rules.sonarsource.com

C++ static code analysis: An exception object should not have pointer type

2 minutes

If a pointer to an object is used as an exception, the code that will catch the exception may or may not have to delete the pointed-to object. This is even more complex in the exception case than in classical manual memory management, because of the distance between the `throw` statements and the matching `catch`.

Throwing by value is just simpler and less error prone.

Noncompliant Code Example

class E { /* Implementation */};

```
E globalException;

void fn ( int i )
{
  // In this situation, the developer writing the
'catch' has no way to know if the object
pointed-to by
  // the exception should be deleted or not...
  if ( i > 10 ) {
    throw ( &globalException); // Noncompliant,
the catch is supposed not to delete the pointer
  }
  else {
    throw (new E ); // Noncompliant, the catch is
supposed to delete the pointer
  }
}
```

Compliant Solution

```
class E { /* Implementation */};
E globalException;

void fn ( int i )
```

```
{
  if ( i > 10 ) {
    throw ( globalException); // Throws a copy
of the global variable
  }
  else {
    throw (E{} ); // Throws a new object
  }
}
```

See

- MISRA C++ 2008, 15-0-2 - An exception object should not have pointer type.