Proposal for C23
WG14 N2880

Title: overflow and underflow definitions (N2805 update)
Author, affiliation: C FP group
Date: 2021-12-23
Proposal category: Technical
Reference: N2596, N2746, N2805

This document updates N2805 (which updated N2746). N2805 was approved at the November 2021 WG 14 meeting.

1. This update broadens the overflow definition to allow an overflow threshold beyond the largest normalized floating-point number. This change accommodates the extra full-precision range of double-double formats.

2. This update also loosens the current requirement to return the value of the **HUGE_VAL** macro of the type when the result overflows by allowing functions to return a number with reduced precision instead. This change accommodates the extra reduced-precision range in some double-double implementations and allows returning the best result (for those implementations) on overflow. The alternative of raising the overflow threshold (to allow reduced-precision results without an overflow) is not desirable because it might silently undermine reasonable user expectations about precision.

3. Considerations in item 2 highlighted an ambiguity in the specification of the **HUGE_VAL** macros, one that is particularly troublesome for double-double implementations with extra reduced-precision range. It is unclear how these implementations could define the **HUGE_VAL** macros. 7.12 #5, where the **HUGE_VAL** macros are introduced, does not explicitly specify a value for them. 7.12.1 #5, 7.22.1.5 #10, and 7.29.4.1.1 #10 implicitly specify the value to be the value returned when a library function result overflows under the default rounding mode. Some (most?) users believe the **HUGE_VAL** macros give the largest number in the type (infinity or the value of *type***_MAX**). (Thus, the meaning of the **HUGE_VAL** macros is unclear. Is it the overflow result for the default rounding mode, or is it the maximum number in the type. Reduced-precision overflow results (see item 2) would vary from cases to case, depending on the severity of overflow. However, the implicit specification for the macro values relies on overflow results being fixed under the default rounding mode. Another problematic case (unrelated to double-double) for the current specification of the **HUGE_VAL** macros occurs if library function results are rounded toward zero but the maximum number in the type is infinity. The suggested changes below address this issue.

4. The specifications for the **nextup** (7.12.11.5) and **nextdown** (7.12.11.6) functions refer to the **HUGE_VAL** macros, though the intention of the functions — to

step through representable values — is not related to overflow or rounding mode. A suggested editorial change below expresses the intention of the specification without referring to the **HUGE_VAL** macros.

5. The definition of "correctly rounded" in 3.9 is incorrect in the case of overflow if the result format contains infinity. The suggested editorial change below adds a note to address this case.

Suggested changes (change marks relative to N2596)**:**

NOTE At the end we show the new changes suggested by this document relative to N2596 + N2805.

Change in 3.9:

> [3] **Note 2 to entry:** IEC 60559 or implementation-defined rules apply for extreme magnitude results if the result format contains infinity.

Changes in 7.12.1:

> [5] A floating result overflows if ~~the~~ a finite result value with ordinary accuracy*) would have magnitude (absolute value) ~~of the mathematical result is finite but so large that the mathematical result cannot be represented without extraordinary roundoff error~~ too large for representation with full precision in the specified type. A result that is an exact infinity does not overflow. If a floating result overflows and default rounding is in effect, then the function returns the value of the macro **HUGE_VAL**, **HUGE_VALF**, or **HUGE_VALL** according to the return type, with the same sign as the correct value of the function; however, for types with reduced-precision representations of numbers beyond the overflow threshold, the function may return a representation of the result with less than full precision for the type. If a floating result overflows and ~~if~~ the integer expression **math_errhandling & MATH_ERRNO** is nonzero, the integer expression **errno** acquires the value **ERANGE**~~;~~. If a floating result overflows and ~~if~~ the integer expression **math_errhandling & MATH_ERREXCEPT** is nonzero, the "overflow" floating-point exception is raised.
> . ...
>
> [6] The result underflows if ~~the~~ a nonzero result value with ordinary accuracy would have magnitude (absolute value) ~~of the mathematical result is nonzero and~~ less than the minimum normalized number in the type; however, a zero result that is specified to be an exact zero does not underflow. Also, a result with ordinary accuracy and the magnitude of the minimum normalized number may underflow.249) ...

*)Ordinary accuracy is determined by the implementation. It refers to the accuracy of the function where results are not compromised by extreme magnitude.

249)The term underflow here is intended to encompass both "gradual underflow" as in IEC 60559 and also "flush-to-zero" underflow. IEC 60559 underflow can occur in cases where the magnitude of the rounded result (accurate to the full precision of the type) equals the minimum normalized number in the format.

Change 7.12.1 #5:

[5] The macro

HUGE_VAL

expands to a positive **double** constant expression, not necessarily representable as a **float**~~.~~, whose value is the maximum value returned by library functions when a floating result of type **double** overflows under the default rounding mode, either the maximum finite number in the type or positive or unsigned infinity.

In 7.12.11.5 #2, change the last sentence:

~~**nextup(HUGE_VAL)** is **HUGE_VAL.**~~ If *x* is the positive number (finite or infinite) of maximum magnitude in the type, **nextup**(*x*) is *x*.

In 7.12.11.6 #2, change the last sentence:

~~**nextdown(-HUGE_VAL)** is **-HUGE_VAL.**~~ If *x* is the negative number (finite or infinite) of maximum magnitude in the type, **nextdown**(*x*) is *x*.

Suggested changes (change marks relative to N2596 + N2805**):**

Change in 3.9:

[3] **Note 2 to entry:** IEC 60559 or implementation-defined rules apply for extreme magnitude results if the result format contains infinity.

Changes in 7.12.1:

[5] A floating result overflows if a finite result value with ordinary accuracy*) would have magnitude (absolute value) ~~larger than the maximum normalized number~~too large for representation with full precision in the

specified type. ~~(~~A result that is an exact infinity does not overflow.~~)~~ If a floating result overflows and default rounding is in effect, then the function returns the value of the macro **HUGE_VAL**, **HUGE_VALF**, or **HUGE_VALL** according to the return type, with the same sign as the correct value of the function; however, for types with reduced-precision representations of numbers beyond the overflow threshold, the function may return a representation of the result with less than full precision for the type. If a floating result overflows and ~~if~~ the integer expression **math_errhandling & MATH_ERRNO** is nonzero, the integer expression **errno** acquires the value **ERANGE**~~;~~. If a floating result overflows and ~~if~~ the integer expression **math_errhandling & MATH_ERREXCEPT** is nonzero, the "overflow" floating-point exception is raised.

. ...

In 7.12.11.5 #2, change the last sentence:

~~**nextup(HUGE_VAL)** is **HUGE_VAL.**~~ If *x* is the positive number (finite or infinite) of maximum magnitude in the type, **nextup**(*x*) is *x*.

In 7.12.11.6 #2, change the last sentence:

~~**nextdown(-HUGE_VAL)** is **-HUGE_VAL.**~~ If *x* is the negative number (finite or infinite) of maximum magnitude in the type, **nextdown**(*x*) is *x*.