

"pthread\_mutex\_t" should be properly

"pthread\_mutex\_t" should not be consecutively locked or unlocked

initialized and destroyed

📆 Bug





"std::move" and "std::forward" should not be confused



A call to "wait()" on a "std::condition\_variable" should have a condition



A pointer to a virtual base class shall only be cast to a pointer to a derived class by means of dynamic\_cast



Functions with "noreturn" attribute should not return

📆 Bug

RAII objects should not be temporary



"memcmp" should only be called with pointers to trivially copyable types with no padding



"memcpy", "memmove", and "memset" should only be called with pointers to trivially copyable types

📆 Bug

"std::auto\_ptr" should not be used

📆 Bug

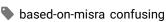
Destructors should be "noexcept"

📆 Bug

### Loops should not have more than one "break" or "goto" statement

# Analyze your code





Restricting the number of exits from a loop is done in the interests of good structured programming. One break or goto statement is acceptable in a loop since this allows, for example, for dual-outcome loops or optimal coding.

#### **Noncompliant Code Example**

With the default threshold of 1:

```
for (int i = 0; i < 10; i++) {
  if (...) {
   break;
               // Compliant
  else if (...) {
               // Non-compliant - second jump from loop
    break;
  else {
  }
}
while (...) {
  if (...) {
    break;
                // Compliant
  if (...) {
    break;
                // Non-compliant - second jump from loop
  }
}
```

#### **Compliant Solution**

```
for (int i = 0; i < 10; i++) {
  if (...) {
               // Compliant
    break;
  }
while (...) {
  if (...) {
    break;
             // Compliant
}
```

## See

- MISRA C:2004, 14.6 For any iteration statement there shall be at most one break statement used for loop termination.
- MISRA C++:2008, 6-6-4 For any iteration statement there shall be no more than one break or goto statement used for loop termination.
- MISRA C:2012, 15.4 There should be no more than one break or goto statement used to terminate any iteration statement

Available In:



© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved. **Privacy Policy**