rules.sonarsource.com

C++ static code analysis: Inheriting constructors should be used

2 minutes

When inheriting from a class, if you need to inherit its constructors without additional initialization you should prefer `using-declaration` to inherit all base class's constructors instead of writing them by hand.

`using-declaration` for inheriting constructor is a C++11 feature that makes all constructors of the base visible to the overload resolution when initializing the derived class.

If you need to change the accessibility of one of the inherited constructors, you can do it by keeping the `using-declaration` and declaring that constructor explicitly as private:

```
class Base {
public:
  Base(int p) {}
  Base(int p1, int p2) {}
};

class Derived : public Base {
  using Base::Base;
  Derived(int p1, int p2);  // Changes constructor to private
accessibility
};

int f(){
  Derived b(1); // Base(int p) is visible when initializing the derived
class
  Derived b1(1,2); // Compilation error: Base(int p1, int p2) is not
visible when initializing the derived class
}
```

This rule raises an issue when a constructor inherits the base class constructor without requiring any additional initialization.

Noncompliant Code Example

```
class Base {
public:
  Base(int p) {}
};

class Derived : public Base {
public:
  Derived(int p) : Base(p) {} // Noncompliant
};
```

Compliant Solution

```
class Base {
public:
  Base(int p) {}
};

class Derived : public Base {
  using Base::Base;
};
```

See

- C++ Core Guidelines C.52 - Use inheriting constructors to import constructors into a derived class that does not need further explicit initialization