Subject: Recognizing double precision?
Posted by wlandsman on Fri, 05 Oct 2007 15:12:40 GMT
View Forum Message <> Reply to Message

About once a year I receive a complaint about my code because someone inputs a Julian date like this

IDL> jd = 2441636.1

and then gets mysterious results because the value of jd is "truncated"

IDL> print, jd, f='(f10.2)'
2441636.00

So it would it be reasonable to request that the IDL compiler recognize a number as double precision, if it has too many digits to be stored as a floating point number? After all, IDL does do something like this (in default mode) for short and long integers:

IDL> a = 32767 & help,a
A INT = 32767
IDL> a = 32768 & help,a
A LONG = 32768

I can't imagine how adding this capability would break existing code.

Does anyone know if other interpreted languages can recognize a double precision number when they encounter one? Thanks, --Wayne

Subject: Re: Recognizing double precision?
Posted by edward.s.meinel@aero. on Tue, 09 Oct 2007 15:11:24 GMT
View Forum Message <> Reply to Message

On Oct 8, 9:18 am, wlandsman <wlands...@gmail.com> wrote:
> Bob, Bringfried,
>
>> Well, to play devil's advocate, the programmer should not let the
>> user input single precision data in such a case.
>
> Except that single precision or even a long integer (e.g. 2441636)
> *could* be valid, if the user was not interested in fractional
> days. I agree that it is probably safest to force the user to
> input double precision, but this does not feel like the "IDL way".
> Thanks, --Wayne

Actually, it gets rounded:

IDL> jd = 2441636.1
IDL> print, jd, f='(f10.2)'
2441636.00
IDL> jd = 2441636.9
IDL> print, jd, f='(f10.2)'
2441637.00

You're right, that is not a good thing. Mathematica has been doing it
correctly for a long time, why can't IDL?

Ed M

Subject: Re: Recognizing double precision?
Posted by Maarten[1] on Tue, 09 Oct 2007 15:24:04 GMT
View Forum Message <> Reply to Message

On Oct 9, 3:11 pm, "edward.s.mei...@aero.org" <mei...@aero.org> wrote:
> On Oct 8, 9:18 am, wlandsman <wlands...@gmail.com> wrote:
>
>> Except that single precision or even a long integer (e.g. 2441636)
>> *could* be valid, if the user was not interested in fractional
>> days. I agree that it is probably safest to force the user to
>> input double precision, but this does not feel like the "IDL way".
>> Thanks, --Wayne
>
> Actually, it gets rounded:
>
> IDL> jd = 2441636.1
> IDL> print, jd, f='(f10.2)'
> 2441636.00
> IDL> jd = 2441636.9
> IDL> print, jd, f='(f10.2)'
> 2441637.00
>
> You're right, that is not a good thing. Mathematica has been doing it
> correctly for a long time, why can't IDL?

Well, we have
  compile_opt defint32
what I want is
  compile_opt defdouble
to have floating point constants on the command line and in code be in
double precision by default. If needed, one can use float() to
explicitly downcast a double, the other way round is impossible.

Best,

Maarten

Subject: Re: Recognizing double precision?
Posted by R.G.Stockwell on Tue, 09 Oct 2007 15:50:30 GMT
View Forum Message <> Reply to Message

"wlandsman" <wlandsman@gmail.com> wrote in message
news:1191597160.614557.153160@50g2000hsm.googlegroups.com...
> About once a year I receive a complaint about my code because someone
> inputs a Julian date like this
>
> IDL> jd = 2441636.1

btw, this may be obvious to all, but you can force the input
with a read command, read it as string, and cast it to double.

IDL> .GO

: 2441636.1

S STRING = ' 2441636.1'

2441636.1

single:

2441636.00000

double

2441636.10000

Cheers,
bob

Subject: Re: Recognizing double precision?
Posted by edward.s.meinel@aero. on Wed, 10 Oct 2007 14:07:45 GMT
View Forum Message <> Reply to Message

On Oct 9, 11:50 am, "R.G. Stockwell" <noem...@please.com> wrote:

> "wlandsman" <wlands...@gmail.com> wrote in message
>
> news:1191597160.614557.153160@50g2000hsm.googlegroups.com...
>
>> About once a year I receive a complaint about my code because someone
>> inputs a Julian date like this
>
>> IDL> jd = 2441636.1
>
> btw, this may be obvious to all, but you can force the input
> with a read command, read it as string, and cast it to double.
>
> IDL> .GO
>
> : 2441636.1
>
> S STRING = ' 2441636.1'
>
> 2441636.1
>
> single:
>
> 2441636.00000
>
> double
>
> 2441636.10000
>
> Cheers,
> bob

Sure, you _could_ do that, but that is even worse than

IDL>jd = 2441636.1d

The point is that IDL should be smart enough to cast the variable into
the correct type automatically. After all, it works for integers:

y = 1
INT

y = 123456
LONG

y = 12345678901234
LONG64

Ed

Subject: Re: Recognizing double precision?
Posted by R.G.Stockwell on Wed, 10 Oct 2007 15:53:10 GMT
View Forum Message <> Reply to Message

<meinel@aero.org> wrote in message
news:1192025265.336346.261400@o3g2000hsb.googlegroups.com...
> On Oct 9, 11:50 am, "R.G. Stockwell" <noem...@please.com> wrote:
>> "wlandsman" <wlands...@gmail.com> wrote in message
>>
>> news:1191597160.614557.153160@50g2000hsm.googlegroups.com...
>>
>>> About once a year I receive a complaint about my code because someone
>>> inputs a Julian date like this
>>
>>> IDL> jd = 2441636.1
>>
>> btw, this may be obvious to all, but you can force the input
>> with a read command, read it as string, and cast it to double.
>>
>> IDL> .GO
>>
>> : 2441636.1
>>
>> S STRING = ' 2441636.1'
>>
>> 2441636.1
>>
>> single:
>>
>> 2441636.00000
>>
>> double
>>
>> 2441636.10000
>>
>> Cheers,
>> bob
>
> Sure, you _could_ do that, but that is even worse than
>
> IDL>jd = 2441636.1d

