The New C Standard (Excerpted material)

An Economic and Cultural Commentary

Derek M. Jones
derek@knosof.co.uk

6.5.14 Logical OR operator

logical-OR-expression syntax

1256

```
logical-OR-expression:
                logical-AND-expression
                logical-OR-expression || logical-AND-expression
```

Commentary

logical-AND-expression syntax

The discussion on the logical-AND operator is applicable here.

Other Languages

Ada supports two kinds of logical-OR operations: **or** and **or else**. The latter has the same semantics as the logical-OR operator in C (short-circuit evaluation). Support for an unsigned integer type was added in Ada 95 and the definition of the logical operators was extended to perform bitwise operations when their operands had this type.

Coding Guidelines

inclusive-OR-expression syntax

The issue of swapping usages of the | and || operators is discussed elsewhere.

conditionals conjunctive/disjunctive

In English (and other languages) the word *or* is sometimes treated as having an exclusive rather than an inclusive meaning. For instance, "John has an M.D. or a Ph.D." is likely to be interpreted in the exclusive sense, while "John did not buy ice cream or chips" is likely to be interpreted in the inclusive sense (e.g., John did not buy ice cream and did not buy chips); the exclusive sense supports surprising possibilities (e.g., John buying ice cream and buying chips). A number of studies[1] have found that people make more mistakes when answering questions that involve disjunctive (i.e., the logical-OR operator) relationships than when answering questions that involve conjunctive (i.e., the logical-AND operator) relationships.

A study by Noveck, Chierchia, and Sylvestre[2] (performed using French) found that the exclusive interpretation occurs when a conclusion $Q or R$ is more informative or relevant and is prompted by an implication (e.g., *but not both*).

Usage

logical-AND-expression syntax

Usage information is given elsewhere.

Constraints

Each of the operands shall have scalar type.

1257

Commentary

&& operand type

The discussions in the various subsections of the logical-AND operator are applicable here.

C++

5.15p1

> *The operands are both implicitly converted to* **bool** *(clause 4).*

Boolean conversions (4.12) covers conversions for all of the scalar types and is equivalent to the C behavior.

Semantics

The || operator shall yield 1 if either of its operands compare unequal to 0;

1258

|| operand compared against 0

Commentary

&& operand compare against 0

The discussion on the logical-AND operator is applicable here.

C++

It returns **true** *if either of its operands is* **true**, *and* **false** *otherwise.* 5.15p1

The difference in operand types is not applicable because C++ defines equality to return **true** or **false**. The difference in return value will not cause different behavior because **false** and **true** will be converted to 0 and 1 when required.

1259 otherwise, it yields 0.

Commentary

That is, a value of zero having type **int**.

1260 The result has type **int**.

||
result type

Commentary

The discussion in the various subsections of the logical-AND operator are applicable here.

&&
result type

C++

The result is a **bool**. 5.15p2

The difference in result type will result in a difference of behavior if the result is the immediate operand of the **sizeof** operator. Such usage is rare.

1261 Unlike the bitwise | operator, the || operator guarantees left-to-right evaluation;

Commentary

The discussion on the logical-AND operator is applicable here.

&&
evaluation order

1262 there is a sequence point after the evaluation of the first operand.

operator ||
sequence point

Commentary

The discussion on the logical-AND operator is applicable here.

&&
sequence point

C++

All side effects of the first expression except for destruction of temporaries (12.2) happen before the second expression is evaluated. 5.15p2

The differences are discussed elsewhere.

&&
sequence point

1263 If the first operand compares unequal to 0, the second operand is not evaluated.

Commentary

The discussion on the logical-AND operator is applicable here. An alternative way of looking at this operator is that `x || y` is equivalent to `x ? 1 : (y?1:0)`.

&&
second operand

References

1. J. S. B. T. Evans and S. E. Newstead. A study of disjunctive reasoning. *Psychological Research*, 41(4):373–388, Apr. 1980.

2. I. Noveck, G. Chierchia, and E. Sylvestre. Linguistic-pragmatic factors in interpreting disjunctions. *Thinking and Reasoning*, 8(4):297–326, 2002.