N3505 - Preprocessor integer expressions, II

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 constraint the type of tokens that can result after preprocessing tokens have been converted to tokens. In order to addresss the comments raised by the previous version of this paper, we add the possibility that some implementation define other sequences of tokens that may remain.

In addition, we note that the conversion of pp-tokens to tokens is the last step in the transformation of the tokens that follow the **`#if`** or **`#elif`** directive, when all **`defined`**, **__has_...** operators have been processed, all macros expanded and remaining identifiers which are not defined replaced by `0` and those which are defined as macro names replaced by `1`.

Add the following constraint at the end of the **Constraints** section in **Conditional inclusion**:

11 After each preprocessing token has been converted into a token, the resulting tokens shall be solely integer literals, character literals, punctuators and other implementation-defined sequences of tokens.

The wording “implementation-defined sequences of tokens” instead of “implementation defined tokens” has been chosen because the latter seems to say that it is the tokens themselves which are implementation defined. What is implementation defined is which tokens may reamin (which may include implementation-defined tokens).