Glossary

might need to provide two header files — one containing the opaque enumeration declaration and a second (which may include the first) that provides the full definition; see Section 2.1. "Opaque **enum**s" on page 660. Opaque **enum**s (663)

operator – a kind of function that has a non-function-like syntax known to the compiler and consisting of either a keyword or other token (typically comprising just punctuation characters) that can used as part of an expression alongside its operands — e.g., **`sizeof`**`(a + b)`, where both `+` and **`sizeof`** are operators. Token-based operators include assignment (`=`), equality comparison (`==`), member access, subscripting (`[]`), sequencing (`,`), conditional (`?:`), function call (`()`), etc. Keyword-based operators include **`sizeof`**, **`new`**, **`delete`**, **`typeid`**, and, as of C++11, **`alignof`**, **`decltype`**, and **`noexcept`**. Many of the built-in token-based operators, along with **`new`** and **`delete`**, can be overloaded for class types; notable exceptions include dot (`.`) and conditional (`?:`). **`constexpr`** Functions (265)

ordinary character type – one that is **`char`**, **`signed char`**, or **`unsigned char`**. Note that **`char8_t`** (introduced in C++20) is *not* an ordinary character type. Generalized PODs '11 (501)

out clause – in law, a clause that permits signatories to a contract to opt out of particular provisions or to terminate the contract early. In software contracts, it is a statement in a contract that (1) allows a function not to achieve its stated goal and (2) typically specifies a channel by which it will inform the caller of its failure to do so and perhaps also an explanation of what precipitated that failure. **`noexcept`** Specifier (1117)

out of contract – implies, for a given invocation of a function, that one or more of its preconditions (explicitly stated or otherwise) was not satisfied. *Rvalue* References (744), **`noexcept`** Specifier (1117)

outermost expression – the expression E, for a given expression S, such that S is a subexpression of E and E is not a subexpression of any other expression; see also full expression. *Rvalue* References (820)

over-aligned – ~~implies, for a given type, that its alignment requirement exceeds that of what would otherwise be its minimal required alignment; see also natural alignment.~~ ~~`alignof`~~ ~~(185)~~

overload – (1) a member of a set of functions or operators that have the same name but different signatures or (2) the act of creating such a similarly named function or operator (see also overloading). *Rvalue* References (741)

overload resolution – the process by which, after name lookup, the C++ compiler determines which, if any, function from the set of candidate functions is the *unique* best match for a given argument list. Deleted Functions (53), *Rvalue* References (710), User-Defined Literals (841)

overload set – the set of (viable) candidates (overloads), for a given invocation of a function (or operator), that the compiler refines during overload resolution until it finds the *best viable function*, if one exists, for the supplied argument list.

overloading – the act of creating an overload.

overriding – providing, for a **`virtual`** function declared in a base type, a suitable implementation specific to a derived type. Inheriting ~~Ctors~~ (539)

owned resource – one, such as dynamic memory, a socket, or a shared-memory handle, that is managed by an object (a.k.a. the *owner*), typically with the expectation that the owner will release the resource when it no longer needs it, e.g., in the owner's destructor. Move operations typically transfer an owned resource from one owner to another. On occasion, a resource can have more than one owner — such as in the case of `std::shared_ptr` — in which case the last owner to be destroyed is typically responsible for releasing the resource. *Rvalue* References (741)