What does -1.#IND mean?: A survey of how the Visual C runtime library prints special floating point values

devblogs.microsoft.com/oldnewthing/20130221-00

February 21, 2013

Raymond Chen

As every computer scientist knows, the IEEE floating point format reserves a number of representations for infinity and non-numeric values (collectively known as NaN, short for not a number). If you try to print one of these special values with the Visual C runtime library, you will get a corresponding special result:

Output	Meaning
`1#INF`	Positive infinity
`-1#INF`	Negative infinity
`1#SNAN`	Positive signaling NaN
`-1#SNAN`	Negative signaling NaN
`1#QNAN`	Positive quiet NaN
`-1#QNAN`	Negative quiet NaN
`1#IND`	Positive indefinite NaN
`-1#IND`	Negative indefinite NaN

Positive and negative infinity are generated by arithmetic overflow, or when the mathematical result of an operation is infinite, such as taking the logarithm of positive zero. (Don't forget that IEEE floating point supports both positive and negative zero.) For math nerds: IEEE arithmetic uses affine infinity, not projective, so there is no point at infinity.

Signaling and quiet NaNs are not normally generated by computations (with one exception noted below), but you can explicitly create one for a floating-point type by using the `std::numeric_limits<T>` class, methods `signaling_NaN()` and `quiet_NaN()`.

Recall that there is not just one signaling and quiet NaN, but rather a whole collection of them. The C runtime does not distinguish among them when printing, however. All signaling NaNs are reported as `1#SNAN`, regardless of the signal bits. The C runtime does report the sign of the NaN, for what little that is worth.

The weird one is the *Indefinite NaN*, which is a special type of quiet NaN generated under specific conditions. If you perform an invalid arithmetic operation like add positive infinity and negative infinity, or take the square root of a negative number, then the IEEE standard requires that the result be a quiet NaN, but it doesn't appear to specify what quiet NaN exactly. Different floating point processor manufacturers chose different paths. The term *Indefinite NaN* refers to this special quiet NaN, whatever the processor ends up choosing it to be.

Some floating point processors generate a quiet NaN with the signal bits clear but the sign bit set. Setting the sign bit makes the result negative, so on those processors, you will see the indefinite NaN rendered as a negative indefinite NaN. (The x86 is one of these processors.)

Other floating point processors generate a quiet NaN with the signal bits and the sign bit all clear. Clearing the sign bit makes the result positive, so on those processors, you will see the indefinite NaN rendered as a positive indefinite NaN.

In practice, the difference is not important, because either way, you have an indefinite NaN.

Raymond Chen

Follow