Subject: Re: bizarre number transformation
Posted by wilms on Thu, 25 Jul 2002 12:11:18 GMT
View Forum Message <> Reply to Message

In article <b98a2b54.0207250346.381336bf@posting.google.com>,
merle <merlecorp@hotmail.com> wrote:
>
> 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

> or seven decimal places of significance").
>
> Since I knew that I had successfully read in numbers much greater than
> 443496.984 in the past, I created temp.dat with just the number
> 443496.984 in it, and read this into a variable, x3. If x3 is cast as
> a float, it doesn't work, i.e. the number is 443496.969. But, if x3
> is cast as a double, then it contains the correct value. Why isn't a
> float sufficient (443496.984 << 10^38 and contains only 3 decimal
> places)? And, why doesn't x2=Double(443496.984) produce the correct
> result?

There seems to be a misconception on what a "decimal place" is. In your
example, your number has 9 decimal places, not three.

x2=double(443496.984) casts a single precision variable to double. If
you want to declare your number as a double, write
x2=443496.984D0

Cheers,

Joern
>
> Here's the code & the output:
>
> ; -- --------------
> PRO huh
>
> ; -- print with single precision -> NFG
> X1 = Float(443496.984)
> Print, X1, Format='("X1 = ", (F10.3))'
>
> ; -- print with double precision -> NFG
> X2 = Double(443496.984)
> Print, X2, Format='("X2 = ", (F10.3))'
>
>
> ; -- read the number from a file & print it out -> works fine

```
> OpenR, lun, 'temp.dat', /Get_Lun, ERROR = err
>
> ; -- Note: X3 must be cast as a double or else the number becomes
> 443496.969
> X3 = Double(0)
>
> ReadF, lun, X3
> Print, X3, Format='("X3 = ", (F10.3))'
>
> Free_Lun, lun
>
> END
>
>
> IDL> .COMPILE "D:\IDL\workdir\huh.pro"
> % Compiled module: HUH.
> IDL> huh
> X1 = 443496.969
> X2 = 443496.969
> X3 = 443496.984
>
> ;  ---------------------------------------------------------- --------------
>
```

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

--
Dr. Joern Wilms   Linsenbergstr. 33
Univ. Tuebingen, Institute for Astronomy   D-72074 Tuebingen
Sand 1   (phone) +49 7071 29-76128
D-72076 Tuebingen, Germany   wilms@astro.uni-tuebingen.de

Subject: Re: bizarre number transformation
Posted by Don J Lindler on Thu, 25 Jul 2002 12:25:22 GMT

View Forum Message <> Reply to Message

>
> 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

> or seven decimal places of significance").
>

When you count the seven decimal places of significance, you must count the
all of the digits, not just the ones to the right of the decimal point.
443496 and 9 are the seven digits of significance.

> Since I knew that I had successfully read in numbers much greater than
> 443496.984 in the past, I created temp.dat with just the number
> 443496.984 in it, and read this into a variable, x3. If x3 is cast as
> a float, it doesn't work, i.e. the number is 443496.969. But, if x3
> is cast as a double, then it contains the correct value. Why isn't a
> float sufficient (443496.984 << 10^38 and contains only 3 decimal
> places)? And, why doesn't x2=Double(443496.984) produce the correct
> result?
>

When you execute x2=Double(443496.984), IDL recongnizes 443496.984 as a
single precision floating point constant. It is placed into a single
precision temporary variable prior to conversion to double. The precision
is
already lost before the conversion. What you want is a double precision
constant:

```
          x2 = 443496.984D0
```

Don

Subject: Re: bizarre number transformation
Posted by James Kuyper on Thu, 25 Jul 2002 13:51:14 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").

```
                    ^^^^^^^^^^^^^^^^^^^
```

> Since I knew that I had successfully read in numbers much greater than
> 443496.984 in the past, I created temp.dat with just the number
> 443496.984 in it, and read this into a variable, x3. If x3 is cast as
> a float, it doesn't work, i.e. the number is 443496.969. But, if x3
> is cast as a double, then it contains the correct value. Why isn't a
> float sufficient (443496.984 << 10^38 and contains only 3 decimal
> places)? And, why doesn't x2=Double(443496.984) produce the correct
> result?

