

Re: double money

Source: <http://coding.derkeiler.com/Archive/C/ CPP/comp.lang.c/2004-04/1801.html>

From: P.J. Plauger (pjp_at_dinkumware.com)

Date: 04/15/04

Date: Thu, 15 Apr 2004 04:16:22 GMT

"CBFalconer" <cbfalconer@yahoo.com> wrote in message
news:407E03D2.15466081@yahoo.com...

> *Keith Thompson wrote:*

> >> *Dan Pop wrote:*

> >>

> >>> *Floating point types can use *any* integer base greater or equal*

> >>> *to two.*

> >>

> *... snip ...*

> >

> > *But there's a good chance that a new form of decimal floating-point*

> > *may catch on in the not too distant future. It uses 10 bits for*

> > *each 3 decimal digits, so the mantissa representation is nearly*

> > *(1000/1024) as efficient as pure binary.*

>

> *That is not the point. Normalizing decimal values requires*

> *multiply/divide by 10, so the eventual range, at best, is from 1.0*

> *to 10.0- (9.999...999) for some value of the exponent. For base 2*

> *the corresponding range is from 1.0 to 2.0-. Besides the loss of*

> *a bit position the maximum possible percentage deviation is higher*

> *by 5 for the decimal organization. The extra bit makes that a*

> *factor of 10.*

Uh, that's a bit arm waving, but there is indeed a loss of effective precision in decimal due to this "wobbling precision". It's less bad than base 16, which we've lived with for forty years.

> *If the normalization is by factors of 1000, as suggested by the 3*

> *dec. digits above, the situation is worse.*

But it's not. The digits are grouped by threes in the packed representation, but the exponent is to the nearest power of ten.

> *So you need a good reason to use any such decimal organization.*

Indeed, and several have been cited:

comp.lang.c: Re: double money

- way faster conversion between external decimal and internal form
- no roundoff problems in conversion, either way
- exact representation of decimal fractions (important for tax laws)
- way easier rounding to Nth decimal place
- $0.1 * 10.0 == 1.0$ (reducing reflector traffic)

People seldom notice the loss of a bit or two of precision -- witness all the truly awful math functions out there whose sins have gone unnoticed for decades. They *do* notice slow conversions, unexpected truncations, etc. as cited above.

P.J. Plauger
Dinkumware, Ltd.
<http://www.dinkumware.com>