**Common Causes of Undefined Behavior in C++**

**Pointer**

* Dereferencing a NULL pointer
* Dereferencing a pointer returned by a "new" allocation of size zero
* Using pointers to objects whose lifetime has ended (for instance, stack allocated objects or deleted objects)
* Dereferencing a pointer that has not yet been definitely initialized
* Performing pointer arithmetic that yields a result outside the boundaries (either above or below) of an array.
* Dereferencing the pointer at a location beyond the end of an array.
* Converting pointers to objects of incompatible types
* [Using memcpy to copy overlapping buffers](http://stackoverflow.com/a/11211519/241536).

**Buffer overflows**

* Reading or writing to an object or array at an offset that is negative, or beyond the size of that object (stack/heap overflow)

**Integer Overflows**

* Signed integer overflow
* Evaluating an expression that is not mathematically defined
* Left-shifting values by a negative amount (right shifts by negative amounts are implementation defined)
* Shifting values by an amount greater than or equal to the number of bits in the number (e.g. int64\_t i = 1; i <<= 72 is undefined)

**Types, Cast and Const**

* Casting a numeric value into a value that can't be represented by the target type (either directly or via static cast)
* Using an automatic variable before it has been definitely assigned (e.g., int i; i++; cout << i;)
* Using the value of any object of type other than volatile or sig\_atomic\_t at the receipt of a signal
* Attempting to modify a string literal or any other const object during its lifetime
* Concatenating a narrow with a wide string literal during preprocessing

**Function and Template**

* Not returning a value from a value-returning function (directly or by flowing off from a try-block)
* Multiple different definitions for the same entity (class, template, enumeration, inline function, static member function, etc.)
* Infinite recursion in the instantiation of templates
* Calling a function using different parameters or linkage to the parameters and linkage that the function is defined as using.

**OOP**

* Cascading destructions of objects with static storage duration
* The result of assigning to partially overlapping objects
* Recursively re-entering a function during the initialization of its static objects
* Making virtual function calls to pure virtual functions of an object from its constructor or destructor
* Referring to nonstatic members of objects that have not been constructed or have already been destructed

**Source file and Preprocessing**

* A non-empty source file that doesn't end with a newline, or ends with a backslash (prior to C++11)
* A backslash followed by a character that is not part of the specified escape codes in a character or string constant (this is implementation-defined in C++11).
* Exceeding implementation limits (number of nested blocks, number of functions in a program, available stack space ...)
* Preprocessor numeric values that can't be represented by a long int
* Preprocessing directive on the left side of a function-like macro definition
* Dynamically generating the defined token in a #if expression

**To be classified**

* Calling exit during the destruction of a program with static storage duration