The number of decimal places after the decimal point is irrelevant. What
matters is the number of significant digits in your number. 443496.984
has 9 significant digits. As you've found, 4-byte floating point numbers
are typically good for only about 7 significant digits. If you want more
precision than that, you'll have to use double precision variables.

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
```

Subject: Re: bizarre number transformation
Posted by Michael Ganzer on Thu, 25 Jul 2002 18:48:12 GMT
View Forum Message <> Reply to Message

As plenty postings already were dealing about how to use a double precision
number i wanted to ask u something different...

Whatever you do with 443496.984 in multiplication or something else.....
does it really matter at that number size if there is more than one digit
exact after the digit separator???
I mean.. specially in your case.... think about it and tell us maybe..

I just want to mention that before trying to waste memory with double
precision numbers, u probably could tell us what u wanna do.... For example
the circular magnitude calculated with a radius of four light years is exact
about 4 cm (1.5 inches) in relation to Pi using 31 compared to 32 digits
after the separator........ (i for myself already calculated Pi as exact as
2000 digits after the separator with a simple program... but it has no sense
for use really, does it?!)
And numeric approximations like in computer calculations often produce
higher mistake percentages than u expect...
And if u really need it for a thesis or something then u better do not
overdo it.... no professor will say "nice result" if u do not give a span of

And your "almost half a million" for a calculation already is a big
number...
Let's just say.. i am interested in what u r doing with that number ;)

Subject: Re: bizarre number transformation
Posted by Paul Van Delst[1] on Thu, 25 Jul 2002 19:22:26 GMT
View Forum Message <> Reply to Message

Michael Ganzer wrote:
>
> As plenty postings already were dealing about how to use a double precision
> number i wanted to ask u something different...
>
> Whatever you do with 443496.984 in multiplication or something else.....
> does it really matter at that number size if there is more than one digit
> exact after the digit separator???

My goodness. 443496.984 is not a "big" number. What if you have to add it to 0.004657?

<snip>

> And numeric approximations like in computer calculations often produce

> higher mistake percentages than u expect...

Usually due to people thinking computers can represent floating point numbers exactly. The most important word in your sentence is "approximations"

> And your "almost half a million" for a calculation already is a big
> number...

If the number represented dollars (or euros...which are probably worth more now :o), I agree. It's big.

If the number represents the number of water vapour molecules per metre^3 it's amazingly teeny tiny.

paulv

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

Subject: Re: bizarre number transformation
Posted by Paul Van Delst[1] on Thu, 25 Jul 2002 19:40:07 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
>
> There is such a thing as absolute TRUTH
> or BEAUTY. Why, consider IDL as an example.

Isn't that an oxymoron? :o)

paulv

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

Subject: Re: bizarre number transformation
Posted by David Fanning on Thu, 25 Jul 2002 19:40:43 GMT
View Forum Message <> Reply to Message

Paul van Delst (paul.vandelst@noaa.gov) writes:

> If the number represented dollars (or euros...which are probably worth more now :o), I
> agree. It's big.
>
> If the number represents the number of water vapour molecules per metre^3 it's amazingly
> teeny tiny.

Now, gentlemen, let's not get into the whole "relativism"
argument again. There is such a thing as absolute TRUTH
or BEAUTY. Why, consider IDL as an example.

Cheers,

David

P.S. Let's just say my object widget library is working
especially well today. :-)

--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: bizarre number transformation
Posted by James Kuyper on Thu, 25 Jul 2002 19:58:59 GMT

View Forum Message <> Reply to Message

Paul van Delst wrote:
>
> Michael Ganzer wrote:
>>
>> As plenty postings already were dealing about how to use a double precision
>> number i wanted to ask u something different...
>>
>> Whatever you do with 443496.984 in multiplication or something else.....
>> does it really matter at that number size if there is more than one digit
>> exact after the digit separator???
>
