N1538
Jim Thomas and Fred Tydeman
2010/11/04

The question: Must promoting casts remove extra range and precision?

At least three sections in the draft say or assume that they do.

A key Annex F goal is that use of IEEE types and standard evaluation methods give predictable results - which requires that they do.

There's a vein of understanding that the draft is intended to not require that they do.

Assuming some implementations do retain extra range and precision in promoting casts (I suspect they do) there's a valid concern that changing a compiler to removing extra range and precision might change some code results in a way that would disturb a user.

This proposal leaves it unspecified whether promoting casts remove extra range and precision, and requires it for Annex F implementations.

The following changes are intended to clarify when the result of a conversion might have extra range and precision.

5.2.4.2.2 #9 Change

> Except for assignment and cast (which remove all extra range and precision),

to

> Except for assignment and cast,

6.3.1.4 #2 (integer to floating conversion): Append

> Results of conversions may be represented with extra range and precision, except where explicitly stated otherwise.

6.3.1.5 #1, #2 change

> When a float is promoted to double or long double, or a double is promoted to long double, its value is unchanged (if the source value is represented in the precision and range of its type).

> When a double is demoted to float, a long double is demoted to double or float, or a value being represented in greater precision and range than required by its semantic type (see 6.3.1.8) is explicitly converted (including to its own type), if the value being converted can be represented exactly in the new type, it is unchanged. If the value being converted is in the range of values that can be represented but cannot be represented exactly, the result is either the nearest higher or nearest lower representable value, chosen in an implementation-defined manner. If the value being converted is outside the range of values that can be represented, the behavior is undefined.

to

> When a value of real floating-point type is converted to a real floating type, if the value being converted can be represented exactly in the new type, it is unchanged. If the value being converted is in the range of values that can be represented but cannot be represented exactly, the result is either the nearest higher or nearest lower representable value, chosen in an implementation-defined manner. If the value being converted is outside the range of values that can be represented, the behavior is undefined. Results of conversions may be represented with extra range and precision, except where explicitly stated otherwise.

6.3.1.8, footnote 63: change

> 63) The cast and assignment operators are still required to perform their specified conversions as described in 6.3.1.4 and 6.3.1.5.

to

> 63) The cast and assignment operators are still required to perform their specified conversions.

6.5.4 (Cast operator) #6 To the current paragraph

> If the value of the expression is represented with greater precision or range than required by the type named by the cast (6.3.1.8), then the cast specifies a conversion even if the type of the expression is the same as the named type.

add the paragraph

> When a double is demoted by cast to float, a long double is demoted by cast to double or float, or a value being represented in greater precision and range is cast to its own type, then any extra range and precision is removed.

6.8.6.4 footnote 158: Change

> The representation of floating-point values may have wider range or precision than implied by the type; a cast may be used to remove this extra range and precision.

to

> The representation of floating-point values may have wider range or precision than implied by the type. Casts may be used to remove this extra range and precision. For example, any extra range and precision will be removed by (long double)(long double)(f * f), were f is a float and the product is evaluated with more range or precision than long double. Note that two casts may be needed because the promotion of float to long double is not required to remove extra range and precision.

Add a new Annex F section:

> F.? Casts operators
>
> A cast to a floating type converts the value as if by assignment to the type specified by the cast. Thus all casts to floating types remove extra range and precision.

F.6 (The return statement): Change

> If the return expression is evaluated in a floating-point format different from the return type, the expression is converted to the return type of the function and the resulting value is returned to the caller.

to

> If a function has floating return type, the return expression is converted as if by assignment to the type of the function and the resulting value is returned to the caller.