C++ static code analysis: Unevaluated operands should not have side effects

2 minutes

Operands of `sizeof`, `noexcept` and `decltype` are unevaluated. So side effects in these operands (which are all the effects an expression can have in addition to producing a value), will not be applied. And that may be surprising to the reader.

Additionally, the operand of `typeid` may or may not be evaluated, depending on its type: it will be evaluated if it is a function call that returns reference to a polymorphic type, but it will not be evaluated in all the other cases. This difference of behavior is tricky to apprehend and that is why both cases are reported here.

This rules reports an issue when operands of such operators have side-effects.

Noncompliant Code Example

```
class A {
public:
  virtual ~A();
  /*...*/
};
class B : public A { /* ....*/ };
A& create(string const &xml);

void test(string const &xml) {
  int i = 0;
  cout << noexcept(++i); // Noncompliant, "i" is not incremented
  cout << typeid(++i).name(); // Noncompliant, "i" is not incremented
  auto p1 = malloc(sizeof(i = 5)); // Noncompliant, "i" is not changed

  cout << typeid(create(xml)).name(); // Noncompliant, even if the
side-effects will be evaluated in this case
}
```

Compliant Solution

```
class A {
public:
  virtual ~A();
  /*...*/
};
class B : public A { /* ....*/ };
A& create(string const &xml);

void test(string const &xml) {
  int i = 0;
  ++i;
  cout << noexcept(i); // Compliant
  ++i;
  cout << typeid(i).name(); // Compliant
  i = 5;
  auto p1 = malloc(sizeof(i)); // Compliant

  auto a = create(xml);
  cout << typeid(a).name(); // Compliant
}
```