Subject: Re: Basic Math: I _hope_ this is a stupid question
Posted by David Fanning on Sun, 24 Apr 2005 21:48:42 GMT
View Forum Message <> Reply to Message

Ed Hyer writes:

> IDL> print,288*180
> -13696
> IDL> print,long(288*180)
> -13696
> IDL> print,ulong64(288*180)
> 18446744073709537920
> IDL> print,288.*180
> 51840.0
>
> I just want to multiply two integers! Is that so wrong?

Here are a couple of articles for you:

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

Where you see "double" you should read "long integer", and where
you see "float" you should read "integer".

Basically, casting a short multiplication (288*180) to a
long *after* you do the arithmetic, doesn't gain you
anything. You have to do it *before* you to the arithmetic. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Basic Math: I _hope_ this is a stupid question
Posted by btt on Mon, 25 Apr 2005 12:44:46 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> Ed Hyer writes:
>
>

>> IDL> print,288*180
>>   -13696
>> IDL> print,long(288*180)
>>   -13696
>> IDL> print,ulong64(288*180)
>>   18446744073709537920
>> IDL> print,288.*180
>>   51840.0
>>
>> I just want to multiply two integers! Is that so wrong?
>
>
> Here are a couple of articles for you:
>
>   http://www.dfanning.com/math_tips/sky_is_falling.html
>   http://www.dfanning.com/math_tips/double.html
>
> Where you see "double" you should read "long integer", and where
> you see "float" you should read "integer".
>
> Basically, casting a short multiplication (288*180) to a
> long *after* you do the arithmetic, doesn't gain you
> anything. You have to do it *before* you to the arithmetic. :-)
>

Hi,

You might also consider using the COMPILE_OPT compile flag DEFINT32 at
the beginning of your routines. I have taken Mark Hadfield's
recommendation to heart and now use COMPILE_OPT IDL2 pretty much everywhere.

Cheers,
Ben

Subject: Re: Basic Math: I _hope_ this is a stupid question
Posted by Geoff Cureton on Wed, 27 Apr 2005 16:26:05 GMT
View Forum Message <> Reply to Message

Ed Hyer wrote:

> IDL> print,288*180
>   -13696
> IDL> print,long(288*180)
>   -13696
> IDL> print,ulong64(288*180)
>   18446744073709537920
> IDL> print,288.*180

> 51840.0
>
> I just want to multiply two integers! Is that so wrong?

What about appending "L" to the integers to make sure they are of type
long integer. Thus...

```
IDL> print,288L*180L
      51840
```

Like someone wrote, your ints must be cast as long int *before* the
computation, so it is done with long int arithmetic. Trying to cast
the result as long int afterwards won't work, the damage is done :-)

Hope this helps,
Geoff

Subject: Re: Basic Math: I _hope_ this is a stupid question
Posted by MarioIncandenza on Wed, 27 Apr 2005 23:14:33 GMT
View Forum Message <> Reply to Message

Thanks all. I had encountered this before, and gotten used to writing
for i=0l,bignumber (which I do a lot less of since I started read
comp.lang.idl-pvwave), but I never knew exactly where the threshold
was, and always sort of assumed it was somewhere up there, certainly
not 32768. I guess that's why integer math is fast.

Subject: Re: Basic Math: I _hope_ this is a stupid question
Posted by K. Bowman on Thu, 28 Apr 2005 13:15:59 GMT
View Forum Message <> Reply to Message

In article <1114641742.781488.240230@f14g2000cwb.googlegroups.com>,
"Ed Hyer" <ejhyer@gmail.com> wrote:

> Thanks all. I had encountered this before, and gotten used to writing
> for i=0l,bignumber (which I do a lot less of since I started read
> comp.lang.idl-pvwave), but I never knew exactly where the threshold
> was, and always sort of assumed it was somewhere up there, certainly
> not 32768. I guess that's why integer math is fast.

I think the best available approach is to include

COMPILE_OPT IDL2

in all functions and procedures, and your startup.pro. That makes LONG the

default integer type and requires the use of square brackets for array subscripting (to avoid confusion with function references).

Ken Bowman
