Proposal for C23
WG14 N2848

Title: Contradiction about INFINITY macro
Author, affiliation: C FP group
Date: 2021-10-13
Proposal category: Editorial
Reference: N2596

This proposal is based on issues reported to CFP by Vincent Lefevre and others.

5.2.4.2.2#16 says

> [16] The macro **`INFINITY`** expands to a constant expression of type **`float`** representing positive or unsigned infinity, if available; else to a positive constant of type **`float`** that overflows at translation time. 29)
>
> 29)In this case, using **`INFINITY`** will violate the constraint in 6.4.4 and thus require a diagnostic.

There are two problems specifically with the "else" case.

Problem 1. The constraint in 6.4.4 is

> [2] Each constant shall have a type and the value of a constant shall be in the range of representable values for its type.

If the type does not have infinities, a constant must be in the range of finite representable numbers and would not overflow, making it impossible to define **`INFINITY`** as a constant that overflows.

Problem 2. Wide expression evaluation methods evaluate **`float`** constants to a format wider than **`float`**. To assure an "overflow" constraint violation, the number represented must exceed the range of the evaluation format.

There is also a more fundamental problem.

Problem 3. The "else" case gives a constraint violation, which could be useful only in implementation-specific ways. And, any "else" case definition means the **`INFINITY`** macro cannot be used for a feature test. This is unlike the **`NAN`** macro, which is defined if and only if quiet NaNs are supported in type **`float`**.

With the recommended change below, the **INFINITY** macro (analagous to the **NAN** macro) is defined if and only if infinity is supported in type **float**. This removes the "else" case and resolves all three problems.

The recommended change could break code that depended on the way an implementation treated the constraint violation.

In case the recommend change is not acceptable, we provide an alternate change that is intended to resolve the first two problems and retain the intended meaning of the current specification.

Suggested change:

In 5.2.4.2.2 change:

> [16] The macro **INFINITY** is defined if and only if the implementation supports an infinity for the type **float**. It expands to a constant expression of type **float** representing positive or unsigned infinity~~, if available; else to a positive constant of type **float** that overflows at translation time~~. ~~29)~~
>
> ~~29)In this case, using **INFINITY** will violate the constraint in 6.4.4 and thus require a diagnostic.~~

Alternate change:

In 5.2.4.2.2 change:

> [16] The macro **INFINITY** expands to a constant expression of type **float** representing positive or unsigned infinity, if available; else to a ~~positive constant of type **float** that overflows at translation time~~ character sequence in the form of a constant of type **float** (6.4.4.2) and representing a positive number beyond the range of representable values in the evaluation format for type **float**. 29)
>
> 29)In this case, using **INFINITY** will violate the constraint in 6.4.4 and thus require a diagnostic.