Subject: rounding errors
Posted by Dominic R. Scales on Fri, 27 Apr 2001 08:51:24 GMT
View Forum Message <> Reply to Message

HELP!

What gives? Is there any numerical math guy/gal out there
who can tell me how this happens? It seems to me, that
the accuracy of the second/third cast ist WAY off.

```
 a=double('2.56989')
 b=double( 2.56989 )
 c=double(float('2.56989'))

 print,a,b,c,format='(d)'

     2.5698900000000000 <---- this is what i want to have
     2.5698900222778320
     2.5698900222778320
```

So, the question is: why can the cast from a string give a much
more accurate result than a cast from a literal constant (or a
float variable, for that matter)? Do I really have to cast with:

```
d=double(2.56989*1000000L)/1000000.
print,d,format='(d)'

     2.5698900000000000
```

or even

```
d=double(string(2.56989))
print,d,format='(d)'

     2.5698900000000000
```

Ah yes, and while I'm at it... Have you ever compared
5.2e-6, 5.2*1e-6, 5.2*10*1e-7 ?

```
print, 5.2e-6, 5.2*1e-6, 5.2*10*1e-7, format='(e20.10)'
  5.2000000323e-06
  5.1999995776e-06
  5.2000000323e-06
```

Cheers,
Dominic

--

Dipl. Phys. Dominic R. Scales | Aero-Sensing Radarsysteme GmbH
Tel: +49 (0)8153-90 88 90 | c/o DLR Oberpfaffenhofen
Fax: +49 (0)8153-908 700 | 82234 Wessling, Germany
WWW: aerosensing.de | email: Dominic.Scales@aerosensing.de

Subject: Re: rounding errors
Posted by Dominic R. Scales on Wed, 02 May 2001 07:44:15 GMT
View Forum Message <> Reply to Message

Thanx alot to all of you,

this discussion has been very interesting and fruitful.
I had completely forgotten about human vs computer predjudice
regarding bases.
And, to add insult to injury, I forgot my own teaching that
"a computer is a dumb device. It does what you tell it, not
what you expect it to do".

Hm, just shows that one shouldn't spent 8h in a row at the
computer and then skip a nights sleep...

Special thanks to JD and Liam for the links, they make an
interesting reading, and William: should have thought of
looking at the binary and hex representation myself...

Dominic

--
Dipl. Phys. Dominic R. Scales | Aero-Sensing Radarsysteme GmbH
Tel: +49 (0)8153-90 88 90 | c/o DLR Oberpfaffenhofen
Fax: +49 (0)8153-908 700 | 82234 Wessling, Germany
WWW: aerosensing.de | email: Dominic.Scales@aerosensing.de

Subject: Re: rounding errors
Posted by Paul van Delst on Wed, 02 May 2001 16:46:55 GMT
View Forum Message <> Reply to Message

James Kuyper wrote:
>
> Paul van Delst wrote:
>>
>> Randall Skelton wrote:
>>>
>>> On Fri, 27 Apr 2001, Liam E. Gumley wrote:
>>>

>>>> This is a subtle but important point. DOUBLE() is a type conversion
>>>> function, and
>>>>
>>>> a = double(2.348339)
>>>>
>>>> shows a FLOAT argument being converted to a DOUBLE. The safest way to
>>>> 'cast' a double variable is
>>>>
>>>> a = 2.348339d
>>> [snip]
>>>
>>> Wow... I am glad that I have now learned that particular 'IDL feature'
>>> early on in my PhD. Just yesterday, I convinced the department that we
>>> really need a few good IDL programming books as the current
>>> 'learning-by-fire' approach could have some unfortunate consequences ;)
>>
>> This "feature" has absolutely *nothing* to do with IDL. The same thing occurs in other
>> languages, e.g. Fortran, C, etc. Floating point numbers, in general, cannot be represented
>> exactly and you have to keep that in mind when writing code
>
> I think you're misunderstanding the "feature" of IDL that surprised me
> as much as it surprised Randall and Liam. This has everything to do with
> IDL, and nothing to do with expecting exact representation of a
> finite-length decimal fraction. By default, in C 2.348339 represents a
> double precision number, not a single precision one, and I'd never
> realized that the IDL convention was different.

Ahh, I see. I guess it depends on what you started with. I code in Fortran mostly so when I think "default floating point number" I think single-precision. In Fortran at least (and by association IDL??), that convention _probably_ grew out of memory limitations of computers and whatnot back in olden day. Nowadays it (mostly) doesn't matter I guess.

I wonder what other languages use as a default? (e.g. Matlab sticks everything in double doesn't it? Probably strings as well.... :o)

paulv

--
Paul van Delst A little learning is a dangerous thing;
CIMSS @ NOAA/NCEP Drink deep, or taste not the Pierian spring;
Ph: (301)763-8000 x7274 There shallow draughts intoxicate the brain,
Fax:(301)763-8545 And drinking largely sobers us again.
Alexander Pope.

Subject: Re: rounding errors
Posted by Jaco van Gorkom on Fri, 04 May 2001 13:17:54 GMT

Paul van Delst wrote:
> James Kuyper wrote:
>> Paul van Delst wrote:
>>> Randall Skelton wrote:
>>>> On Fri, 27 Apr 2001, Liam E. Gumley wrote:
>>>>
>>>> > This is a subtle but important point. DOUBLE() is a type conversion
>>>> > function, and
>>>> >
>>>> > a = double(2.348339)
>>>> >
>>>> > shows a FLOAT argument being converted to a DOUBLE. The safest way
to
>>>> > 'cast' a double variable is
>>>> >
>>>> > a = 2.348339d
>>>> [snip]
>>>>
>>>> Wow... I am glad that I have now learned that particular 'IDL
feature'
>>>> early on in my PhD.

