a given allocating object's type, that its API supports the ability to supply an external resource to the class's constructor, used by the object to obtain memory; see also scoped allocator model.

amortized constant time (of a repeated operation) – a bound on the runtime complexity of a given operation such that when it is repeated N times (where N is a sufficiently large number), the total time spent is proportional to N, leading to a constant *average* time spent per operation. Note that any single iteration might not have a fixed limit on its run time and thus not execute in constant time. The classic example involves populating a default-constructed `std::vector` with allocating objects via repeated calls to `push_back` (assuming dynamic memory allocation itself is slow but still considered a constant-time operation); see also constant time. `noexcept` Operator (636)

API – short for application programming interface. Generalized PODs '11 (402), *Rvalue* References (793), `inline namespace` (1056)