# **N3459** - Integer and arithmetic constant expressions

**Author:** Javier A. Múgica

## Versions

**Initial version:** 2024 - oct - 03
**this version:** 2025 - feb - 03

## Contents



## The problem

The standard uses the sentence *the result is an integer constant expression*, applied to **`sizeof`** and **`alignof`** expressions. This is not right, since what is an integer constant expression is the expression itself, as the term conveys; the result in those cases is an integer constant; it is a property of the value of the expression, or of the evaluation of the expression, and exists only if the expression is evaluated. *constant expression* is something which, among other requirements, has the form of a conditional expression, so it cannot be referred to as the result of an evaluation.

"Integer constant" was the term used until recently in those places, which was changed by the adoption of the proposal N3239 *Some constants are literally literals*. As that proposal argues, the use of "integer constant" there was incorrect because this term is defined as a certain category of tokens. That paper changed the term to "integer literal" and changed the sentence *the result is an integer constant* to append *expression* after it. In other places, the proposal changed this to "constants of integer type".

We like the choice of N3239 of talking about **`sizeof`** and **`alignof`** expressions as integer constant expressions, but has to be expressed in some other form. For this, a wording is proposed in "First change".

The second change proposes a simplification of the wording for arithmetic constant expressions. The third change proposes a reelaboration of the wording for those two kinds of consant expressions.

## Wording

## First change

In 6.5.4.5 The **`sizeof`**, **`_Lengthof`** and **`alignof`** operators, pp. 2, 3 and 5, change

the result is an integer constant expression *to* the expression is an integer constant expression

In 6.6 Constant expressions, pp. 8 and 10 and in appendix J.2, items 50 and 52, change

whose results are integer constant expressions *to* which are integer constant expressions

In 6.9 External definitions, in the constraints,

— part of the operand of a **sizeof** operator whose result is an integer constant expression;

— part of the operand of a **_Lengthof** operator whose result is an integer constant expression;

— part of the operand of an **alignof** operator whose result is an integer constant expression;

change those items to

— part of the operand of a **sizeof** expression which is an integer constant expression;

— part of the operand of a **_Lengthof** expression which is an integer constant expression;

— part of the operand of an **alignof** operator;

## Second change

In the paragraph on arithmetic constant expressions, the operands “integer literals”, “character literals” **sizeof**, **_Lengthof** and **alignof** expressions can be included in "integer constant expressions":

10 An *arithmetic constant expression* shall have arithmetic type and shall only have operands that are floating literals, named or compound literal constants of arithmetic type and integer constant expressions. Cast operators in an arithmetic constant expression shall only convert arithmetic types to arithmetic types, except as part of an operand to the typeof operators, **sizeof** operator, **_Lengthof** operator or **alignof** operator.

## Third change

Finally, we understand that the current wording does not allow an expression like `3` or `'a'` to be an ICE by itself, for example in **int** `a[3];` or **case** `'a':`, since there `3` and `'a'` are not operands of anything. The opposite interpretation; namely, that becuase they are not operands none of those *shall*'s apply to them, would make an ICE of any identifier with integer type, which is obviously not. Even if it is understood that it does allow those expressions, we believe it is clearer to state that literals of the appropriate type are integer/arithmetic constant expressions.

Furthermore, it is not clear that **_Generic** selections may include other kinds of expressions in the generic associations which are not taken.

We propose the following wording in accordance to this:

8 An *outermost operand* of an expression or typeof specifier is an operand of the same for which there does not exist another expression or type name lying strictly between the operand and the expression or typeof specifier.

9 *syntactic integer constant expressions* (s.i.c.e.) and *integer constant expressions* are defined recursively as follows:

10 Every integer constant expressions is a syntactic integer constant expression.

11 The following are integer constant expressions: integer literals, named and compound literal constants of integer type, character literals, **alignof** expressions and some **sizeof** and **_Lengthof** expressions, as specified in 6.5.4.5. Cast expressions where the type of the cast is an integer type and the operand is a named or compound literal constant of arithmetic type are syntactic integer constant expressions.

12 In addition, the following are syntactic integer constant expressions:

— a parethesized s.i.c.e.

— An expression of integer type other than a generic selection whose outermost operands are s.i.c.e.

— A generic selection where the result expression is a s.i.c.e.

13 A syntactic integer constant expressions that evaluates to a constant that is in the range of representable values for its type is an integer constant expression.

15 The following are *syntatic arithmetic constant expressions* (s.a.c.e.): syntactic integer constant expressions, floating literals and named or compound literal constants of arithmetic type. Other syntactic arithmetic constant expressions are defined recursively:

— a parethesized s.a.c.e.

— An expression of arithmetic type other than a generic selection whose outermost operands are s.a.c.e.

— A generic selection where the result expression is a s.a.c.e.

16 An *arithmetic constant expression* is a syntactic arithmetic constant expressions that evaluates to a constant that is in the range of representable values for its type.

## Notes on the third change

1. The term "syntactic" in "syntactic integer/arithmetic constant expression" may be changed to "potential", "candidate" or any other term the committee whishes to adopt.

2. In the sentence *An expression of integer type other than a generic selection whose outermost operands are s.i.c.e.*, the clause "of integer type" is so that in a cast expression the type need be an integer type, and similarly for s.a.c.e.

3. With the wording of the current draft, **`alignof`**, **`sizeof`** and **`_Lengthof`** expressions which are integer constant expressions are defined as such twice. First, in the subclause where these operators are defined, then in "Constant expressions". This has been so ever since they were defined as integer constant expression at the place where the operators are defined.

Althought there is no harm in it, we have adopted a wording that avoids it, by defining syntactic integer constant expressions and integer constant expressinons recursively, jointly. The definition of certain **`sizeof`** and **`_Lengthof`** expressions as integer constant expressions at the place these operands are defined are part of, and at the same time need, the recursive definition, for if the operand to any of those operators is the type name of an array type, the expression is or is not an i.c.e. according to whether the size of the array is given by an i.c.e. or not (and that of the element's type, if it is itself an array, etc.).