>> [snip] the "feature" of IDL that surprised me
>> as much as it surprised Randall and Liam. This has everything to do with
>> IDL, and nothing to do with expecting exact representation of a
>> finite-length decimal fraction. By default, in C 2.348339 represents a
>> double precision number, not a single precision one, and I'd never
>> realized that the IDL convention was different.
>
> Ahh, I see. I guess it depends on what you started with. I code in Fortran
mostly so when
> I think "default floating point number" I think single-precision. In
Fortran at least (and
> by association IDL??), that convention _probably_ grew out of memory
limitations of
> computers and whatnot back in olden day. Nowadays it (mostly) doesn't
matter I guess.
>
> I wonder what other languages use as a default? (e.g. Matlab sticks
everything in double
> doesn't it? Probably strings as well.... :o)

I was thinking, wouldn't the logical IDL behaviour be to take whatever data
type is necessary to hold a constant? I mean, it works that way with integer
constants:

IDL> help, 17, 33000, 2150000000
<Expression> INT = 17
<Expression> LONG = 33000
<Expression> LONG64 = 2150000000

Obviously IDL does not do the same type of overflow checking for floating-point constants:

IDL> help,1e38, 5e38, 5d38
<Expression> FLOAT = 1.00000e+038
<Expression> FLOAT = Inf
<Expression> DOUBLE = 5.0000000e+038
(note that no arithmetic error is thrown in this case, since no arithmetic is being done, really.)

When specifying more decimal places for a constant, the choice between integer and float data types is made automagically:

IDL> help, 2, 2.0
<Expression> INT = 2
<Expression> FLOAT = 2.00000

Logical behaviour then would seem that constants with more than, say, six decimal places are interpreted as double-precision floats. 'Logical' in an IDL sort of way: if you don't explicitly state the data type you want for a constant, then you'll get something which is at least capable of holding it.

Just my thoughts on what could have been... It would have made life for especially inexperienced users less tricky (together with smarter data typing of FOR-loop counters, easier printing, and what have we). One practical survival strategy in IDL: always explicitly declare the data type of constants (by appending L, LL, D, ..).

Jaco

Subject: Re: rounding
Posted by Amar Nayegandhi on Wed, 04 Sep 2002 14:40:53 GMT
View Forum Message <> Reply to Message

Hi,
Sorry for this easy question... the number should have been 'double', but it was 'float'.
-amar

On Wed, 4 Sep 2002, Amar Nayegandhi wrote:

> Hi,
> IDL seems to round off large decimal numbers. For e.g.,
> IDL> x = 284.766117
> IDL> print, format='(I11)',x*1000000L
> 284766112
>
> IDL> x = 284.766119
> IDL> print, format='(I11)',x*1000000L
> 284766112
>
> Is there any way around this?
>
> Thanks,
> -amar
>
>
> **********
> Amar Nayegandhi
> **********
>
>

Subject: Re: rounding
Posted by Craig Markwardt on Wed, 04 Sep 2002 14:45:00 GMT
View Forum Message <> Reply to Message

Amar Nayegandhi <anayegan@csee.usf.edu> writes:
> Hi,
> IDL seems to round off large decimal numbers. For e.g.,
> IDL> x = 284.766117
> IDL> print, format='(I11)',x*1000000L
> 284766112
>
> IDL> x = 284.766119
> IDL> print, format='(I11)',x*1000000L
> 284766112
>
> Is there any way around this?

Yes, use double precision.

x = 284.766117D

Of course, that only gets you out to 17 digits of precision or so.
Normal FLOATs keep about 7 digits.

Craig

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

Subject: Re: rounding
Posted by David Fanning on Wed, 04 Sep 2002 14:52:08 GMT
View Forum Message <> Reply to Message

Amar Nayegandhi (anayegan@csee.usf.edu) writes:

> IDL seems to round off large decimal numbers. For e.g.,
> IDL> x = 284.766117
> IDL> print, format='(I11)',x*1000000L
> 284766112
>
> IDL> x = 284.766119
> IDL> print, format='(I11)',x*1000000L
> 284766112
>
> Is there any way around this?

Here is an article that might shed some light on this:

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: rounding
Posted by thompson on Wed, 04 Sep 2002 14:59:05 GMT
View Forum Message <> Reply to Message

Amar Nayegandhi <anayegan@csee.usf.edu> writes:

> Hi,
> IDL seems to round off large decimal numbers. For e.g.,
> IDL> x = 284.766117
> IDL> print, format='(I11)',x*1000000L
> 284766112

> IDL> x = 284.766119
> IDL> print, format='(I11)',x*1000000L
> 284766112

> Is there any way around this?

This isn't IDL, it's just the way binary computers work. To get around it, use double precision instead of single precision.

```
IDL> x = 284.766117d0
IDL> print, format='(I11)',x*1000000L
  284766117
```

Another way to look at what's going on is to print out the floating point numbers to high precision.

```
IDL> x = 284.766117
IDL> print, format='(F22.11)',x
      284.76611328125
IDL> x = 284.766117d0
IDL> print, format='(F22.11)',x
      284.76611700000
IDL> print, format='(F27.16)',x
      284.7661170000000100
```

It isn't that IDL rounds off the numbers, it's that some precision is lost in converting back and forth between the decimal notation that people use and the binary notation that computers use internally. Double precision numbers have the same problem, as shown above, only not so bad.

William Thompson
