

# Empty statements should be removed

Code Smell  
Minor

- [based-on-misra](#)
- [cert](#)
- [unused](#)

Empty statements, i.e. `;`, are usually introduced by mistake, for example because:

- It was meant to be replaced by an actual statement, but this was forgotten.
- There was a typo which lead the semicolon to be doubled, i.e. `;;`.

## Noncompliant Code Example

```
void doSomething() {  
    ; // Noncompliant -  
    was used as a kind of TODO marker  
}
```

## Compliant Solution

```
void doSomething() {  
}
```

## Exceptions

In the case of empty expanded macro and in the case of 2 consecutive semi-colons when one of the two is part of a macro-definition then the issue is not raised.

Example:

```
#define A(x) x;  
#define LOG(x)  
  
void fun() {  
    A(5);  
    LOG(X);  
}
```

## See

- MISRA C:2004, 14.3 - Before preprocessing, a null statement shall only occur on a line by itself; it may be followed by a comment provided that the first character following the null statement is a white-space character.

- MISRA C++:2008, 6-2-3 - Before preprocessing, a null statement shall only occur on a line by itself; it may be followed by a comment, provided that the first character following the null statement is a white-space character.
- [CERT, MSC12-C.](#) - Detect and remove code that has no effect or is never executed
- [CERT, MSC51-J.](#) - Do not place a semicolon immediately following an if, for, or while condition
- [CERT, EXP15-C.](#) - Do not place a semicolon on the same line as an if, for, or while statement

© 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.