C static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your C code

"switch" statements should have at least 3 "case" clauses

Analyze your code

Code Smell Minor based-on-misra bad-practice

`switch` statements are useful when there are many different cases depending on the value of the same expression. For just one or two cases however, the code will be more readable with `if` statements.

Moreover, `if` statements are obviously more suitable when the condition of the `switch` is boolean.

Here are the rules to count the cases:

- `default` is counted as a case.
- If there is no `default` clause, the `case` count is incremented by one (to account for the `else` branch of an equivalent `if`).
- All the cases falling through to `default` are not counted (they would all be the `else` branch of the equivalent `if`).

Noncompliant Code Example

```
switch (variable) {
  case 0:
    doSomething();
    break;
  default:
    doSomethingElse();
    break;
}
```

Compliant Solution

```
if (variable == 0) {
  doSomething();
} else {
  doSomethingElse();
}
```

See

- MISRA C:2012, 16.6 - Every switch statement shall have at least two switch-clauses

Available In:

sonarlint sonarcloud sonarqube Developer Edition

Stack allocated memory and non-owned memory should not be freed

Bug

Closed resources should not be accessed

Bug

Dynamically allocated memory should be released

Bug

Freed memory should not be used