Subject: Re: machine precision
Posted by jameskuyper on Wed, 20 May 2009 01:41:51 GMT
View Forum Message <> Reply to Message

Jeff N. wrote:
...
> I think I have the beginnings of an understanding of this problem
> after having seen the topic pop up so much here in the newsgroup and
> after having read David's article several times. But now i'm
> wondering how other languages handle this situation? I've tried two
> other languages now, perl and VB, and they both seem to not give the
> "correct-but-not-expected" results that trip so many people up. For
> example, this snippet of perl code:
>
> $i = int(4.70*100);
> print "$i\n";
>
> prints 470 whereas the equivalent IDL code: print, fix(4.70*100)
> prints 469.

You've misunderstood the problem if you think that proves that the problem doesn't occur in perl. Just because perl gave you the result you expected in this case doesn't mean that it always will. When I typed 'man perlfun' on my home machine and searched for "int EXPR', I found:

"You should not use this function for rounding: one because it truncates towards 0, and two because machine representations of floating point numbers can sometimes produce counterintuitive results. For example, "int(-6.725/0.025)" produces -268 rather than the correct -269; that’s because it’s really more like -268.99999999999994315658 instead."

> Does anyone have any idea how other languages deal with the "sky is
> falling" problem?

The "sky is falling problem" is fundamentally a problem with programmer's expectations being out of line with the way floating point numbers actually work. The solution is eduction, there's really not a lot that the language can do about it. For this specific kind of problem, if you've got a value 'x' that should be a number close to an integer, but which might be slightly more or slightly less than the exact integer value, fix(x) is the wrong way to handle it. round() is more appropriate.