DCL04-C. Do not declare more than one variable per declaration

Every declaration should be for a single variable, on its own line, with an explanatory comment about the role of the variable. Declaring multiple variables in a single declaration can cause confusion regarding the types of the variables and their initial values. If more than one variable is declared in a declaration, care must be taken that the type and initialized value of the variable are handled correctly.

Noncompliant Code Example

In this noncompliant code example, a programmer or code reviewer might mistakenly believe that the two variables \( \text{src} \) and \( c \) are declared as \( \text{char *} \). In fact, \( \text{src} \) has a type of \( \text{char *} \), whereas \( c \) has a type of \( \text{char} \).

```
char *src = 0, c = 0;
```

Compliant Solution

In this compliant solution, each variable is declared on a separate line:

```
char *src;    /* Source string */
char c;       /* Character being tested */
```

Although this change has no effect on compilation, the programmer's intent is clearer.

Noncompliant Code Example

In this noncompliant code example, a programmer or code reviewer might mistakenly believe that both \( i \) and \( j \) have been initialized to 1. In fact, only \( j \) has been initialized, and \( i \) remains uninitialized.

```
int i, j = 1;
```

Compliant Solution

In this compliant solution, it is readily apparent that both \( i \) and \( j \) have been initialized to 1:

```
int i = 1;
int j = 1;
```

Exceptions

DCL04-C-EX1: Multiple loop control variables can be declared in the same for statement, as shown in the following function:

```
#include <limits.h> /* For CHAR_BIT */
#include <stddef.h> /* For size_t */
extern size_t popcount(uintmax_t);
#define PRECISION(umax_value) popcount(umax_value)

size_t bitcount(size_t n) {
    const size_t limit = PRECISION(SIZE_MAX);
    size_t count = 0;
    for (size_t i = 0, j = 1; i < limit; ++i, j <<= 1) {
        if (n & j)
            ++count;
    }
    return count;
}
```
The \texttt{PRECISION()} macro provides the correct precision for any integer type and is defined in \texttt{INT35-C}. Use correct integer precisions—see that rule for more information.

**DCL04-C-EX2:** Multiple, simple variable declarations can be declared on the same line given that there are no initializations. A simple variable declaration is one that is not a pointer or array.

```c
int i, j, k;
```

**Risk Assessment**

Declaring no more than one variable per declaration can make code easier to read and eliminate confusion.

<table>
<thead>
<tr>
<th>Recommendation</th>
<th>Severity</th>
<th>Likelihood</th>
<th>Remediation Cost</th>
<th>Priority</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCL04-C</td>
<td>Low</td>
<td>Unlikely</td>
<td>Low</td>
<td>P3</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>Axivion Bauhaus Suite</td>
<td>6.9.0</td>
<td>CertC-DCL04</td>
<td></td>
</tr>
<tr>
<td>CodeSonar</td>
<td>5.2p0</td>
<td>LANG.STRUCT.DECL.ML</td>
<td>Multiple Declarations on Line</td>
</tr>
<tr>
<td>ECLAIR</td>
<td>1.2</td>
<td>CC2.DCL04</td>
<td>Fully implemented</td>
</tr>
<tr>
<td>LDRA tool suite</td>
<td>9.7.1</td>
<td>579 S</td>
<td>Fully implemented</td>
</tr>
<tr>
<td>Parasoft C/C++test</td>
<td>10.4.2</td>
<td>CERT_C-DCL04-a</td>
<td>Each variable should be declared in a separate declaration statement</td>
</tr>
<tr>
<td>SonarQube C/C++ Plugin</td>
<td>3.11</td>
<td>SingleDeclarationPerStatement</td>
<td></td>
</tr>
</tbody>
</table>

**Related Vulnerabilities**

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

**Related Guidelines**

- SEI CERT C++ Coding Standard: VOID DCL04-CPP. Do not declare more than one variable per declaration