

Re: On writing negative zero – with or without sign

Source: <http://coding.derkeiler.com/Archive/Fortran/comp.lang.fortran/2007-07/msg01050.html>

- *From:* tholen@xxxxxxxxxxxxx
 - *Date:* 21 Jul 2007 21:01:02 GMT
-

James Giles writes:

....

But, the sign is present like it or not.

Only because you insist that a sign must be present. As I noted in another posting, the situation is more complex than that.

No. The sign bit exists whether you like it or not because the IEEE standard says so. It would still say so whether I insist or not. There's a bit in each floating point value representing the sign. That bit is always present. It has one of two values. That follows from the fact that it's a bit. Nothing you've said in any thread in any newsgroup at any time in the past or even the future will make it more or less complex than that. It's actually pretty simple.

But we're talking about the Fortran standard here, not the IEEE standard.

No sign means the same as a plus sign.

Doesn't have to be that way. Sure, in a binary representation, if you allocate only a single bit to the sign, then you can have only two states. But we're talking about the external representation, where in theory we can have many different characters, including minus, plus, blank, greater than, less than, and even plus-minus.

Re: On writing negative zero – with or without sign

And, you keep failing to specify what magic you think the I/O library is going to use to infer more than one bit of information from the one bit that's present.

Wasn't "interval arithmetic" your magic, Giles?

The sign has two states.

When limited to one bit. Just like the character set has 256 states when limited to 8 bits. But now we have double byte character sets. What's to prevent some new generation of computer and compiler to come along with interval arithmetic and multi-bit sign?

Using a different convention for zero magnitude values still leaves the necessity to report which of those two states the sign bit was in.

Suppose the convention was changed such that an extra bit was allocated for the sign? We've gone from four-bit processors to eight-bit processors, to sixteen-bit processors, to 32-bit processors, to 64-bit processors. How long before we have 128-bit processors and enough bits for a multi-bit sign? Some compilers already support quad precision. Wouldn't necessarily need 128-bit CPUs.

And using said different notation just complicates the language for no purpose. The present convention of using a minus sign if the sign bit is set, and a plus sign (or nothing at all) if the sign bit is clear answers admirably. It tells the whole story.

How can you say that when a single bit offers only two states, but there are more than two possible outcomes?

At least all of the story that the I/O library knows without arbitrarily making up information out of the air!

There's more to the matter than just the I/O library.

Re: On writing negative zero – with or without sign

That includes a correct indication of the sign of the value being printed. This is a very simple concept: the I/O library can't discern the meaning of the value or whether the sign is meaningful or not.

Precisely. So in your thermometer example, where the sign isn't meaningful, the processor doesn't know that, thus it has a fifty percent chance of getting it wrong by being forced to display a sign.

And it has a 100% chance of being wrong if it claims that the answer is *exact* zero.

But you claimed that there is no such thing. Although I disagree, that doesn't mean I believe the thermometer example represents a case of an exact zero.

That's not an improvement.

Irrelevant, given that I never said it would be an improvement.

The existing implementation is writing all zero digits for the value already. To anyone with an understanding of float, that means the value (the result of a subtract), by itself, has no significance at all, much less does the sign mean anything.

So a single sign bit isn't sufficient.

You keep claiming that the answer is being written wrong.

Incorrect. I said that the answer has a fifty percent chance of being wrong.

It isn't, you're just interpreting it wrong.

Ironically, you interpreted my statement wrong.

Re: On writing negative zero – with or without sign

Re: On writing negative zero – with or without sign

If you insist on interpreting a notational convention differently than it was intended, you'll continue to make mistakes.

Unfortunately, the compiler doesn't spit out the intention of notational conventions along with the numerical results. Do you have any Fortran compiler documentation that explains how signs are handled?

As I said, you can insist that when I say "tea" it doesn't mean a beverage: you'll usually be wrong.

Irrelevant, given that I haven't done so.

The user is frequently different from the programmer. It's up to the programmer to supply sufficient information so that the user has what he needs.

Then by gosh the programmer can write a string of plusses and minuses or draw a picture of a clown if (s)he wants.

Imagine that, extra bits to work with. Do the propagation of error computation, show what the uncertainty in the result is, and then you don't have to mess with the interpretation of a sign. I guess that's "magic" to you.

Just don't lie to the later reader by pretending it to be the actual report of the actual calculation. Such an actual report can already be made clearly by the output convention usually recommended.

I disagree. A single bit has only two states, but there are more than two possible outcomes. Doing a propagation of error computation carries far more information.

What a responsible programmer will do is label the output. Something like: "first differences of temperature readings".

You think that's more responsible than a full propagation of

Re: On writing negative zero – with or without sign

Re: On writing negative zero – with or without sign

error computation? Of course, you have to do the computation properly. Too often quantization noise is ignored.

If the reader of the data doesn't know what first differences are, then such a reader won't make much sense of the data even if you have some bizarre notation introduced for the zeros. If he reader does know what first differences are, I doubt the sign on any zero values will be the least disconcerting.

Are you using that to try and justify that only two states are sufficient for a sign?

.