Proposal for C23
WG14 N2747

Title: Annex F overflow and underflow
Author, affiliation: C FP group
Date: 2021-05-20
Proposal category: Editorial
Reference: N2596

F.10 characterizes when floating-point "underflow" and "overflow" exceptions are raised:

[8] The "overflow" floating-point exception is raised whenever an infinity — or, because of rounding direction, a maximal-magnitude finite number — is returned in lieu of a value whose magnitude is too large.

[9] The "underflow" floating-point exception is raised whenever a result is tiny (essentially subnormal or zero) and suffers loss of accuracy.

... 402)

[11] Whether or when library functions raise an undeserved "underflow" floating-point exception is unspecified.403) Otherwise, as implied by F.8.6, these functions do not raise spurious floating-point exceptions (detectable by the user), other than the "inexact" floating-point exception.

403) It is intended that undeserved "underflow" and "inexact" floating-point exceptions are raised only if avoiding them would be too costly.

Problem 1: The underflow characterization in #9 is from IEEE 754-1985 and does not correctly reflect the current IEC 60559 specification for underflow.

Problem 2: #11 is missing the qualification “not bound to operations in IEC 60559” which was recently added in similar contexts.

Problem 3: #11 allows spurious "underflow" floating-point exceptions. However, C (7.12.1) does not allow spurious underflow range errors. Therefore, implementations supporting range errors via floating-point exceptions must avoid raising spurious "underflow" floating-point exceptions that do not meet the C definition of underflow. It would be helpful to note this in Annex F.

Problem 4: Footnote 403) to #11 uses “underserved” instead of “spurious” which is used in similar contexts in C.
**Problem 5:** The overflow characterization in F.10 #8 might erroneously suggest that raising an “overflow” floating-point exception would be appropriate for the computation of an exact infinity.

The suggested changes below address these problems. They can be regarded as editorial since Annex F adopts IEC 60559 by reference. We do not suggest including the complete definition of IEC 60559 underflow because the details are esoteric and so rarely matter to users.

**Suggested change:**

Changes in F.10:

[8] The "overflow" floating-point exception is raised whenever an infinity — or, because of rounding direction, a maximal-magnitude finite number — is returned in lieu of a finite value whose magnitude is too large.

[9] The "underflow" floating-point exception is raised whenever a computed result is tiny (essentially subnormal or zero) and suffers loss of accuracy 402) and the returned result is inexact.

... 402)

[11] Whether or when library functions not bound to operations in IEC 60559 (F.3) raise an undeserved a spurious "underflow" floating-point exception is unspecified not specified by this annex 403) Otherwise, as implied by F.8.6, these functions do not raise spurious floating-point exceptions (detectable by the user), other than the "inexact" floating-point exception.

[11a] As implied by F.8.6, library functions do not raise spurious “invalid”, “overflow”, or “divide-by-zero” floating-point exceptions (detectable by the user).

402) IEC 60559 allows different definitions of underflow. They all result in the same values, but differ on when the floatingpoint exception is raised. Tiny generally indicates having a magnitude in the subnormal range. See IEC 60559 for details about detecting tininess.

403) It is intended that undeserved spurious "underflow" and "inexact" floating-point exceptions are raised only if avoiding them would be too costly. 7.12.1 specifies that if math.errhandling & MATH_ERREXCEPT is nonzero, then an “underflow” floating-point exception shall not be raised unless an underflow range error occurs.