Glossary

undefined behavior (UB) – that which results from executing a piece of C++ code for which no behavior is defined by the language or library and, in theory, could be anything. A well-formed construct in the language, e.g., dereferencing a pointer to access an object, can result in language undefined behavior (a.k.a. hard UB) under certain conditions (e.g., if the pointer is null). A function having a narrow contract makes no guarantees when invoked with one or more of its preconditions unsatisfied; doing so results in library undefined behavior (a.k.a. soft UB), which is not in and of itself language UB but might well lead to it; see also ill formed. Attribute Syntax (18), Delegating Ctors (50), Function `static` '11 (70), `long long` (90), `noreturn` (97), `alignof` (190), Braced Init (218), Generalized PODs '11 (401), `initializer_list` (556), Range `for` (692), *Rvalue* References (715), `final` (1024), `friend` '11 (1049), `inline namespace` (1077), `noexcept` Specifier (1104), `union` '11 (1175), `auto` Return (1187)

undefined symbol – the name (possibly mangled) of an entity that has been odr-used but not defined. `extern template` (363)

underlying type (UT) – the integral type used to represent values of an enumeration. ~~Unicode Literals (131),~~ `alignas` (171), `constexpr` Variables (308), `enum class` (333), Generalized PODs '11 (501), Inheriting Ctors (542), Opaque `enum`s (660), Underlying Type '11 (829)

unevaluated context – an unevaluated operand or subexpression thereof. Note that the expression (0 ? a : b) does not provide an unevaluated context for a, even though a will never be evaluated. `decltype` (31), `noexcept` Specifier (1132)

unevaluated operand – an expression used as an argument to a compile-time operator — e.g., `sizeof`, `typeid` (except for expressions of polymorphic type), `alignof`, `decltype`, or `noexcept` — that does not evaluate the expression but merely introspects some aspect of that expression's static type. An unevaluated operand is not ODR-used. As of C++20, either a `requires` expression or an expression used in a `requires` clause is also an unevaluated operand. `noexcept` Operator (615)

Unicode – an international standard encoding used to represent most text characters found in the world's various languages and writing systems, as well as special symbols and control characters. Unicode has a maximum of $1,112,064$ code points, of which $144,697$ were assigned as of Unicode 14.0.0; see **unicode**.

unification – the algorithmic process in C++ of finding the appropriate template arguments that can make two parameterized symbolic types equivalent, e.g., when identifying a type for a template parameter during template argument deduction. Variadic Templates (901)

uniform initialization – a colloquial term for a set of C++11 features, braced initialization in particular, that allow the use of braces ({ and }) as a consistent syntax for initializing an object, whether of scalar type, class type, or array type — e.g., `int` x{0}; S s{}; `char` a[]{'p', 'q'}; see also most vexing parse. Braced Init (215)

unique object address – implies, for a given object of a given type, that no other object of that type resides at that same address at the same time. In general, two non-bit-field objects having overlapping lifetimes must have distinct addresses unless one is nested within the other (e.g., a base class subobject and the enclosing derived class object, or an object and its first non`static` data member) or they are of different types and at least one is an empty base class (e.g., a base class and a non`static` data member with a different type at offset 0). As of C++20, the requirement for an object to have a distinct address may be relaxed under certain circumstances (e.g., for an empty member object of class type) through use of the [[no_unique_address]] attribute. Generalized PODs '11 (418)