# When is a Destructor Called in C++?

* Destruction of an object: When an object of a class goes out of scope or is deleted, the destructor is called to clean up. Example: { MyClass obj; } // Destructor called when obj goes out of scope.
* Deleting a dynamically allocated object: When an object created with 'new' is deleted using 'delete', the destructor is called. Example: MyClass\* obj = new MyClass(); delete obj;
* Returning by value from a function: When a function returns an object by value, destructors are called for the temporary objects involved. Example: MyClass function() { return MyClass(); } // Temporary object is destroyed.
* Throwing an exception: If an exception is thrown and caught, the destructor is called for the object if it goes out of scope in the catch block. Example: try { throw MyClass(); } catch (MyClass obj) {}
* When a container is resized or cleared: Objects removed from a container (like vector, list) have their destructors called. Example: std::vector<MyClass> v(10); v.clear(); // Destructors called for all objects.
* End of program: Destructors for static and global objects are called when the program terminates. Example: static MyClass obj; // Destructor called at program termination.
* Explicit destructor calls: In advanced cases, destructors can be explicitly called using the object's destructor function. Example: obj.~MyClass(); // Not typically recommended outside of placement new scenarios.