Glossary

function scope – used (colloquially within this book) to mean block scope. Note that the C++ Standard as published (through C++20) has an entirely different definition of this term, applying only to labels; that definition was removed from the Standard via a DR (see **herring20**). Function `static` '11 (68)

function template – one that defines a compile-time parameterized function.

function-template-argument type deduction – a process effected by the compiler during overload resolution whereby each function template in the overload set is matched to the specified arguments and, if a valid set of template parameters can be deduced from the argument types, that specialization for the function template remains in the overload set (to be compared with other functions remaining in the set to determine if a unique best match can be found); see also SFINAE. `auto` Variables (195)

function try block – one, including its **`catch`** clause, that serves as a function body, primarily used to provide exception handling in a constructor — e.g., `C::C(X x) : d_x(x) try { /*...*/ } catch (...) { /*...*/ }` — as exceptions thrown from the member initializer list will also be caught and acted upon (but not swallowed) by the exception handler of the function try block. `constexpr` Functions (268)

functor – a callable object. Lambdas (573), *Generic* Lambdas (968)

functor class – a class type that supports callable objects. Lambdas (574)

functor type – the type of a callable object. Lambdas (573)

fundamental alignment – any alignment that is not greater than `std::max_align_t`; note that an alignment is always an integral power of 2 (1, 2, 4, 8, ...). `alignas` (168)

fundamental integral type – a synonym for integral type. `long long` (89)

fundamental type – either an arithmetic type, **`void`**, or `std::nullptr_t`. `noexcept` Operator (622), *Rvalue* References (803), `final` (1014), `union` '11 (1174)

general-purpose machine – a kind of computer that is designed for general applications, used widely in a variety of organizations. `long long` (93)

generalized PODs – a superset of C++03 POD types that are characterized in modern C++ as the intersection of trivial types and standard-layout types; see Section 2.1."Generalized PODs '11" on page 401.

generic – implies, for a given class, function, object, or code fragment, that properties intrinsic to its behavior will be specified (at compile time) at the point where it is used (e.g., the relevant template is instantiated); see also generic programming. Generalized PODs '11 (474)

generic lambda – one having function parameters specified using **`auto`** so as to create a closure having a ~~templated~~ function-call operator. Lambdas (592), *Generic* Lambdas (968)

generic programming – writing and using software in which the types of the objects being manipulated are themselves parameterized (e.g., using C++ templates). `noexcept` Operator (615)

generic type – one that is itself parameterized to accept, at compile time, type (or even non-type) arguments at the point at which it is instantiated; generic types are typically implemented as class templates in C++. Variadic Templates (878)

glvalue – a value category that characterizes an expression as one from which the identity of an object or function can be determined; a value that is not a *glvalue* is necessarily an *rvalue*. *Rvalue* References (717), `decltype(auto)` (1259)