# Glossary

**reachable** – implies, for a given expression, that its value has identity and that the address of its underlying object representation is accessible. To a good approximation, an object is reachable if it is a *glvalue*, that is, an *lvalue* or an *xvalue* but not a *prvalue*; there are, however, pathological prvalues that are nonetheless reachable. Consider, for example, a type `S` that holds an `int` and stores its `this` pointer as a global variable on construction — `struct S; S* p; struct S { int v; S(int i) : v(i) { p = this; } }; int x = (S(3), 2 * p->v);` — and sets the value of `x` to 6. See Section 2.1."*Rvalue* References" on page 710. *Rvalue* References (712)

**reaching scope** – the set of enclosing scopes containing a lambda, up to and including the innermost enclosing function and its parameters. This scope defines the set of automatic-storage-duration variables that a lambda can *capture* or to which it can refer; see captured by copy and captured by reference. Lambdas (587)

**recursion** – (1) invoking a function that is called (directly or indirectly) from that same function — e.g., `int f(int n) { return n > 0 ? n * f(n-1) : 1; }` — or (2) defining an entity in terms of itself, e.g., a type list (see Section 2.1."Variadic Templates" on page 873). Variadic Templates (875)

**redundant check** – one that is superfluous in a defect-free program (a.k.a. defensive check). `static_assert` (115)

**ref-qualifier** – one of `&` or `&&` applied to a non`static` member function declaration, thereby enabling overloading based on the value category of the object from which that member function is invoked. Forwarding References (380), Ref-Qualifiers (1154)

**reference collapsing** – the C++ language rule for applying `&` or `&&` to a type alias or `decltype` expression that is itself a reference (`T&` or `T&&`), i.e., when there are two *reference* operators being applied to the same underlying type. The resulting reference will be an lvalue reference (`T&`) unless both operators are `&&`, in which case the result will be an rvalue reference (`T&&`). Forwarding References (380)

**reference related** – implies, for a given type `T`, that some other type `U` differs in only cv-qualification (at any level) or `T` is a direct or indirect base class of `U`. *Rvalue* References (726)

**reference type** – one that denotes an alias to an object of the referenced type and can be either an lvalue reference or an rvalue reference. `alignof` (184), `union` '11 (1174)

**reflection** – a language feature allowing a program to inspect and modify its own structure and behavior. The C++ Standards Committee, and specifically its Study Group 7, is actively working toward incorporating static (compile-time) reflection capabilities into a future version of C++. Generalized PODs '11 (520)

**regression test** – one designed to detect the recurrence of some previously corrected undesired property (e.g., a bug) in the software.

**regular type** – one that emulates syntactically the operations (and corresponding behaviors) of built-in types such as an `int`, e.g., that they be copy constructible, copy assignable, destructible, and equality comparable; see **stepanov09**, section 1.5, "Regular Types," pp. 6–8. Note that, due to being copyable, all such types are implicitly also movable. Though required by the definition of regular type, default construction is not typically needed in generic contexts. A type that does not provide equality-comparison operators but would otherwise be considered a regular type is called *semiregular*; see **stepanov15**, section 10.3, "Concepts," pp. 181–184, specifically p. 184. ~~`alignof` (187),~~ *Rvalue* References (751)