rules.sonarsource.com

C++ static code analysis: Return type of functions shouldn't be const qualified value

2 minutes

There is no reason to add a const qualifier to the return type of a function that returns by **value**.

At best, it will be superfluous. At worst, it will be a performance bug: it can prevent move operation (when copy elision doesn't take place). When an object is const-qualified the copy constructor/assignment will be a better match than the move constructor/assignment.

One might think about adding this qualifier in order to forbid the call of unintended functions on the returned object. A common example is to avoid unintended assignments:

```
X x1, x2, x3;
if (x1 + x2 = x3) { // Compiler will complain since const object cannot be assigned. Should be "x1 + x2 == x3"
...
}
```

C++11 introduced reference qualifiers for member functions. This feature provides a better approach to forbid calling unintended functions:

```
struct X {
X& operator=(const X& other) &;
};
...
X x1, x2, x3;
if (x1 + x2 = x3) { // Compiler will complain since assignment cannot be called on r-value.  Should be "x1 + x2 == x3"
...
}
```

Noncompliant Code Example

```
class A {...};
const A f();  // Noncompliant
```

Compliant Solution

```
class A {...};
A f();
```

See

- C++ Core Guidelines F.20 - Flag returning a const value.