C++ keywords

This is a list of reserved keywords in C++. Since they are used by the language, these keywords are not available for re-definition or overloading.

A – C	D – P	R – Z

`alignas` (since C++11)
`alignof` (since C++11)
`and`
`and_eq`
`asm`
`atomic_cancel` (TM TS)
`atomic_commit` (TM TS)
`atomic_noexcept` (TM TS)
`auto` (1)
`bitand`
`bitor`
`bool`
`break`
`case`
`catch`
`char`
`char8_t` (since C++20)
`char16_t` (since C++11)
`char32_t` (since C++11)
`class` (1)
`compl`
`concept` (since C++20)
`const`
`consteval` (since C++20)
`constexpr` (since C++11)
`constinit` (since C++20)
`const_cast`
`continue`
`co_await` (since C++20)
`co_return` (since C++20)
`co_yield` (since C++20)
	`decltype` (since C++11)
	`default` (1)
	`delete` (1)
	`do`
	`double`
	`dynamic_cast`
	`else`
	`enum`
	`explicit`
	`export` (1) (3)
	`extern` (1)
	`false`
	`float`
	`for`
	`friend`
	`goto`
	`if`
	`inline` (1)
	`int`
	`long`
	`mutable` (1)
	`namespace`
	`new`
	`noexcept` (since C++11)
	`not`
	`not_eq`
	`nullptr` (since C++11)
	`operator`
	`or`
	`or_eq`
	`private`
	`protected`
	`public`
		`reflexpr` (reflection TS)
		`register` (2)
		`reinterpret_cast`
		`requires` (since C++20)
		`return`
		`short`
		`signed`
		`sizeof` (1)
		`static`
		`static_assert` (since C++11)
		`static_cast`
		`struct` (1)
		`switch`
		`synchronized` (TM TS)
		`template`
		`this`
		`thread_local` (since C++11)
		`throw`
		`true`
		`try`
		`typedef`
		`typeid`
		`typename`
		`union`
		`unsigned`
		`using` (1)
		`virtual`
		`void`
		`volatile`
		`wchar_t`
		`while`
		`xor`
		`xor_eq`

- (1) — meaning changed or new meaning added in C++11.
- (2) — meaning changed in C++17.
- (3) — meaning changed in C++20.

Note that `and`, `bitor`, `or`, `xor`, `compl`, `bitand`, `and_eq`, `or_eq`, `xor_eq`, `not`, and `not_eq` (along with the digraphs `<%`, `%>`, `<:`, `:>`, `%:`, and `%:%:`) provide an alternative way to represent standard tokens.

In addition to keywords, there are *identifiers with special meaning*, which may be used as names of objects or functions, but have special meaning in certain contexts.

`final` (C++11) `override` (C++11) `transaction_safe` (TM TS) `transaction_safe_dynamic` (TM TS) `import` (C++20) `module` (C++20)

Also, all identifiers that contain a double underscore `__` in any position and each identifier that begins with an underscore followed by an uppercase letter is always reserved and all identifiers that begin with an underscore are reserved for use as names in the global namespace. See identifiers for more details.

The namespace `std` is used to place names of the standard C++ library. See Extending namespace std for the rules about adding names to it.

The name `posix` is reserved for a future top-level namespace. The behavior is undefined if a program declares or defines anything in that namespace. (since C++11)

The following tokens are recognized by the preprocessor when in context of a preprocessor directive:

`if`
`elif`
`else`
`endif`
	`ifdef`
	`ifndef`
	`define`
	`undef`
		`include`
		`line`
		`error`
		`pragma`
			`defined`
			`__has_include` (since C++17)
			`__has_cpp_attribute` (since C++20)
				`export` (C++20)
				`import` (C++20)
				`module` (C++20)

The following tokens are recognized by the preprocessor *outside* the context of a preprocessor directive:

`_Pragma` (since C++11)