rules.sonarsource.com

C++ static code analysis: The right template argument should be specified for std::forward

2 minutes

`std::forward` forwards lvalues either as lvalues or as rvalues based on its template argument.

`std::forward` should always take as a non-template argument a forwarding reference which is defined by the standard as:

rvalue reference to a cv-unqualified template parameter that does not represent a template parameter of a class template.

If you don’t pass forwarding reference as an argument to `std::forward` {rule:cpp:S5417} will be triggered.

If you don’t pass the template parameter referred to by the forwarded reference or the `decltype` of the forwarded expression this rule will be triggered.

Noncompliant Code Example

```
template <class T>
void g(T&& t);

template <class T>
void f(T&& t) {
 g(std::forward<T&&>(t)); // Noncompliant
 g(std::forward<T&>(t)); // Noncompliant
}
```

Compliant Solution

```
template <class T>
void g(T&& t);

template <class T>
void f(T&& t) {
  g(std::forward<T>(t)); // Compliant
}

struct StrWrapper {
  std::string s = "rand";
  std::string getStr() && {
    return s;
  }
  std::string& getStr() & {
    return s;
  }
};
```

```
template <class T>
void fstr(T&& str);

template <class T>
void wrapper(T&& strWrapper ) {
  fstr(forward<decltype(forward<T>(strWrapper).getStr())>
(forward<T>(strWrapper).getStr())); // Compliant
}
```