> My goodness. 443496.984 is not a "big" number. What if you have to add it to 0.004657?

The point is, that it's pretty rare to need that many significant
digits. There aren't many real-world numbers that can be measured to
within one part in a billion. Precision needs like that can come up in
intermediate steps of a calculation, (for instance, if you need to

calculate "sin(theta)-theta" for small values of theta), but that's
merely an indication that the calculation is badly organised (for small
theta, you can get more accurate results with the equivalent series
expansion: "-(theta^3)/6+(theta^5)/120-...")

However, having written a lot of such code, I've found that loss of
precision due to roundoff can sneak up on you far too easily. It's
almost always a lot faster (considering CPU time + developer time) to
use double precision. I save such tricks for the somewhat rarer cases
where double precision is inadequate.

Subject: Re: bizarre number transformation
Posted by David Fanning on Fri, 26 Jul 2002 04:02:09 GMT
View Forum Message <> Reply to Message

Paul van Delst (paul.vandelst@noaa.gov) writes:

> Don't confuse the magnitude of the number with its precision.

Every time I see this question I remember how remiss I am
at getting things posted to my web page. :-(

So, anyway, I finally got around to posting some of the better
answers to this question in my Tips section. Thanks to Paul,
and Liam, and William Clodius for helping us all more fully
appreciate this topic. There are some good links in the article
for those of you who find this interesting. :-)

http://www.dfanning.com/math_tips/sky_is_falling.html

Cheers,

David

--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: bizarre number transformation
Posted by Craig Markwardt on Fri, 26 Jul 2002 04:41:53 GMT
View Forum Message <> Reply to Message

James Kuyper <kuyper@gscmail.gsfc.nasa.gov> writes:

> Paul van Delst wrote:
>>
>> Michael Ganzer wrote:
>>>
>>> As plenty postings already were dealing about how to use a double precision
>>> number i wanted to ask u something different...
>>>
>>> Whatever you do with 443496.984 in multiplication or something else.....
>>> does it really matter at that number size if there is more than one digit
>>> exact after the digit separator???
>>
>> My goodness. 443496.984 is not a "big" number. What if you have to add it to 0.004657?
>
> The point is, that it's pretty rare to need that many significant
> digits. There aren't many real-world numbers that can be measured to
> within one part in a billion. Precision needs like that can come up in
> intermediate steps of a calculation, (for instance, if you need to
> calculate "sin(theta)-theta" for small values of theta), but that's
> merely an indication that the calculation is badly organised (for small
> theta, you can get more accurate results with the equivalent series
> expansion: "-(theta^3)/6+(theta^5)/120-...")
>
> However, having written a lot of such code, I've found that loss of
> precision due to roundoff can sneak up on you far too easily. It's
> almost always a lot faster (considering CPU time + developer time) to
> use double precision. I save such tricks for the somewhat rarer cases
> where double precision is inadequate.

Okay, I'll give a couple examples from my own needs:

 * absolute pulsar timing at the microsecond level, measured in Julian
   days, requires a fractional precision of 4d-13

 * the most stringent pulsar timing (not mine) requires better than
   100 cm positioning within the solar system, or 6 parts in 1d12

 * one can determine pulse frequencies of 400 Hz pulsars to a
   precision of 1 nHz, or 3 parts in 1d12

 * the most precise Doppler tracking of spacecrafts requires 2
   milliHertz precision using a carrier of 2 GHz, or 1 part in 1d12

Admittedly those are pretty specialized applications :-)

Most ordinary differential equations, especially if they are numerically stiff, require double precision.

Also, solving a curve fitting problem with MPFIT, where the parameters
vary in magnitude by more than one part in 1d7, will fail unless
double precision is used.

So, for me at least, double precision is the de facto choice for most
applications, unless the memory usage is prohibitive.

Craig

--
 -- --------------
Craig B. Markwardt, Ph.D.   EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
 -- --------------

Subject: Re: bizarre number transformation
Posted by R.Bauer on Fri, 26 Jul 2002 06:58:53 GMT

View Forum Message <> Reply to Message

Craig Markwardt wrote:
>
> James Kuyper <kuyper@gscmail.gsfc.nasa.gov> writes:
>
>> Paul van Delst wrote:
>>>
