Glossary

debug build – a mode in which code can be compiled that favors the ability to detect and diagnose bugs, often including the enabling of assertions via `<cassert>` or other assertion libraries. Generalized PODs '11 (468)

decay (of a type) – a colloquial term for implicit trivial conversions such as array to pointer, function to pointer, and *lvalue* to *rvalue*. *Rvalue* References (815)

decimal floating-point (DFP) – a representation of floating-point numbers that involves storing the mantissa as a power of 10, as opposed to the classical (binary) floating-point representation in C and C++, which stores the mantissa as a power of 2. User-Defined Literals (862)

declaration – a statement that declares an entity, such as a variable, function, or type; a declaration might be part of a definition; see also nondefining declaration and opaque declaration. `static_assert` (121), `constexpr` Variables (315), Variadic Templates (879), `noexcept` Specifier (1105)

declarator operator – either `*`, `&`, `&&`, `::*`, `[]`, or `()`, any of which can be applied to form a compound type, respectively a pointer, an lvalue reference, an rvalue reference, a pointer to member, an array, or a function. Variadic Templates (889)

declare – to introduce into a scope the existence (and perhaps properties) of a (typically) named entity such as a variable, function, type, template, or type alias (such as a **`typedef`**). Deleted Functions (54), Forwarding References (390), *Rvalue* References (762)

declared type (of an object) – the type with which a (typically) named entity was declared. `decltype` (25)

deduced return type – a type that, for a given function whose return type is represented using the **`auto`** keyword, is deduced from (possibly implicit) **`return`** statements within the function definition (see Section 1.1."Trailing Return" on page 124); note that **`auto`** at the start of a function declaration might instead be used to introduce a trailing, but nonetheless explicit, return type (see Section 1.1."Trailing Return" on page 124). Lambdas (593)

deduction guide – an implicit or user-defined rule that tells the compiler how to deduce template arguments for a class template from a provided initializer — available as of C++17.

default constructed – implies, for a given object, that it has been created without an initializing expression. Generalized PODs '11 (478), *Rvalue* References (752)

default constructor – one that can be invoked without arguments. Generalized PODs '11 (437), *Rvalue* References (754), `noexcept` Specifier (1136)

default initialization – initialization that (1) for a class type, invokes the constructor that would be invoked if the initializer were empty parentheses (i.e., `()`), (2) for an array type, default initializes each of its elements, and (3) in all other cases, performs no initialization. Note that default initialization can occur either because no initializer is supplied or as the result of value initialization. Braced Init (216), `constexpr` Functions (273), *Rvalue* References (765)

default initialized – implies, for a given object, that it was initialized via default initialization. Defaulted Functions (36), Braced Init (221), Default Member Init (322), Generalized PODs '11 (493), *Rvalue* References (752)

default member initializer – one that is associated with the declaration of a nonstatic data member of a class type that may be used during construction when no other initializer is provided (e.g., from a constructor's member initializer list). `constexpr` Functions (270), Default Member Init (318), Generalized PODs '11 (426)

defaulted (function) – implies a defaulted special member function. Generalized PODs '11 (483), `noexcept` Operator (649), *Rvalue* References (757), `noexcept` Specifier (1086)