WG14/N455
X3J11/95-05

# Defect Report NNI-1

Submission date: 21 Aug 95
Submittor: NNI
Source: E.G. Keizer

**Short description of problem:**

The Standard contains conflicting words on whether "f()" can be considered a call to function-like macro with one empty argument.

**Problem description:**

There seems to be a problem with empty arguments in macro calls in the current standard. An example:

```
#define foo()    A
#define bar(a)   B

foo()            // no arguments
bar()            // one empty argument ??
```

There seem to be two choices when there are no preprocessing tokens between the ( and the ) in a macro call: a single empty argument or no arguments.
It is generally accepted that the call to foo has no arguments.
The call to bar is different, according to some the call violates a constraint, others are of the opinion that the call to bar is undefined behavior and that is can be seen a a call with a single, empty argument..

Quotes from paragraph 6.8.3 of the Standard: fourth paragraph of constraints:

> The number of arguments in an invocation of a function-like macro shall agree with the number of parameters in the macro definition, and there shall exist a ) preprocessing token that terminates the invocation.

last paragraph of semantics:

> If (before argument substitution) any argument consists of no preprocessing tokens, the behavior is undefined.

The question

> is the call to bar in the example a constraint violation or undefined behavior?

seems to show an ambiguity in the Standard.

The X3J11 archives contain several requests to allow empty macro arguments and refusals to do so. The refusals stated that the reasons would be included in the rationale, but the rationale is silent on this matter. Hearsay indicates that X3J11 decided to leave the issue of empty macro arguments open.

**Solutions:**

There seem to be four ways to change the Standard:

1. Resolve the ambiguity by considering "bar()" and "foo()" calls with no parameters.
2. Resolve the ambiguity by considering "bar()" and "foo()" calls with one empty parameter.
3. Resolve the ambiguity by making the interpretation of the empty preprocessing token sequence context dependent and consider "bar()" a call with one empty parameter and "foo()" a call without parameters.
4. Leave the issue open by making clear that empty arguments are undefined behavior and that a call without preprocessing tokens between the parenthesis to a function-like macro defined with one parameter does not violate a constraint.

**Arguments for a choice:**

- Solution 2 causes a constraint violation in "foo()" and thus forbids calls to macros defined without parameters. Argument-less macros are often used in current C programs. Thus making this change would violate the first part of guiding principle 1 of the C9X charter: Existing code is important.
- Solution 4 allows compiler writers to choose between solution 1 and solution 3.
- Solutions 1 and 2 are in conflict with N418, the proposal to allow empty arguments in macro replacements
- Solution 3 conflicts with our rule to "keep it simple" and will create confusion. The following example illustrates serves as an illustration.
  A user had a program in which he used a function-like macro without arguments. While doing an overhaul of the program he decided that he needed one argument and changed the function definition accordingly. When he edited his program to change the calls, he forgot one call. His program compiled, but showed strange behavior. After several hours of debugging he found the one call with the missing arguments and was very surprised that the compiler had not complained.

**Proposal**

The arguments above led us to believe that allow empty arguments to function-like macros would be bad from a software engineering viewpoint and cause confusion. Thus we propose to use solution 1. This can be done by inserting the following sentence after the sentence from the semantics section of 6.8.3 quoted above.

> A call to a function-like macro without preprocessing tokens between the opening and closing parenthesis is taken to be a call without arguments, not a call with a single empty argument.