N3465 - Preprocessor integer expressions

Author: Javier A. Múgica

The problem

The expression that must follow an **`#if`** or **`#elif`** directive after macro replacement and final simplification is "integer constant expression". Many constructions that are possible in those expressions in general cannot appear in the controlling expression of those directives. Roughly, anything involving the type system. Proposal N3464 would enlarge the kind of subexpressions allowed in contexts that are discarded relative to the expression (in the terminology of that paper). It is not reasonable to require the preprocessor to handle all of those. For example:

```
1 ? 1 : (""[0] += 5)
```

(Example from J. Myers).

Even without the extensions from that paper, this problem already exists. Implementation are allowed to enlarge the class of expressions they consider integer constant expressions. Thus, an implementation is currently allowed to consider the above an i.c.e. with value 1. It is unlikely that those implementations want their preprocessors to handle those cases.

Proposed solution

Since what the preprocessor can handle is very limited, we think it is better to provide a direct definition of what is allowed as the controlling expression for those directives. It should be noted that the constraint that the expression shall be an integer constant expression already forces it to be a syntactically valid conditional expression and places some constraints in it.

Hence, we propose to insert one more constrain in **Conditional inclusion**:

11 After each preprocessing token has been converted into a token, the resulting tokens shall be solely of the following kind: integer literal, character literal and punctuator.

We note that this forces any well formed expression (i.e., any expression) that can be evaluated to be an integer constant expression, except for the restriction on the comma operator.