Not worse, just different. The problem is that we are talking about
different things. One is a user interface, in which the developer should
be responsible for inputting the correct variable type. (thus, read string,
and then do error checking, range checking, valid input checking and casting
to the appropriate type). Because one thing is certain, if a user _can_
do something that will crash the code, then the user _will_do something

to crash the code (I've even run into malicious users who try to crash
the code so they can stop working.)

The second, above, is input at the IDL command line. That is, in my
opinion, identical to a line of code in a program. If the developer types in
jd = 2441636.1
in their routine when they should have made it a double, then that is a
programming error.
 It is entirely the programmers responsibility to have the correct type for
their variables.

I don't think the compiler should take the defined floating point variable
and force it to double unless the programmer tells it to (either explicitly
or implicitly). Just my opinion anyway.

Cheers,
bob

Subject: Re: Recognizing double precision?
Posted by edward.s.meinel@aero. on Wed, 10 Oct 2007 18:21:52 GMT
View Forum Message <> Reply to Message

On Oct 10, 11:53 am, "R.G. Stockwell" <noem...@please.com> wrote:
> <mei...@aero.org> wrote in message
>
> news:1192025265.336346.261400@o3g2000hsb.googlegroups.com...
>
>
>
>> On Oct 9, 11:50 am, "R.G. Stockwell" <noem...@please.com> wrote:
>>> "wlandsman" <wlands...@gmail.com> wrote in message
>
>>> news:1191597160.614557.153160@50g2000hsm.googlegroups.com...
>
>>>> About once a year I receive a complaint about my code because someone
>>>> inputs a Julian date like this
>
>>>> IDL> jd = 2441636.1
>
>>> btw, this may be obvious to all, but you can force the input
>>> with a read command, read it as string, and cast it to double.
>
>>> IDL> .GO
>
>>> : 2441636.1

>
>>> S STRING = ' 2441636.1'
>
>>> 2441636.1
>
>>> single:
>
>>> 2441636.00000
>
>>> double
>
>>> 2441636.10000
>
>>> Cheers,
>>> bob
>
>> Sure, you _could_ do that, but that is even worse than
>
>> IDL>jd = 2441636.1d
>
> Not worse, just different. The problem is that we are talking about
> different things. One is a user interface, in which the developer should
> be responsible for inputting the correct variable type. (thus, read string,
> and then do error checking, range checking, valid input checking and casting
> to the appropriate type). Because one thing is certain, if a user _can_
> do something that will crash the code, then the user _will_do something
> to crash the code (I've even run into malicious users who try to crash
> the code so they can stop working.)
>
> The second, above, is input at the IDL command line. That is, in my
> opinion, identical to a line of code in a program. If the developer types in
> jd = 2441636.1
> in their routine when they should have made it a double, then that is a
> programming error.
>   It is entirely the programmers responsibility to have the correct type for
> their variables.
>
> I don't think the compiler should take the defined floating point variable
> and force it to double unless the programmer tells it to (either explicitly
> or implicitly). Just my opinion anyway.
>
> Cheers,
> bob

Oh, I absolutely agree with the GUI issue. I do the same thing you do.

The thing is that Wayne was complaining about command line
inconsistency, not a problem with his routine. IDL correctly assigns

integer types, but not floating point types. Once you type something on the command line and it gets assigned a variable type, if it is the wrong one, you're hosed. Example:

IDL>jd = 2441636.1
IDL>result = WL_ROUTINE(jd)

IDL just changed the value of jd without notifying the user. Do you consider that good programming practice? The programming error is on IDL's part, not Wayne.

Ed

Subject: Re: Recognizing double precision?
Posted by R.Bauer on Wed, 24 Oct 2007 18:15:33 GMT
View Forum Message <> Reply to Message

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

wlandsman schrieb:
> About once a year I receive a complaint about my code because someone
> inputs a Julian date like this
>
> IDL> jd = 2441636.1
>
> and then gets mysterious results because the value of jd is
> "truncated"
>
> IDL> print, jd, f='(f10.2)'
> 2441636.00
>
> So it would it be reasonable to request that the IDL compiler
> recognize a number as double precision, if it has too many digits to
> be stored as a floating point number? After all, IDL does do
> something like this (in default mode) for short and long integers:
>
> IDL> a = 32767 & help,a
> A INT = 32767
> IDL> a = 32768 & help,a
> A LONG = 32768
>
> I can't imagine how adding this capability would break existing code.
>
> Does anyone know if other interpreted languages can recognize a double
> precision number when they encounter one? Thanks, --Wayne
>

I would prefer a good working function which does get the minimum type
needed to represent the data. That would help on an other issue too.
e.g. you could then read data type dependent into structures.

This function could be used for the compiler too but it should not be
hidden from the user.

cheers
Reimar

-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.5 (GNU/Linux)
Comment: Using GnuPG with SUSE - http://enigmail.mozdev.org

iD8DBQFHH4vD5aOc3Q9hk/kRAlVkAKC0mr00iPkpCKDKqJZXLioBT6IdWACg qRO8
5vQL005oOUN8ch0Tf5rYQps=
=Z4Fu
-----END PGP SIGNATURE-----