DCL41-C. Do not declare variables inside a switch statement before the first case label

According to the C Standard, 6.8.4.2, paragraph 4 [ISO/IEC 9899:2011],

A switch statement causes control to jump to, into, or past the statement that is the switch body, depending on the value of a controlling expression, and on the presence of a default label and the values of any case labels on or in the switch body.

If a programmer declares variables, initializes them before the first case statement, and then tries to use them inside any of the case statements, those variables will have scope inside the switch block but will not be initialized and will consequently contain indeterminate values.

Noncompliant Code Example

This noncompliant code example declares variables and contains executable statements before the first case label within the switch statement:

```c
#include <stdio.h>
extern void f(int i);

void func(int expr) {
    switch (expr) {
        int i = 4;
        f(i);
        case 0:
            i = 17;
            /* Falls through into default code */
        default:
            printf("%d\n", i);
    }
}
```

Implementation Details

When the preceding example is executed on GCC 4.8.1, the variable `i` is instantiated with automatic storage duration within the block, but it is not initialized. Consequently, if the controlling expression `expr` has a nonzero value, the call to `printf()` will access an indeterminate value of `i`. Similarly, the call to `f()` is not executed.

<table>
<thead>
<tr>
<th>Value of expr</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>Nonzero</td>
<td>Indeterminate</td>
</tr>
</tbody>
</table>

Compliant Solution

In this compliant solution, the statements before the first case label occur before the switch statement:
```c
#include <stdio.h>
extern void f(int i);

int func(int expr) {
    /*
     * Move the code outside the switch block; now the statements
     * will get executed.
     */
    int i = 4;
    f(i);

    switch (expr) {
    case 0:
        i = 17;
        /* Falls through into default code */
    default:
        printf("%d\n", i);
    }
    return 0;
}
```

**Risk Assessment**

Using test conditions or initializing variables before the first case statement in a `switch` block can result in unexpected behavior and undefined behavior.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Severity</th>
<th>Likelihood</th>
<th>Remediation Cost</th>
<th>Priority</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCL41-C</td>
<td>Medium</td>
<td>Unlikely</td>
<td>Medium</td>
<td>P4</td>
<td>L3</td>
</tr>
</tbody>
</table>

**Automated Detection**

<table>
<thead>
<tr>
<th>Tool</th>
<th>Version</th>
<th>Checker</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Astrée</td>
<td>19.04</td>
<td>switch-skipped-code</td>
<td>Fully checked</td>
</tr>
<tr>
<td>Axivion Bauhaus Suite</td>
<td>6.9.0</td>
<td>CertC-DCL41</td>
<td>Fully implemented</td>
</tr>
<tr>
<td>Clang</td>
<td>3.9</td>
<td>-Wsometimes-uninitialized</td>
<td></td>
</tr>
<tr>
<td>Coverity</td>
<td>2017.07</td>
<td>MISRA C 2004 Rule 15.0</td>
<td>Implemented</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MISRA C 2012 Rule 16.1</td>
<td></td>
</tr>
<tr>
<td>LDRA tool suite</td>
<td>9.7.1</td>
<td>385 S</td>
<td>Fully implemented</td>
</tr>
<tr>
<td>Parasoft C/C++test</td>
<td>10.4.2</td>
<td>CERT_C-DCL41-a</td>
<td>A switch statement shall only contain switch labels and switch clauses, and no other code</td>
</tr>
<tr>
<td>Polyspace Bug Finder</td>
<td>R2019b</td>
<td>CERT C: Rule DCL41-C</td>
<td>Checks for ill-formed switch statements (rule fully covered)</td>
</tr>
<tr>
<td>PRQA QA-C</td>
<td>9.7</td>
<td>3234 2008 2882</td>
<td>Partially implemented</td>
</tr>
<tr>
<td>PVS-Studio</td>
<td>7.07</td>
<td>V622</td>
<td></td>
</tr>
<tr>
<td>RuleChecker</td>
<td>19.04</td>
<td>switch-skipped-code</td>
<td>Fully checked</td>
</tr>
<tr>
<td>TrustInSoft Analyzer</td>
<td>1.38</td>
<td>initialisation</td>
<td>Exhaustively detects undefined behavior (see the compliant and the non-compliant example).</td>
</tr>
</tbody>
</table>

**Related Vulnerabilities**

Search for vulnerabilities resulting from the violation of this rule on the CERT website.

**Related Guidelines**

Key here (explains table format and definitions)
### Bibliography