

The New C Standard (Excerpted material)

An Economic and Cultural Commentary

Derek M. Jones

derek@knosof.co.uk

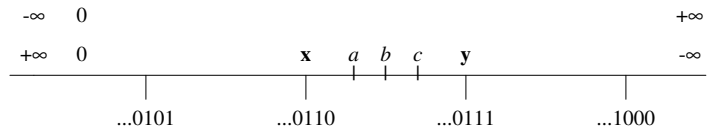


Figure 64.1: Some exactly representable values and three values (*a*, *b*, and *c*) that are not exactly representable.

3.9

correctly rounded result

correctly rounded result

representation in the result format that is nearest in value, subject to the effective current rounding mode, to what the result would be given unlimited range and precision

Commentary

In Figure 64.1 the value *b* is midway between the representable values *x* and *y*. The representable value it will round to depends on the sign of its value, whether the rounding mode is round-to-even (least significant bit not set), round-to-zero, or round-to \pm infinity. The values *a* and *c* have a *nearest* value to round to, if that rounding mode is in effect.

FLT_ROUNDS

$DBL_MAX * (1 + DBL_EPSILON)$ is *nearest* to DBL_MAX (in the Euclidean sense) but it is rounded to infinity.

The result referred to here is the result of a single arithmetic operation, not the result of an expression containing more than one operation. For instance, the expression $a + b - c$ consists of the operation $a + b$, whose result then has *c* subtracted from it. If $a = 1$, $b = 1E-25$, and $c = 1$, the correctly rounded result (assuming IEC 60559 single-precision) of $a + b$ is 1, and after the subtraction operation the final result of the complete expression is 0 (the expression can be rewritten to return the result $1E-25$).

The notation used to indicate rounded arithmetic operations is to place a circle around the operator; for instance:

$$x \oplus y = \text{round}(x + y) \tag{64.1}$$

where $x + y$ denotes the exact mathematical result. Based on the preceding example it is obvious that, in general:

$$a \oplus b \ominus c \neq a \ominus c \oplus b \tag{64.2}$$

The issue of correctly rounded results for some of the functions defined in the header `<math.h>` is discussed by Lefèvre and Muller.^[2]

The wording was changed by the response to DR #286.

C++

This term is not defined in the C++ Standard (the term *rounded* only appears once, when discussing rounding toward zero).

Common Implementations

Implementations that don't always round correctly include some Cray processors and the IBM S/360 (truncates).

Some implementations use a different rounding process when performing I/O than when performing floating-point operations. For instance, should the call `printf("%.0f", 2.5)` produce the same correctly rounded result (as output on `stdout`) as the value stored in an equivalent assignment expression?

Parks^[3] discusses methods for generating test cases that stress various boundary conditions that occur when rounding floating-point values.

Coding Guidelines

The fact that the term *correctly rounded result* can denote different values under different circumstances is something that should be included in the training of developers' who write code that uses floating-point types.

Obtaining the correctly rounded result of an addition or subtraction operation requires an additional bit in the significand (as provided by the IEC 60559 guard bit) to hold the intermediate result.^[1] When multiplying two p bit numbers the complete $2p$ bits of the intermediate result need to be generated, before producing the final correctly rounded, p bit, result. A floating-point calculation that frequently occurs, for instance, in the evaluation of the determinant of a matrix is:

```
1 float d, w, x, y, z;
2
3 /* Use casts in case the minimum evaluation format is float. */
4
5 d = ((double)w) * x - ((double)y) * z;
```

If the relative precision of the type **double** is at least twice that of the type **float**, then the result will be correctly rounded (implementations where the types **float** and **double** have the same precision are not uncommon).

References

1. D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1):5–48, Mar. 1991.
2. V. Lefèvre and J.-M. Muller. Worst case for correct rounding of the elementary functions in double precision. Technical Report No. 4004, Unité de recherche INRIA Rhône-Alpes, Nov. 2000.
3. M. Parks. Number-theoretic test generation for directed rounding. *IEEE Transactions on Computers*, 49(2):651–658, July 2000.