Proposal for C2Y
WG14 N3737

Title:	Refine the language of error reporting
Author, affiliation:	CFP group
Date:	2025-11-09
Proposal category:	Editorial
References:	N3685, N3405

Rationale:

Proposal N3405 added language now in the draft to clarify the difference between the notion of an "exceptional computational condition" and whether an "error occurs" in the sense of the standard. This proposal follows up on N3405 to address a few places requiring clearer application of these concepts. This proposal also simplifies the organization of 7.12.2.

The first change emphasizes that the word "occurs" has the specific stated meaning introduced by N3405. This strengthens the concept that an error "occurs" only if it is reported. (Some exceptional circumstances are not reported.)

The second change is a simplification that brings the concept of "may define additional errors" up to the second paragraph. Removing the redundancies streamlines the discussions of the individual cases and clarifies that this is a uniform concept.

A byproduct of the second change is the removal of footnote 275). We see nothing that precludes the specification for errors from applying to infinities, so there is no need to address this with a footnote.

The final changes avoid the mistaken implications that overflow and underflow exceptional conditions are always reported. They also clarify over/underflow behavior with regard to the rounding mode.

Suggested changes:

Changes in 7.12.2#2:

From:
An error is said to ~~occur~~ when...

To:
An error is said to *occur* when…

From:
... Not all exceptional conditions are required to be reported as errors.

To:

… Not all exceptional conditions are required to be reported as errors. The description of each function says an error “occurs” to indicate any required errors. An implementation may define additional errors, provided that such errors are consistent with the mathematical definition of the function and the meaning of the exceptional condition. Required and implementation-defined errors are reported as specified in this subclause. Function descriptions say an error “may occur” to indicate some of the cases suitable for implementation-defined errors.

Change in 7.12.2#3:

Delete:

…~~The description of each function lists any required domain errors; an implementation may define additional domain errors, provided that such errors are consistent with the mathematical definition of the function.275)~~. …

Change in footnote 275):

Delete:

~~275) In an implementation that supports infinities, this allows an infinity as an argument to be a domain error if the mathematical domain of the function does not include the infinity.~~

Change in 7.12.2#4:

Delete:

… ~~The description of each function lists any required pole errors; an implementation may define additional pole errors, provided that such errors are consistent with the mathematical definition of the function.~~ …

Change in 7.12.2#5:

From:

… ~~The description of each function lists any required range errors; an implementation may define additional range errors, provided that such errors are consistent with the mathematical definition of the function and are the result of either overflow or underflow. Range errors that are required or implementation-defined shall or may be reported, as specified in this subclause, respectively.~~

To:

… Range errors in non-default rounding modes are implementation-defined.*)

*) The Annex F specification for “overflow" and “underflow" floating-point exceptions applies to all IEC 60559 rounding modes.

Change in 7.12.2#6:

From:

… ~~If a floating result overflows and default rounding is in effect and the integer expression~~ ~~`math_errhandling & MATH_ERRNO`~~ ~~is nonzero, then the integer expression~~ ~~`errno`~~ ~~acquires the value~~ ~~`ERANGE`~~~~. If a floating result overflows, and the integer expression~~

~~`math_errhandling & MATH_ERREXCEPT` is nonzero, the "overflow" floating-point exception is raised (regardless of whether default rounding is in effect).~~

To:

… If a range error occurs because of floating-point overflow and the integer expression `math_errhandling & MATH_ERRNO` is nonzero, the integer expression `errno` acquires the value `ERANGE`. If a range error occurs because of floating-point overflow and the integer expression `math_errhandling & MATH_ERREXCEPT` is nonzero, the "overflow" floating-point exception is raised.

Change in 7.12.2#7:

From:

… If the result underflows, the function returns an implementation-defined value whose magnitude is no greater than the smallest normalized positive number in the specified type~~; if~~ the integer expression ...

To:

… If the result underflows, the function returns an implementation-defined value whose magnitude is no greater than the smallest normalized positive number in the specified type. If a range error occurs because of underflow and the integer expression ...