EXP13-C. Treat relational and equality operators as if they were nonassociative

The relational and equality operators are left-associative in C. Consequently, C, unlike many other languages, allows chaining of relational and equality operators. Subclause 6.5.8, footnote 107, of the C Standard [ISO/IEC 9899:2011], says:

> *The expression a<b<c is not interpreted as in ordinary mathematics. As the syntax indicates, it means (a<b)<c; in other words, "if a is less than b, compare 1 to c; otherwise, compare 0 to c."*

These operators are *left-associative*, which means the leftmost comparison is performed first, and the result is compared with the rightmost comparison. This syntax allows a programmer to write an expression (particularly an expression used as a condition) that can be easily misinterpreted.

Noncompliant Code Example

Although this noncompliant code example compiles correctly, it is unlikely that it means what the author of the code intended:

```
int a = 2;
int b = 2;
int c = 2;
/* ... */
if (a < b < c) /* Misleading; likely bug */
/* ... */
if (a == b == c) /* Misleading; likely bug */
```

The expression `a < b < c` evaluates to true rather than, as its author probably intended, to false, and the expression `a == b == c` evaluates to false rather than, as its author probably intended, to true.

Compliant Solution

Treat relational and equality operators as if it were invalid to chain them:

```
if ( (a < b) && (b < c) ) /* Clearer and probably what was intended */
/* ... */
if ( (a == b) && (a == c) ) /* Ditto */
```

Risk Assessment

Incorrect use of relational and equality operators can lead to incorrect control flow.

Rule	Severity	Likelihood	Remediation Cost	Priority	Level
EXP13-C	Low	Unlikely	Medium	P2	L3

Automated Detection

Tool	Version	Checker	Description
Astrée	19.04	**chained-comparison**	Fully checked
ECLAIR	1.2	**CC2.EXP13**	Fully implemented
GCC	4.3.5		Option `-Wparentheses` warns if a comparison like `x<=y<=z` appears; this warning is also enabled by `-Wall`
LDRA tool suite	9.7.1	**433 S**	Fully implemented
Polyspace Bug Finder	R2018a	Possibly unintended evaluation of expression because of operator precedence rules	Operator precedence rules cause unexpected evaluation order in arithmetic expression
PRQA QA-C	9.5	**3392** **3401** **4111** **4112** **4113**	Fully implemented

PVS-Studio	6.23	**V709**	
RuleChecker	19.04	**chained-comparison**	Fully checked

Related Guidelines

SEI CERT C++ Coding Standard	VOID EXP17-CPP. Treat relational and equality operators as if they were nonassociative

Bibliography

[ISO/IEC 9899:2011]	Subclause 6.5.8, "Relational Operators"