Subject: Re: bizarre number transformation
Posted by Paul Van Delst[1] on Thu, 25 Jul 2002 15:50:13 GMT
View Forum Message <> Reply to Message

merle wrote:
>
> Hello,
>
> I ran into a number transformation error yesterday that is still
> confusing me this morning. At first I thought I was doing something
> silly with String() & StrTrim(), but then I wrote a little program
> (see huh.pro) with no conversions that still contained the problem.
> FYI, I'm using IDL Version 5.5 Win32 (x86).
>
> The problem is that the number 443496.984 is being turned into the
> number 443496.969 from basic assignments using Float() or Double(),
> despite the fact that even floats should easily be able to handle a
> number this large (floats can handle "ï¿½10^38, with approximately six
> or seven decimal places of significance").

Don't confuse the magnitude of the number with its precision. This all comes about because most floating point numbers can't be represented exactly in binary (I say most because numbers like 0.0 and 1.0 usually can. It's the stuff like 0.1 and 443496.984 that cause the heartache). Numbers are stored with a mantissa (describes the actual number you want) and an exponent (describes the, well, the exponent.). So, **assuming** that 7 significant figures is a definite "cutoff" for precision, then

1.1e+37 == 2 significant figures. Precision probably good to 1.100000e+37
1.01e+37 == 3 significant figures. Precision probably good to 1.010000e+37
....
1.000001e+37 == 7 significant figures. At the limit of precision
etc..

and for your number

```
6.984 == 4 sigfig, good to          6.984000
96.984 == 5 sigfig, good to         96.98400
496.984 == 6 sigfig, good to        496.9840
3496.984 == 7 sigfig, good to       3496.984
43496.984 == 8 sigfig, good to      43496.98
443496.984 == 9 sigfig, good to 443496.9
```

So, if the 7 sigfig assumption is a good one (may not be), then if you set a single precision value to 443496.984 and print it out, a result of 443496.9XX where the XX can be anything is perfectly reasonable.

> And, why doesn't x2=Double(443496.984) produce the correct
> result?

Because 443496.984 is, again, a *single precision* literal constant. That is, you're converting a single precision number (where the last 2 dp can be anything, see above) to a double precision one. You'll then have a very precise, "nearly correct" number. As others have pointed out you need to create the number as a double to begin with,

```
  x2 = 443496.984d0
```

so that you'll have, say, 16-17 significant figures, or a number "good" to 443496.9840000000.

I reckon that, if in doubt, *always* use double precision for floating point variables. There's nothing worse that trying to debug code and discovering weird results are related to the precision of the represetation (well, maybe apart from an insidious compiler bug, but that would nbever happen with IDL! :o)

> Could someone please explain this to me? Or at least duplicate this
> error so that I don't think that I'm going crazy?

You're definitely not bonkers. And, although it may not seem like it, you have discovered/learned something very important (at least I think so).

paulv

--
Paul van Delst
CIMSS @ NOAA/NCEP/EMC   Beer is good.
Ph: (301)763-8000 x7274   My wife.
Fax:(301)763-8545