Subject: Re: Simple MODULO question.
Posted by peter on Wed, 03 Jul 1996 07:00:00 GMT
View Forum Message <> Reply to Message

S. Penzes (penzes@dres.dnd.ca) wrote:
: It truly is a simple question and the online help doesn't seem to
: answer it. So it's either a bug or I have a short cicuit in a synapse
: someplace. Anyway here it is:

: Why does print,1.0 mod 0.25 return 0.00000 (as expected)
:   generally 1.0 mod (1.0/2^n) returns 0
: while   print,1.0 mod 0.2
: or   print,1.0 mod (1.0/5.0) return 0.200000
:   generally 1.0 mod (1.0 / n)

Probably because, in floating point, 0.25 is represented exactly, so
that 1.0/0.25 = 0.0, while 0.2 is not represented exactly. I'll guess
that 0.2 is actually something like 0.200000001, so that 5*0.2 is
greater than 1.0, while 1.0-4*0.2 is 0.199999996, which prints at 0.2.

Relying on true equality of floating point numbers as a test (which this
is doing, relying on 5.0*0.2=1.0) is, well, unreliable.

Peter

Peter Webb, HP Labs Medical Dept
E-Mail: peter_webb@hpl.hp.com
Phone: (415) 813-3756

Subject: Re: Simple MODULO question.
Posted by Peter Mason on Wed, 03 Jul 1996 07:00:00 GMT
View Forum Message <> Reply to Message

On 2 Jul 1996, S. Penzes wrote:
> Why does print,1.0 mod 0.25 return 0.00000 (as expected)
>   generally 1.0 mod (1.0/2^n) returns 0
> while   print,1.0 mod 0.2
> or   print,1.0 mod (1.0/5.0) return 0.200000
>   generally 1.0 mod (1.0 / n)
>
> In case you're wondering, I am trying to determine if:
> for x mod y whether x is an integer multiple of y.

I guess you're experiencing one of the pitfalls of working with floating-
point numbers - they're not necessarily exact.
I'd have to hazard a guess at exactly what's happening in this example:

. In the first case with 1.0 mod 0.25, both 1.0 and 0.25 (== 1/8) can be
  represented exactly in floating-point, and so 0.25 divides an integral
  number of times into 1.0 and the mod (remainder) is 0.
. In the second case with 1.0 mod 0.2, although 0.2 looks like a
  simple, exact number in base 10, it is a bit of a headache in base 2
  and can't be represented exactly - rather like 1/3 in base 10. My
  guess is that it's floating-point representation is fractionally
  larger than 0.2, and so 1.0 / "0.2" is fractionally less than 5.
  So 1.0 mod "0.2" = 5.0 - 4 * "0.2", which is fractionally less than 0.2.
  You can get a hint that there's a problem with: PRINT,1.0D/0.2. The
  result printed is 4.9999999. (This is a fluke of the print formatting, I
  think - both 1.0/0.2 and 1.0D/0.2D return 5!)

I think that you might have to use a steam-driven method for your test:
Instead of testing ((x mod y) eq 0.0), test:
 (abs(x - y*round(x/y)) lt some_small_tolerance)
or such.

Peter Mason