>>> Michael Ganzer wrote:
>>>>
>>>> As plenty postings already were dealing about how to use a double precision
>>>> number i wanted to ask u something different...
>>>>
>>>> Whatever you do with 443496.984 in multiplication or something else.....
>>>> does it really matter at that number size if there is more than one digit
>>>> exact after the digit separator???
>>>
>>> My goodness. 443496.984 is not a "big" number. What if you have to add it to 0.004657?
>>
>> The point is, that it's pretty rare to need that many significant
>> digits. There aren't many real-world numbers that can be measured to
>> within one part in a billion. Precision needs like that can come up in
>> intermediate steps of a calculation, (for instance, if you need to
>> calculate "sin(theta)-theta" for small values of theta), but that's
>> merely an indication that the calculation is badly organised (for small
>> theta, you can get more accurate results with the equivalent series
>> expansion: "-(theta^3)/6+(theta^5)/120-...")
>>

>> However, having written a lot of such code, I've found that loss of
>> precision due to roundoff can sneak up on you far too easily. It's
>> almost always a lot faster (considering CPU time + developer time) to
>> use double precision. I save such tricks for the somewhat rarer cases
>> where double precision is inadequate.
>
> Okay, I'll give a couple examples from my own needs:
>
> * absolute pulsar timing at the microsecond level, measured in Julian
>   days, requires a fractional precision of 4d-13
>
> * the most stringent pulsar timing (not mine) requires better than
>   100 cm positioning within the solar system, or 6 parts in 1d12
>
> * one can determine pulse frequencies of 400 Hz pulsars to a
>   precision of 1 nHz, or 3 parts in 1d12
>
> * the most precise Doppler tracking of spacecrafts requires 2
>   milliHertz precision using a carrier of 2 GHz, or 1 part in 1d12
>
> Admittedly those are pretty specialized applications :-)
>
> Most ordinary differential equations, especially if they are
> numerically stiff, require double precision.
>
> Also, solving a curve fitting problem with MPFIT, where the parameters
> vary in magnitude by more than one part in 1d7, will fail unless
> double precision is used.
>
> So, for me at least, double precision is the de facto choice for most
> applications, unless the memory usage is prohibitive.
>
> Craig
>
> --
> -- --------------
> Craig B. Markwardt, Ph.D.   EMAIL: craigmnet@cow.physics.wisc.edu
> Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
> -- --------------

Dear Craig,

thanks for these statements we do need double precision too.
I hope someone of INTEL will read sometimes this discussion,
because they have dropped double precision from their processors.

Reimar

--
Reimar Bauer

Institut fuer Stratosphaerische Chemie (ICG-I)
Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de
 -- -------
 a IDL library at ForschungsZentrum Juelich
 http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_lib_intro. html
 == =======

Subject: Re: bizarre number transformation
Posted by James Kuyper on Fri, 26 Jul 2002 13:47:01 GMT
View Forum Message <> Reply to Message

Reimar Bauer wrote:
...
> thanks for these statements we do need double precision too.
> I hope someone of INTEL will read sometimes this discussion,
> because they have dropped double precision from their processors.

Could you give some details on that? I could imagine INTEL producing
chips that don't have double precision, for use in contexts where the
extra precision isn't needed. I can't imagine them stopping production
of processors that do provide double precision.

Subject: Re: bizarre number transformation
Posted by James Kuyper on Fri, 26 Jul 2002 13:53:27 GMT
View Forum Message <> Reply to Message

Craig Markwardt wrote:
>
> James Kuyper <kuyper@gscmail.gsfc.nasa.gov> writes:
...
>> The point is, that it's pretty rare to need that many significant
>> digits. There aren't many real-world numbers that can be measured to
>> within one part in a billion. Precision needs like that can come up in

...
[some examples where such precision is needed]
> Admittedly those are pretty specialized applications :-)

Precisely. I also do scientific programming, where such needs are pretty common. But the vast bulk of the world's programming involves numbers that can be represented with adequate accuracy using single precision floating point. For instance, how many million-dollar quantities are actually measured with a precision of +/-$1? And how many people would actually care if such quantities were in error by $1 or two?