rules.sonarsource.com

C++ static code analysis: Empty throws ("throw;") should only be used in the compound statements of catch handlers

2 minutes

An empty `throw` re-throws the temporary object that represents the exception being processed. It is intended to allow partially handling an exception at some level in the call stack (for instance, logging an issue) but then forwarding it to a higher level where it will be fully handled.

However, syntactically, there is nothing to prevent `throw;` being used outside a catch handler, where there might be no exception object to re-throw. In such a case, the program would call `std::terminate`, which is probably not the expected behavior.

Noncompliant Code Example

```
void f1(void)
{
  throw;   // Noncompliant - will call std::terminate() if f1 is
called while no exception is active
}

void g1(void)
```

```
{
  try
  {
    f1();
    throw; // Noncompliant
  }
  catch (...)
  {
    // ...
  }
}
```

Compliant Solution

```
void f1(void)
{
  try
  {
    throw(42);
  }
  catch (int32_t i) // int will be handled first here
  {
    if (i > 0)
    {
      throw; // and then re-thrown - Compliant
    }
  }
}

void g1(void)
```

```
{
  try
  {
    f1();
  }
  catch (int32_t i)
  {
    // Handle re-throw from f1()
    // after f1's handler has done what it needs
    throw;
  }
}
```

See

- MISRA C++:2008, 15-1-3 - An empty throw (`throw;`) shall only be used in the compound-statement of a catch handler.