C++ static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your C++ code

All rules 578 | Vulnerability 13 | Bug 111 | Security Hotspot 18 | Code Smell 436 | Quick Fix 68

Tags

Search by name...

Members should be initialized in the order they are declared

Analyze your code

Code Smell | Minor | Quick Fix | cppcoreguidelines cert suspicious

Class members are initialized in the order in which they are declared in the class, not the order in which they appear in the class initializer list. To avoid errors caused by order-dependent initialization, the order of members in the initialization list should match the order in which members are declared in a class.

The initialization order, as described here, is:

1. If the constructor is for the most-derived class, virtual bases are initialized in the order in which they appear in depth-first left-to-right traversal of the base class declarations (left-to-right refers to the appearance in base-specifier lists)
2. Then, direct bases are initialized in left-to-right order as they appear in this class's base-specifier list
3. Then, non-static data members are initialized in order of declaration in the class definition.

Noncompliant Code Example

```
#include <iostream>

struct A {
  A(int num) {
    std::cout << "A(num = " << num << ")" << std::endl;
  }
};

struct B {
  int b;
};

class C : public A, B {
public:
  int x;
  int y;

  C(int i) : B{i}, A{b}, y(i), x(y + 1) { }  // Noncompliant
};

int main() {
  C c(1); // Undefined behavior, might print "A(num = 0)"
  std::cout << c.x << " " << c.y << std::endl;  // might prin
}
```

Compliant Solution

```
#include <iostream>

struct A {
  A(int num) {
    std::cout << "A(num = " << num << ")" << std::endl;
  }
};

struct B {
  int b;
};

class C : public A, B {
```

Bug

"std::move" and "std::forward" should not be confused

Bug

A call to "wait()" on a "std::condition_variable" should have a condition

Bug

A pointer to a virtual base class shall only be cast to a pointer to a derived class by means of dynamic_cast

Bug

Functions with "noreturn" attribute should not return

Bug

RAII objects should not be temporary

Bug

"memcmp" should only be called with pointers to trivially copyable types with no padding

Bug

"memcpy", "memmove", and "memset" should only be called with pointers to trivially copyable types

Bug

"std::auto_ptr" should not be used

Bug

Destructors should be "noexcept"

Bug

```
public:
  int x;
  int y;

  C(int i) : A{i}, B{i}, x(i + 1), y(i) { }
};

int main() {
  C c(1); // prints "A(num = 1)"
  std::cout << c.x << " " << c.y << std::endl;  // prints "2
}
```

See

- CERT, OOP53-CPP. - Write constructor member initializers in the canonical order
- C++ Core Guidelines C.47 - Define and initialize member variables in the order of member declaration

Available In:

sonarlint sonarcloud sonarqube Developer Edition