rules.sonarsource.com

C++ static code analysis: Redundant class template arguments should not be used

2 minutes

Since C++17, class template arguments can be automatically deduced by the compiler, either by looking at the arguments of the class constructors or by using an explicitly defined deduction guide.

Using the class template argument deduction allow to:

- Avoid verbose specification of all template parameter types for a class template.
- Avoid writing helper function that only serves the purpose of deducing the type of a class from its arguments. For example, `std::make_pair`.
- Be able to instantiate a class template with hard to spell or unutterable names, such as the closure type of a lambda.

This rule raises an issue when explicit class template arguments that can be automatically deduced is specified.

Noncompliant Code Example

```
void f() {
   std::vector<int> v1 {1, 2, 3}; // Noncompliant, int could have been deduced
}{code}
```

Compliant Solution

```
using namespace std::literals;
void f() {
   std::vector v1 {1, 2, 3}; // Compliant, int could be deduced
   std::vector<std::string> v2 {"a", "b", "c"}; // Compliant, automatic deduction would create a vector<char const *>
   std::vector v3 {"a"s, "b"s, "c"s}; // Still compliant, another option
}
```